A Case Study of Taking AP Computer Science Principles (AP CSP): A Student's Perspective

Sarah Cameron
University of West Florida, USA

Tony Pham
University of West Florida, USA

Sikha Bagui
University of West Florida, USA

www.ijonest.net

To cite this article:

Cameron, S., Pham, T., & Bagui, S. (2024). A case study of taking ap computer science principles (AP CSP): A student's perspective. *International Journal on Engineering, Science, and Technology (IJonEST), 6*(1), 1-17. https://doi.org/10.46328/ijonest.210

International Journal on Engineering, Science, and Technology
2024, Vol. 6, No. 1, 1- 17 https://doi.org/10.46328/ijonest.210

A Case Study of Taking AP Computer Science Principles (AP CSP): A Student's Perspective

Sarah Cameron, Tony Pham, Sikha Bagui

Article Info

Article History

Received:
28 July 2023
Accepted:
11 December 2023

Keywords

Computer science education
High schools
AP computer science
Principles (AP CSP)
Code.org
K-12

Abstract

With the increased demand for computer science degrees in the work force, computer science is becoming more prominent in high schools. AP Computer Science Principles (AP CSP) is a course that serves as a bridge into computer science. Code.org provides a year long curriculum for this AP course to be led by teachers in the classroom. Beyond an analysis of the pass rates of students, and with the recency of the AP CSP course, a reflection of the AP CSP curriculum from the student's perspective, is in order. This study breaks down the strengths and weaknesses of AP CSP from a student's perspective. Results show there are many strengths in relation to the Code.org curriculum. However, AP CSP can be a little challenging in motivating and engaging students if not executed properly by the teacher.

Introduction

The job market is ever changing and one field that has seen tremendous growth and change is the computing sector. According to the U.S. Bureau of Labor Statistics, computer and information technology occupations were projected to grow 15 percent from 2021 to 2023. Additionally, as the U.S. Bureau of Labor Statistics stated, as of May 2021, computer and information technology occupations had a median annual wage of $97,430, over double the median annual wage of $45,760 for all occupations (U.S. Bureau of Labor Statistics, 2022). With the projected increase in jobs in the field, the number of computer and information technology graduates needs to increase as well, and this, however, is not happening. For example, in the state of Florida (USA), there was an average of 28,088 open computing jobs each month in the year 2022. This exists in direct contrast with only 3,808 graduates in computer science in 2019 (Code.org, 2022). This gap in the job market suggests a need for earlier introduction of computer science to students at the high school level, in order to encourage more computer science majors.

Many states, counties, and schools in the US recognize this need for high school computer science and the percentage of schools offering a computer science course has been steadily growing over the past few years. In the 2022 report on the state of computer science education by the Code.org Advocacy Coalition, 53% of U.S. high schools offered a foundational computer science course, up 2% from the previous year. Also, the report highlights that this increased offering is paired with increased participation by students, with 5.6% of high school students being enrolled in a foundational computer science course, up from 4.7% the previous year. Despite the increased

participation by students, stats show that disparities still exist with underrepresented groups being less likely to take computer science courses. This is seen by the fact that only 32% of young women participate in a computer science course and that Hispanic populations are 1.5 times less likely to enroll than their white and Asian peers (Code.org, CSTA, & ECEP Alliance, 2022). There have been efforts on the state and local levels to combat this disparity, but one of the largest attempts to increase underrepresented representation in computer science education in recent years has been with the introduction of the Advanced Placement (AP) Computer Science Principles (CSP).

Though there are quite a few studies that analyze the success of AP CSP from the success rate of students, there are no works that present a reflection of the AP CSP curriculum from a student's perspective. Hence, the novelty of this paper is that it presents a reflection of the AP CSP curriculum from the student's perspective. This work will also be a valueable resource to teachers who are starting to teach AP CSP. It will help the teachers address, in advance, the shortcomings that the students typically face when taking this course.

The rest of this paper is organized as follows. Section 2 presents the background and related works; section 3 presents an overview of the AP CSP course; section 4 pesents an overview of Code.org; section 5 presents an analysis in terms of strenghts and weaknesses of the AP CSP experience as well as reflecting on the strenghts and weaknesses of Code.org as it relates to the AP CSP curriculum; and section 6 presents the conclusions.

Background and Related Works

The AP Computer Science Principles (AP CSP) course was created with the intent of making computing more accessible to underrepresented groups but has shown mixed results. The AP CSP exam was first offered officially during the AP 2017 exam season. College Board, following the implementation of the course in its catalog, began analyzing the demographic statistics of students taking the course. This analysis focuses on the pipeline of AP CSP into both future AP STEM courses and majoring in STEM related fields. The study concluded that the course results in an increase in participation of underrepresented groups in AP courses, with the course being the first AP STEM course for over fifty percent of students. Additionally, those students who take AP CSP were more likely to major in a STEM-related major (Wyatt, Feng, & Ewing, 2020).

These conclusions, however, contrast with the outcome seen in other reports. In a paper analyzing AP CSP and AP Computer Science A (AP CSA) students, the results were not nearly as high praising as seen in College Board's report. Although there were more underrepresented groups taking AP CSP as compared to AP CSA in the 2017 exam cycle, other factors of analysis saw less fruitful results. Within the report, AP CSP alone was not a pipeline into a computer science major as 16.9% (AP CSP only) of students declared such a major compared to 28.3% (AP CSA only) and 38.1% (both courses). Additionally, the report critiques the AP CSP course for not putting its students in a position to be prepared for college-level computing majors in terms of the acceptance of course credit by colleges as prerequisite course credit (Sax, et al., 2020).

Few reports exist on smaller scale implementations of AP CSP. The state of Maryland's public-school landscape

was analyzed in a report in 2019. Maryland, between the first and second years of offering AP CSP saw declines in the offerings of AP CSA with no increases in the offerings of AP CSP. The state saw an overall decrease in the number of AP CS classes offered by 3% with a drop in the number of schools offering AP CSA from 50% to 38% in just one school year cycle. The increasing number of schools offering AP CSP was at the expense of the AP CSA courses taught at many schools in the state (Killen, Weintrop, & Garvin, 2019). Due to the mixed results on the effectiveness of AP CSP on increasing diversity in computing as well increasing the offerings of computer science courses, there have been reports analyzing the difference in approaches of the course on the increasing diversity metric. Meur's article (2022) considers four approaches to teaching AP CSP and evaluates their effectiveness through four criteria (exam results, self-efficacy and confidence, belongingness and identity, persistence and interest). Through her studies, it was found that the most effective models were the social approach of supporting students through peer learning communities (which saw gains in all four reported categories) and the curricular approach of finding the beauty and joy in computing (which saw gains in all categories except persistence and interest). The social approach's success stemmed from presenting students with the diversity of the field and continued connection between students following the five-day long course. The curricular approach's success stemmed from using a visual approach to learning the language via block coding making the course more approachable (Meur, 2022). There are limitations to the report however as they analyze improvements on smaller scale samples with the implementations of the cataloged approaches.

There are two important perspectives to consider when analyzing the course and computer science education in general, the teachers and the students. In relation to the former, teachers experience a plethora of hurdles. A paper collecting the interviews of 24 teachers highlights many of the hurdles being in both the teaching of the material and external factors. Issues with the understanding of the content and how to assess students alongside the lack of preparation or experience in the field is a common thread highlighted in the article. Many computer science teachers (15 of the 24 interviewed) struggled with pedagogical challenges like keeping students engaged and providing an enriching classroom experience (Yadav, Gretter, Hambrusch, & Sands, 2016).

On the flip side of the coin is the student's perspective. Little research exists on student experience specifically with the AP CSP course although similar courses have available data to discuss. In the city of Chicago, a report on the Taste of Computing project was compiled. Within this report, it was found that the implementation of the course saw high perceived value of the course by students alongside increased awareness of the career opportunity of the field. The study shows that the implementation of computer science courses within high schools can have positive impacts on students if done properly (Dettori, Greenberg, McGee, & Reed, 2016; Kotiash, et al., 2022). Another study was done on student agency within computer science courses and found that culturally relevant and project-based teaching was more engaging for students. Students were interviewed about their experience with the curriculum that centered around the creation of yearlong project covering a problem in their community. Students reported that the ability to create and have their own agency made the course more engaging. Additionally, through this project-centric style of teaching, students felt more engaged with the computer science label and deepened their understanding of computer science in a non-traditional approach (Gale, et al., 2022).

Overall, computer science, in both AP CSP and as a general high school course, lacks studies as it pertains to the

experiences of students, especially on a more critical scale that accounts for the differences in experience level and diversity within the course (i.e., underrepresented groups). With this available literature in mind, a gap in the student experience with AP CSP as a course and experience is seen. The experiences of students in AP CSP as a course, rather than demographic statistics must be examined to add another layer of understanding to the depth at which the AP CSP course accomplishes its goal. Therefore, the following work will discuss the AP CSP course in relation to one of its third-party curriculums, Code.org, from the students' perspective as a case study of the course and its effectiveness in its goal as a prerequisite college course.

Background of the Students Writing This Perspective

This work is a perspective of two students, one female student and one male student. The female student went to high school in Okaloosa County, Florida, and graduated from High School in 2021. She was in a class of 20 students. 2021 was the first and only year the class was offered in this high school. It was also enrolled to be offered the following year. The male student went to a high school in Santa Rosa County, Florida, and was in a course section with 2 class periods of 30 to 35 students.

The female student, at the time of taking the AP CSP course, had taken three web design courses and up to Pre-Calculus Honors (in high school). She took AP CSP in her senior year of high school and is presently pursuing a computer science related degree at The University of West Florida. The male student, at the time of taking the AP CSP course, had no coding experience, and up to Algebra II Honors (in high school). The male student ony took AP CSP in his sophomore year of high school and took AP CSA the following year (junior year of high school). He is pursuing a computer engineering degree at The University of West Florida.

Understanding AP CSP: Course Breakdown

Prior to AP CSP, AP offered a course called AP Computer Science A (CSA). This course was offered in many schools but, much like many computer science courses, had a lack of women, Hispanic, and African American students compared to the white and Asian populations. To combat the lack of underrepresented groups, AP created and launched AP CSP in the 2016-2017 school year (Wyatt, Feng, & Ewing, 2020). Listed on College Board's website about the course, it explains how the course is intended to be "an introductory college-level course computing course that introduces students to the breadth of the field of computer science" (College Board, 2023). The course is taught as a yearlong high school course that has a final exam in the spring season. The course is made up of two parts: a multiple-choice exam worth 70% of the exam score and a create performance task worth 30% of the exam score. The multiple-choice segment is a 2-hour exam with various types of multiple-choice questions including multi-select question types. The create performance task is a coding project and corresponding questions about the project are completed prior to the exam in allotted class time (College Board, 2021). A summarization of the exam breakdown is shown in Table 1.

With the exam in mind, the course itself is broken down into five 'Big Ideas', each constituting different weights of the course evaluation. The five 'Big Ideas', their percentage of course content, and a synopsis of the content is

detailed in Table 2. The course is intended to cover a general ground for all aspects of computer science which is why discussions of privacy, security, and data are paired alongside code development and collaboration.

Table 1. AP CSP Exam Evaluation Breakdown

	Type of Questions	Number of Questions	Time to Complete	Percent of Exam Score
Section 1	Multiple-Choice	70 Questions	2 Hours	70%
Section 2	Create Performance Task	1 Question	At least 12 hours of class time	30%

Note. *AP Computer Science Principles Course Overview* (https://apcentral.collegeboard.org/media/pdf/ap-computer-science-principles-course-overview.pdf) by College Board 2021, College Board. Copyright 2021 by College Board.

Table 2. AP CSP Big Idea Curriculum

Big Idea	Course Content	Exam Weight (Multiple-Choice Section)
Big Idea 1: Creative Development	When developing computing innovations, developers can use a formal, iterative design process or a less rigid process of experimentation, and will encounter phases of investigating and reflecting, designing, prototyping, and testing. Collaboration is an important tool at any phase of development.	10%-13%
Big Idea 2: Data	Data are central to computing innovations because they communicate initial conditions to programs and represent new knowledge.	17%-22%
Big Idea 3: Algorithms and Programming	Programmers integrate algorithms and abstraction to create programs for creative purposes and to solve problems.	30%-35%
Big Idea 4: Computer Systems and Networks	Computer systems and networks are used to transfer data.	11%-15%
Big Idea 5: Impact of Computing	Computers and computing have revolutionized our lives. To use computing safely and responsibly, we need to be aware of privacy, security, and ethical issues.	21%-26%

Note. *AP Computer Science Principles Course Overview* (https://apcentral.collegeboard.org/media/pdf/ap-computer-science-principles-course-overview.pdf) by College Board 2021, College Board. Copyright 2021 by College Board. *AP Computer Science Principles Course and Exam Description* (https://apcentral.collegeboard.org/media/pdf/ap-computer-science-principles-course-and-exam-description.pdf) by College Board, 2020, College Board. Copyright 2020 by College Board.

Understanding Code.org

Although many organizations provide a teacher's guide to teaching AP CSP, in this work the Code.org course for AP CSP will be examined. The website is "recognized by the College Board as an endorsed provider of curriculum and professional development for AP Computer Science Principles" (Code.org, 2023). The website also provides

an extensive support network for teachers in the form of worksheets, lesson plans, and a community of fellow teachers to reach out to.

For the AP CSP course, the curriculum is divided into ten units. The contents of each unit are summarized in Table 4 with the corresponding mapping of the AP 'Big Idea' (from Table 2) indicating the course's coverage. Units with more abstract concepts take a day-by-day lecture approach while units with heavy emphasis on coding employing the Explore, Investigate, Practice, Make (EIPM) model. The EIPM model is an approach that allows teachers to create an approachable path to learning the major programming concepts by allowing for collaboration, creation, and teacher involvement throughout the process of learning each major concept (Code.org, 2023). Each letter is associated with a lesson's organization and demonstration of the information with the specifics of each day being summarized in Table 3. This approach allows teachers to ease students off their assistance and allows the student to become independent in understanding the topic by the end of the unit's subsection.

Table 3. Code.org EIPM (Explore, Investigate, Practice, Make) Education Model Breakdown

	Overview	Goal
Explore	Students explore the new concept through a teacher-led hands-on activity.	Students begin to develop a shared mental model and understand the main ideas of the new concept.
Investigate	Students investigate two or three sample programs that use the new concept.	Students become comfortable reading and modifying programs that use the new concept.
Practice	Students practice using the new concept through a scaffolded series of programming activities.	Students gain confidence in writing and debugging programs that use the new concept.
Make	Students make a target app for which they are given the screen elements but little to no starter code.	Students are able to independently decide when and how to use the new concept in the context of a larger project.

Note. Adapted from *A Short Introduction to EIPM* (https://docs.google.com/document/d/1ncil5b0yWAN4LCyOeXwYuNrNKEHtN4nmAd2o-_K5Psw/preview) by Code.org, 2020, Code.org. Copyright 2022 by Code.org.

Aside from the mini projects seen within most of the EIPM centered subsections, all but two of the units (Unit 6 and 8) have a comprehensive project that ties directly into the concepts of the unit in varying degrees. Additionally, at the end of each unit (excluding Unit 8) is a unit assessment to gage the student's understanding of the unit's material.

Table 4. Code.org AP CSP Course Curriculum Breakdown

Unit Number and Name	Unit Summary	Number of Lessons	AP Big Ideas Covered
Unit 1 – Digital	• Discusses the representation of information in the digital form from binary into complex information like numbers,	14	2 and 5

Information	text, images, and sound. • Discusses intellectual property and the existence of copyright in the digital world.		
Unit 2 – The Internet	• Discusses networks and how information is sent between computers using different protocols. • Discusses the impacts of the Internet in the modern age (both positive and negative).	9	4 and 5
Unit 3 – Intro to App Design	• Introduces important skills including debugging, collaboration, and user testing through the act of developing an app. • Teaches how to create user interfaces and write programs in the App Lab.	11	1 (seen in all units with collaboration and debugging)
Unit 4 – Variables, Conditionals, and Functions	• Using the EIPM model, variables, conditionals, and functions are introduced with increasing complexity.	15	1 and 3
Unit 5 – Lists, Loops, and Traversals	• Using the EIPM model, lists, loops, and traversals are introduced with increasing complexity. • Introduces the integration of large amounts of information in a program through the data library in the App Lab.	18	3
Unit 6 – Algorithms	• Discusses how algorithms are used to solve problems and how algorithm speed impacts the efficiency of the algorithm itself.	6	3 and 4
Unit 7 – Parameters, Return, and Libraries	• Using the EIPM model, parameters, return, and libraries are introduced with increasing complexity. • Introduces the creation of shareable libraries of code.	11	3
Unit 8 – Create PT Prep	• Prepares students for the AP Create Performance Task by teaching the aspects of the rubric through grading other assignments. • Guides students through the creation of their Performance Task using a detailed planning guide.	15	None (Create task is independent of Big Ideas but most concepts are used)
Unit 9 – Data	• Shows students how raw data can be analyzed through data visualizations to reveal patterns of useful information.	10	2 and 5
Unit 10 –	• Discusses how computing innovations have helped and	14	5

Cybersecurity and Global Impacts	harmed the modern world. • Discusses how data is a threat to privacy and how encryption is used to protect it.

Analysis

In order to assess the Code.org curriculum from a student's perspective, both strengths and shortcomings are discussed. First, we discuss the strengths of the AP CSP Course Experience and then discuss the strengths of AP CSP as taught through Code.org experience, and then we discuss the respective weaknesses from both these angles. Finally, the students's reflect on their personal experiences.

Strengths

In this section, we first discuss the AP CSP Course Experience and then present the strenghts of AP CSP being taught using the Code.org curriculum.

The AP CSP Course Experience

Unlike most computer science courses including AP Computer Science A, AP CSP discusses material relating to computer science outside of the coding exclusive sphere. This includes a discussion of the global impacts of technology in our everyday lives, the mechanics behind how computers and networks work including binary and network redundancy, and the niche topics of cybersecurity, privacy, and algorithms. The inclusion of these aspects of computer science allows students to realize the diversity of the field in terms of content.

Another good aspect of the AP CSP course is the way it handles code. Rather than restricting code to a single language, the course instead employs a pseudocode that resembles the basic structure of many coding languages. Additionally, this pseudocode has a reference sheet accessible during the exam for students to look back to ensure that they understand the syntax of the pseudo coding language. By allowing teachers to choose which language they teach their students, it allows more teachers to be able to teach the course as it isn't restricted to a single language like AP CSA is. This can include languages like JavaScript (which is what Code.org employs) which is a scripting language that many web design teachers have a strong understanding of. The course can also be taught by coding instructors who have experience in languages such as Python, C++, and Java. By teaching the concepts of coding like loops and variables, which are seen in all languages, it allows students to be exposed to structures they would learn and relearn in various languages.

Computer science as a field has many important skills outside of coding and logic. This can include topics such as collaboration, creativity, and problem solving through debugging. AP CSP's curriculum does a great job at emphasizing many of these skills through explicitly written aspects of the course that encourage the building of these skills through projects. Additionally, the exam attempts to encapsulate these skills through written questions that provide students with examples of being collaborative and being creative and assessing the approaches. These

skills are extremely important in any field, not just computer science. Being able to collaborate, be creative, and problem solve within the work field is an important skill for any student and it is great that this course encourages the growth of such skills. By teaching these skills the course can benefit students beyond the computer science track and make the course more beneficial to students outside of computer science.

As explained previously the course is assessed in two parts: a project and a written exam. Few AP courses currently employ this system, but this style of assessment is preferable as it eases stress during the exam since there is assessment outside of test day. The project is well assessed as it encapsulates the entire coding portion of the course. Additionally, the requirements of the project are quite simple, so it allows students to approach the project in whatever way they see fit that fulfills the guidelines.

Presentation of AP CSP By Code.Org: Strengths

The organization and presentation of the AP CSP curriculum by Code.org has many strengths that can be listed as: the usage of multimedia presentations; the emphasis on debugging and learning of code through code deconstruction; the focus on collaboration through student connections and pair programming; the involvement of the teacher through the EIPM model in student understanding; the reaffirmation of understanding through project-based assessments; and the assessments and resources found within the course itself.

Usage of Multimedia Presentations

Several studies have been done on the benefits of using multimedia or multiple forms of presentation of information (Kotiash, et al., 2022). A large strength seen throughout most units of the Code.org curriculum is the usage of multiple forms of presentation of information. This includes activities that are both hands-on and digital as well as information displayed in presentations, videos, animations, articles, and other digital medias. The more recent study by (Kotiash, et al., 2022)'s shows that this style of presentation is beneficial to students as it can "stimulate cognitive aspects of learning… [,] increase student motivation… [, and] develop students' fundamental approach to learning…[and] help form a more thorough understanding." This study discusses how the implementation of multimedia lectures can be difficult in the classroom due to financing and time allowance concerns. However, a course like AP CSP, a more non-traditional course is the perfect classroom to introduce this style of learning to students and into general school curriculum. Computer science is a course that requires technology involvement due to its nature so the use of multimedia lecture by classrooms is more likely to overcome possible limitations due to the nature of the course material.

Alongside the general presentation of information, the multimedia approach to lessons influences the recollection and retention of the vocabulary terms that are heavily emphasized by the curriculum. This can be seen with the integration of the terms within the previously mentioned presentations, videos, animations, etc. Using the multimedia presentation of vocabulary compared to traditional methods increased retention of the vocabulary terms alongside increasing enthusiasm about the content for both students and teachers. In another study, it was seen that students not only learned more of the vocabulary but also retained the vocabulary through the multimedia

method (Khiyabani, Ghonsooly, & Ghabanchi, 2014). This correlation supports a benefit of Code.org's lesson arrangement as the integration of computer science vocabulary into the multimedia presentations of information allows for longer retention of the terms compared to a vocabulary list that is traditionally studied.

Skill-Building: Collaboration to Debugging, Skills of Computer Science

As pertains to computer science specifically, there are important skills and learning techniques Code.org employs in their lessons. The most prominent concepts are debugging, pair programming, and collaboration. Firstly, with debugging, it is not only a standard required for the AP curriculum but a concept that is continually emphasized throughout the course. This can be seen in the 'Investigate' aspect of the EIPM-centric lessons that have one day of lesson focused on recognizing errors common in the programming concept being taught. The lessons, alongside project rubrics, emphasize the importance of debugging one's work and understanding the importance of incremental coding in order to check for said bugs. The importance of this debugging skill is emphasized by Jean Griffin's article discussing the ideas of learning through debugging. She emphasizes the research that supports that people learn code more if deconstructionist activities like reading, tracing, and debugging is incorporated (Griffin, 2016). This paper emphasizes that learning, by taking apart and understanding each aspect of the program, allows for rigorous engagement with the work and makes learning code more efficient. These ideas paired with the general importance of debugging in coding helps students realize the importance of debugging alongside learning to code more efficiently. The 'Investigate' lesson of the EIPM model is built with the purpose of ensuring students can engage with the code through reading, modifying, and debugging the code before they ever try and code it themselves.

Another important set of skills emphasized in the Code.org course is pair programming and collaboration. AP's curriculum has an entire 'Big Idea' focused heavily on collaboration. Collaboration is an important technique in not only computer science fields but many other fields of study so emphasizing these skills is great for all students, not just those who are computer science bound. Code.org describes pair programming as a technique where two students share a device and switch between being the active coder (driver) and the spectator (navigator). These roles are switched often to allow students to critically think and to increase student engagement through active work (Code.org). A study on the effectiveness of pair programming showed positive correlations between course competition, final performance, program quality, and confidence and enjoyment. In other words, it was found that pair programming allowed students to create better programs, pass the course with higher marks, and gain greater confidence and enjoyment out of the course itself (McDowell, Werner, Bullock, & Fernald, 2003). Pair programming is not the only form of collaboration found within the Code.org lesson plan. General collaboration is found in class discussions and projects that involve group presentations and collaborations. Collaboration is also seen within the course during the EIPM-centric sections as students are encouraged to assist one another if confusion arises.

Code.org's EIPM model allows for teachers to slowly step away from students as they gain an understanding of the material. One major benefit of this is the teacher's ability to support students beyond simple lecture. As found in a study on teaching approaches, the immediate guidance and feedback found in active learning approaches

(most like the EIPM model) was a major benefit for students. Additionally, student feedback stated that hands-on learning increased attention and understanding (Kay, MacDonald, & DiGiuseppe, 2019). The EIPM model allows for students to receive many of the same benefits as feedback from the teacher on current lessons is enabled through the 'Investigate' and 'Practice' portion of the model as students are working alone or in pairs and can call on the teacher for guidance as they practice the concepts directly. Feedback in general is an important part of improvement and no better is that seen then in the course's projects.

Projects, Assessments, and Handouts: Skill Assessment Towards Success

In terms of projects, nearly all units contain at least one comprehensive project with many units also containing one-day projects in the form of the 'Make' tasks in the EIPM cycle. These projects are completed using a clearly outlined rubric and allow teachers to give students ample feedback. The projects are designed to be all encompassing with regard to the ideas of the unit, whether it be programming concepts or more abstract ideas in the global impacts of technology. A literature review on project-based learning discusses several studies of students learning through project-based learning and found that the projects improved motivation and self-image. From the same literature review, one longitudinal study described, students developed a conceptual understanding rooted in creative and deeper thinking (Kokotsaki, Menzies, & Wiggins, 2016). Overall, the paper describes the benefits of project-based learning as being beneficial but dependent upon the teacher's engagement and guidance. The curriculum's structure and clear outlines of deadlines and expectations alongside teacher involvement allows for students to benefit from project-based learning through the cumulative final projects of each unit.

Many of the coding-focused units use a written question guide to assess the project alongside the code itself. These questions are lifted directly from the AP Create Task project that students submit at the end of the year as part of their examination. By exposing students to these questions early in the course, it allows for students to receive feedback from the teacher on what they may have not answered properly. Through this feedback, students can become familiar with the questions and ensure that their Create Task responses account for their previous errors. Lastly, an aspect independent from the course organization is the materials. Although all projects have rubrics and planning guides and most assignments have an accompanying paper assignment, the most beneficial resource in reflection are those found in Unit 8. Unit 8 is the unit that focuses on students learning the rubric and expectations of the final Create Task before allowing students to create their project and answer the accompanying questions. The unit provides an extensive guide to ensuring that students include all required parts of the project in their code alongside ensuring they understand how to answer the questions properly. This guide, taken with the previous exposure to the project questions through previous projects allows for students to confidently complete the Create Task.

Shortcomings

In this section we present the shortcomings of the AP CSP course experience and then we discuss the shortcomings of the course taught through Code.org.

The AP Course Experience

AP CSP is advertised as a computer science principles course that will cover all the basic aspects of computer science. However, the course does not place much emphasis on more contemporary topics like Cybersecurity or Data Science. Students taking this course do not get a feel for what a Cybersecurity or Data Science jobs will entail. In the course overview and curriculum of AP CSP, the College Board advertises the course as being interchangeable with AP Computer Science A. But, AP CSA teaches students how to code using Java and focuses solely on teaching the ins and outs of this language in an introductory college level coding class approach. AP CSP on the other hand is intended to give an overview of the computer science field including the many nuances of the non-coding aspects of the field. With these understandings, the only viable arrangement would be for AP CSP to be taken prior to AP CSA.

Another issue of the course is the college credit aspect of this College Board course. Many students take AP courses in order to receive college credit for courses that they would take in college in order to minimize the amount of money and time they need to spend. AP CSP was created with this intent in mind. However, in practice this is not the case. For the computer science major at many public universities in the state of Florida for example, the course credit received for passing the AP CSP course is not a requirement for the major. The course is considered too elementary for course credit at the college level. So, although this isn't to the benefit of many computer science majors in terms of credits, it is great for most other majors. It provides a college elective credit at the high school level for a unique topic that directly impacts their own life. Additionally, although credit may not be awarded, it can be a great indicator of interest in the material for computer science majors and allow them to reflect on if they have a genuine interest in the field.

Presentation of AP CSP By Code.Org: Shortcomings

Code.org's curriculum can be divided into two types of units: conceptual and coding. Conceptual units include units 1, 2, 6, 9, and 10 which focus on the ideas of computer science including algorithms, networks, representations of information, and the global impacts of technology. Coding units include units 3, 4, 5, and 7 which use a more creative and hands-on approach as students code and create apps using the App Developer. The conceptual units are a little slower-paced with little information, while the coding units offered too much information with too little time. An issue seen in both conceptual and coding units is the presentation of vocabulary. Lessons which don't employ the integration of the vocabulary would benefit from the integration of the vocabulary into the lesson itself, allowing students to connect terms with concepts as they learn about them.

Projects and Assessments: Disconnected from Curriculum

Another staple of the Code.org curriculum is its projects. A concern seen in relation to projects is pacing. Two units take a unique approach of having the project being worked on while lessons are occurring. The units are set up so that a day is spent on the project, then two days of lecture, then two days spent on the project, etc. The issue with this approach is that constant shift between working and learning that makes it difficult for students to fully

commit to either a working or learning mindset.

For the exams at the end of each unit, an integration of more AP style questions may help students feel more familiar with the multiple-choice question style of AP. Code.org does a great job of doing this with the projects and the exposure to AP Create Task style questions before the project is introduced. Code.org should take a similar approach and expose students to more AP questions to ensure a familiarity with the question style before the final exam. Overall, the Code.org curriculum does a great job at covering all material required to pass the AP CSP exam. It could benefit from the lessons were spaced more evenly so equal amounts of content is learned each day.

Personal Experience

In this section we reflect on the personal experience in terms of resources are well as presentation of the material.

Resources and Presentations

The Good

Code.org has great resources available to students, the best being the ones for Unit 8. Unit 8 relates to the AP Create Task and Code.org does a phenomenal job structuring the process of understanding the requirements and creating one's project. The unit provides handouts that allow students to learn the rubric through grading other projects to ensure that, before students even begin to think of their project, they understand what is expected of them by the rubric. I personally love this style of teaching the rubric prior to attempting to create the project itself. (Female Student)

Although I loved the resources that Code.org provides, when taking the class, I never saw any of the resources for students. My teacher failed to use the website to its fullest potential. Code.org prides itself on being accessible for teachers with worksheets and plans to assist them in teaching the course. Teachers who don't fully use these resources are hurting both themselves and students. In my experience, my class was not given access to any of the resources, including the slideshow presentations that were intended to teach students through teacher lecture. An easy way to combat this issue is to ensure that teachers are aware of the importance of the handouts and presentations in conveying information to students. (Female Student)

The Bad

One of the major issues was with the presentation of information in some sections in Code.org. There are a few topics that are introduced without covering an understanding of the material/topic. An example can be cited for Unit 1, in relation to learning binary conversions. Binary is not a difficult concept to learn if the methods for translating between decimal and binary are taught. However, the website introduces a short cut activity called Flippy-Do which makes translating between decimal and binary easy. But, using this short cut students do not learn how to do the translation when it comes to the exam. Covering the actual methodology rather than a shorthand would have been more beneficial. (Female and male student)

The EIPM Model: A System Failed in Application

The EIPM model is a phenomenal tool. This model provides a variety of presentation of material and changes the type of engagement, making the cyclical nature more enjoyable for the student. This helps in catering to the different learning styles of students, and studies have shown that catering to different learning styles helps students. The EIPM model provides a great structure for teachers to ensure that they present the information in an organized manner. Having the lecture aspect still incorporated while emphasizing student learning through reading and eventually writing of the code is a great way to learn the concepts. Relating to the coding aspects of the course, the overall organization of the material felt more natural, with the progression of skills and with the increasing levels of difficulty and complexity seeming to match the students' increasing skills. (Female and Male student)
However, our teacher did not employ the EIPM model, hence all the inexperienced students were left to drown in confusion. The slideshow presentations provided by Code.org were also not used, which led to unstructured learning and confused students. Code.org provides ample guides for students to use to plan out projects and work through assignments, but students received no guidance from the teacher in terms of using these resources. And, without these resources, the entire course felt disorganized and unproductive. I felt that it would be beneficial if Code.org had a section providing guidance to teachers on how to teach the course. (Female student)

Teacher Involvement: A Curriculum Made for the Engaged

Another great aspect of any course is teacher involvement. The course's introduction of AP Create Task style questions on earlier projects allows for students to receive feedback and grow in their ability to answer these types of questions for the final project. However, many teachers provide no feedback and fail to use the questions in evaluating projects which appeared to make the Create Task style of questions an uncharted territory. (Female student)The course emphasizes collaboration through activities like the pair programming approach, but this approach was not used our class. Since the approach to use pair programming was not used consciously, pairs were often incompatible, for example, inexperienced students were paired together and not monitored. Hence, inexperienced groups fell behind, while pairs with a more experienced coder were carried by one student with the others failing to gain any insights from the experience. (Female student)

Website Design and Skills Past AP CSP

Code.org has a great design in its website that makes it approachable for students and teachers. The block coding UI of Code.org makes coding friendly for new students. Allowing students to fit the pieces of code together allows them to focus on the flow of code rather than the syntax of the code, allowing for a brief understanding of coding in relation to the AP CSP exam requirements. Additionally, in Code.org's website design, at the end of each unit, students can provide feedback to their teacher on what they do and don't understand so that teachers can easily address misunderstanding within the class through constant feedback from students. (Male student)

Code.org does a great job at teaching future skills, both within the spheres of coding and outside of it. The curriculum emphasizes collaboration and communication which is a skill that students need to use in all fields,

computer science or otherwise. Additionally, the curriculum teaches other skills outside of coding including requiring students to complete a one-pager, a skill I had not been exposed to in high school prior to using Code.org and is a skill used in college and the workplace. (Female student).

Another future orientated part of the course is the connection between AP CSP and AP CSA through the skills taught in the course. How Code.org teaches AP CSP serves as a great segue into AP CSA as the website gives students a foundation in programming that is easy to build off. Since AP CSP focuses on the concepts of coding, students who then pursue AP CSA will have a basic understanding of loops, arrays, and functions that allow for a focus on understanding the syntax of coding while learning Java in AP CSA. Code.org also allows for students to code line-by-line as opposed to block coding, which allows students to be exposed to the syntax if they feel confident enough to try it for themselves. (Males student)

Student Conclusions

Overall, our experience with the course would have been a lot more positive if our teacher utilized the many support systems provided by Code.org. With little or no feedback, class direction, or worksheets, the course failed to teach anyone in my class, experienced or inexperienced. I disliked the course while taking it and only upon looking at the teacher resources during this research did I realize it was not a limitation of the material itself, but a limitation of instruction. Teachers need to be made aware of the requirements of the course. To summarize, Code.org is filled with wonderful resources and provides a great support network for educators to create an atmosphere of creativity and understanding for their students if educators utilize the curriculum and website to its fullest potential. (Female Student)

Conclusions

In the general section, beneficial approaches to teaching are found including effective multimedia presentations with effective vocabulary integration; relevant skill building including debugging, pair programming, and collaboration; heavy emphasis on teacher involvement including guidance and feedback during active learning segments of the course like EIPM model lessons. Additionally, Code.org provides well assessed assignments and projects that guide students to a deeper understanding of the material alongside extensive guides and worksheets to complete assignments and thorough rubrics to assess those assignments and projects.

The course still has weak areas which fall mainly on pacing issues. Many units feel empty with little information while other units are full of information but feel rushed. This dichotomy is seen heavily between the conceptual units and the coding units where the former feel slow while the later are often rushed. This issue of pacing could easily by aided by the reevaluation of the number of class days spent on lessons and units with the condensing and expanding of units in relation to the amount of time actually needed for students to gain an understanding of the material. Another issue is minor issues including a need to integrate more AP style multiple choice questions into the end of unit exams to allow for students to experience more exposure to the question types (although this lack of questions is understandable due to the recency of the course and the availability of released questions).

The other issues are easy to combat as the structures to fix the issues already exist. This includes the failure of some lessons to integrate the vocabulary into the presentation or the pacing of projects within lessons.

Overall, from a general perspective, Code.org does a great job at providing extensive resources to teachers and students to ensure success in relation to the AP CSP exam and the course material as a whole. Many issues with the curriculum can easily be aided by the existing evidence of success in the many effective aspects of the course. In relation to a more personal assessment of the course, many of the effective materials and resources fail if not lead by involved teachers. Many personal issues stated by the two students pertained to the failure of engagement by the teachers themselves in executing the provided materials. Praised aspects like the extensive Unit 8 guidelines and the EIPM model fail if the teacher is not involved as an instructor and critique for students to improve their work throughout the course. Even with the many issues in relation to the failure of teachers, taking the course through Code.org overall is beneficial as it provides students with a foundation for computer science with a heavily structured and providing platform that teaches hard skills like coding and debugging but also soft skills like communication and collaboration.

Acknowledgements

This work has been supported by NSF grant no. 2122393

References

Code.org. (2022). *Florida 2022 State of CS Report | CS Advocacy*. Retrieved from CS Advocacy: https://advocacy.code.org/stateofcs

Code.org. (2023). *CS Principles | Code.org*. Retrieved from Code.org: https://code.org/educate/csp

Code.org, CSTA, & ECEP Alliance. (2022). *2022 State of Computer Science Education: Understanding Our National Imperative.* Retrieved from https://advocacy.code.org/stateofcs

College Board. (2021). *AP Computer Science Principles Course Overview.* Retrieved from AP Central: https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course

College Board. (2023). *AP Computer Science Principles Course - AP Central | College Board*. Retrieved from AP Central | College Board: https://apcentral.collegeboard.org/courses/ap-computer-science-principles/course

Dettori, L., Greenberg, R. I., McGee, S., & Reed, D. (2016, March). The Impacts of Meaningful High School Computer Science Experiences in the Chicago Public Schools. *Computing in Science & Engineering (Special Issue: Best of RESPECT 2015)*, pp. 10-17. Retrieved from https://ecommons.luc.edu/cs_facpubs/175/

Gale, J., Alemdar, M., Boice, K., Hernández, D., Newton, S., Edwards, D., & Usselman, M. (2022, June 6). Student Agency in a High School Computer Science Course. *Journal of STEM Education Research*, pp. 270-301. doi:10.1007/s41979-022-00071-9

Khiyabani, H., Ghonsooly, B., & Ghabanchi, Z. (2014, January 6). Using Multimedia in Teaching Vocaulary in High School Classes. *Journal of Advances in English Language Teaching, 2*(1), pp. 1-13.

Killen, H., Weintrop, D., & Garvin, M. (2019, Feburary). AP Computer Science Principles' Impact on the Landscape of. *SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pp. 1060-1066. doi:10.1145/3287324.3287356

Kotiash, I., Shevchuk, I., Porysonok, M., Matviienko, I., Popov, M., Terekhov, V., & Kuchai, O. (2022, June). Possibilities of Using Multimedia Technologies in Education. *International Journal of Computer Science and Network Security*, pp. 727-732. doi:10.22937/IJCSNS.2022.22.6.91

Meur, A. L. (2022). Approaches to Broadening Participation with AP Computer Science Principles. *Scholarly Horizons: University of Minnesota, Morris Undergraduate Journal*. Retrieved from https://digitalcommons.morris.umn.edu/horizons/vol9/iss2/4?utm_source=digitalcommons.morris.umn.edu%2Fhorizons%2Fvol9%2Fiss2%2F4&utm_medium=PDF&utm_campaign=PDFCoverPages

Sax, L. J., Newhouse, K. N., Goode, J., Skorodinsky, M., Nakajima, T. M., & Sendowski, M. (2020, Feburary 26). Does AP CS Principles Broaden Participation in Computing? *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pp. 542-548. doi:10.1145/3328778.3366826

U.S. Bureau of Labor Statistics. (2022, September 8). *Computer and Information Technology Occupations*. Retrieved from U.S. Bureau of Labor Statistics: https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Wyatt, J., Feng, J., & Ewing, M. (2020). *AP®Computer Science Principles and the Computer Science Pipelines.* CollegeBoard.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016, December 05). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, pp. 235-254. doi:10.1080/08993408.2016.1257418

Author Information

Sarah Cameron
https://orcid.org/0000-0001-8167-8179
University of West Florida
USA

Tony Pham
https://orcid.org/0000-0001-5654-163X
University of West Florida
USA

Sikha Bagui
https://orcid.org/0000-0002-1886-4582
University of West Florida
USA
Contact e-mail: *bagui@uwf.edu*