CSAwesome: AP CSA Curriculum and Professional Development

Barbara Ericson
University of Michigan
Ann Arbor, MI
barbarer@umich.edu

Beryl Hoffman
Elms College
Chicopee, MA
hoffmanb@elms.edu

Jennifer Rosato
College of St. Scholastica
Duluth, MN
jrosato@css.edu

ABSTRACT

CSAwesome is a new approved curriculum and professional development (PD) provider for the Advanced Placement (AP) Computer Science (CS) A high school course. AP courses are taken by secondary (typically ages 14-19) students for college placement and/or credit. CSAwesome's free curriculum and teacher resources were developed in 2019 by adapting the CSA Java Review ebook on the open-source Runestone platform. The goals of CSAwesome are to broaden participation in the AP CSA course and to support new-to-CS students and teachers as they transition from the AP Computer Science Principles (CSP) course to the AP CSA course by using inclusive teaching practices and curriculum design. The AP CSP course is equivalent to a first course for non-majors at the college level, while the AP CSA course is equivalent to a first course for majors. Currently, AP CSA attracts a much less diverse student body than AP CSP. This new curriculum supports student engagement and scaffolded learning through an interactive ebook with embedded executable and modifiable code (Active Code), a variety of practice types with immediate feedback, and adaptable mixed-up code (Parsons) problems. Collaborative learning is encouraged through pair programming and groupwork. Our pilot Professional Development (PD) incorporates inclusive teaching strategies and active recruitment with the goal of broadening participation in CSA. This paper presents the design of the CSAwesome curriculum and teacher professional development and initial results from the curriculum use and pilot PD during the first year of CSAwesome.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

AP CSA,CS1,curriculum,professional development, high school, broadening participation inclusive teaching

ACM Reference Format:
Barbara Ericson, Beryl Hoffman, and Jennifer Rosato. 2020. CSAwesome: AP CSA Curriculum and Professional Development. In *Workshop in Primary and Secondary Computing Education (WiPSCE '20), October 28–30, 2020, Virtual Event, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3421590.3421593

WiPSCE '20, October 28–30, 2020, Virtual Event, Germany

ACM ISBN 978-1-4503-8759-0/20/10…$15.00
https://doi.org/10.1145/3421590.3421593

1 AP CSA

The College Board's AP Computer Science A (CSA) course is offered in secondary schools (ages 14 - 19) and is equivalent to an introductory college-level course (CS1) for CS majors [5]. Students can take this course and AP exam for college credit and/or placement. It covers Java programming fundamentals, object-oriented programming, arrays, two-dimensional arrays, lists, searching, sorting, and recursion. Over 64 thousand secondary school students took the AP CSA exam in 2019. However, only 24.5% of the AP CSA national exam-takers were female, 3.9% were African-American, and 12% were Hispanic [13], see Table 1. The success rates of these traditionally underrepresented groups in CS are also lower than the national averages, see Table 2.

Table 1: Demographics of AP CSP and CSA 2019 exam takers

AP Exam	Female	Black	Hispanic
AP CSP	32.7%	7.0%	19.7%
AP CSA	24.5%	3.9%	12.0%

Table 2: Demographics of AP CSP and CSA 2019 exam pass rates (score of 3+)

AP Exam	All	Female	Black	Hispanic
AP CSP	71.7%	69.4%	41.9%	54.4%
AP CSA	69.2%	66.8%	37.1%	46.9%

In contrast, the newer AP Computer Science Principles (CSP) exam [6], which is equivalent to a first course for non-majors at the college level, has higher representation and success for female and underrepresented minority students (32.7% female, 7.0% African-American, and 19.7% Hispanic), see Table 1 and 2, although there is still much room for improvement. The AP CSP course provides a broad overview of CS including programming fundamentals (variables, loops, conditionals, and functions), often in block-based coding environments. It also encourages creative, collaborative, and engaging projects, and support for equity-based pedagogy.

The number of students taking the AP CSP exam more than doubled from 2017 (43,780) to 2019 (94,360) while the AP CSA exam has seen steady growth of about 4,000 exams a year from 2017 (56,088) to 2019 (64,197) [4], see Table 3. Many students are being introduced to CS in a CSP course that uses a blocks-based programming environment and then want to continue to text-based programming in AP CSA. However, to meet this demand more teachers need to be prepared to teach AP CSA, and we need curriculum that engages students and ensures their success.

Table 3: Number of Exam Takers 2017 to 2019

Exam	2017	2018	2019
AP CSP	43,780	70,864	94,360
AP CSA	56,088	60,040	64,197

According to [32], first year college students who took only CSP in high school were more diverse than those who took only CSA, but they had less computing confidence and less interest in computing majors and careers. This suggests that AP CSP followed by an engaging and inclusive AP CSA curriculum in high school may better serve to broaden the pipeline into CS majors and careers.

2 CSAWESOME CURRICULUM DESIGN

CSAwesome [10] is a new free curriculum for AP CSA, developed in 2019 by adapting the CSA Java Review ebook on the open-source Runestone platform [18]. The curriculum adheres to the College Board's 2019 AP CSA unit layout and learning objectives as shown in table 4. CSAwesome has been approved as an official College Board AP CSA curriculum and PD provider for 2020-21. The CSAwesome curriculum has been successful in its first year with over 10,000 registered users and over 1,500 teachers in the Teaching CSAwesome professional learning community and Google group which allows access to teacher resources.

Table 4: AP CSA Units

Unit	Name	Topics
1	Primitive Types	variables, expressions, operators
2	Using Objects	constructors, methods, strings
3	Conditionals	Boolean, control flow, comparing objects
4	Iteration	while loops, for loops, nested for loops
5	Writing Classes	constructors, accessors, mutators
6	Arrays	creating, traversing, enhanced for loop
7	ArrayList	traversing, searching, sorting
8	2D Arrays	creating, traversing
9	Inheritance	constructors, overriding, polymorphism
10	Recursion	tracing, searching, sorting

The goals of CSAwesome are to broaden participation in the AP CSA course by supporting students and teachers as they transition from AP CSP to AP CSA with inclusive curriculum design and teaching practices. The curriculum is based on an online interactive ebook with embedded coding problems with frequent opportunities for creativity and collaboration.

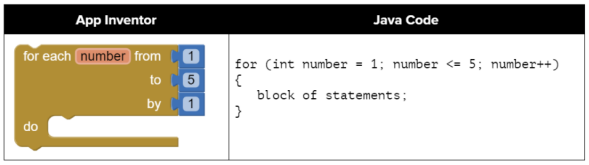
2.1 Ebook Design

The ebook design was informed by the theories of constructivism, cognitive load, desirable difficulties, and social constructionism.

2.1.1 Constructivism. Constructivism posits that people learn best when they are actively constructing their own knowledge and connecting new knowledge to what they already know [39]. The ebook allows learners to actively construct knowledge through interactive practice problems with immediate feedback: multiple-choice questions, fill-in-the-blank questions, matching, and clickable code.

The ebook connects to what people already know through analogies such as arrays are like rows of lockers and by showing how to translate concepts from blocks-based languages to Java as shown in Figure 1. This mapping helps teachers connect the previous context (CSP) and the new context (CSA). This bridging process has been shown to be effective in mediating the transfer of knowledge between two areas [11].

Figure 1: Connecting blocks-based and Java for loops



2.1.2 Cognitive Load. Cognitive load argues that humans have limited working memory and that overloading working memory during instruction impedes learning [34]. Programming courses often require students to write code from scratch, however this can quickly overwhelm working memory [33, 38]. One of the recommended approaches to reducing cognitive load is to use completion tasks rather than whole tasks [37]. One way the ebook does this is by providing mixed-up code problems also known as Parsons problems [29]. In a Parsons problem, the user puts blocks of mixed-up code in order to solve a problem as shown in Figure 2. This type of problem may be particularly helpful in the transition from blocks-based to text-based programming.

Parsons problems can also include incorrect blocks known as distractors and may also require the learner to correctly indent the code blocks as shown in Figure 2. While Java does not require indentation, it is good programming practice and makes code more readable.

Adaptive learning modifies the difficulty of problems to scaffold a learner and attempts to keep the learner in the zone of proximal development (ZPD) [2], which is just beyond what the learner can accomplish unassisted, but can accomplish with help. In the CSAwesome course, there are two types of adaptations for Parsons problems: intra-problem and inter-problem to try to keep the learner in the ZPD. In intra-problem adaptation if the learner is struggling to solve the current problem it can dynamically be made easier by removing distractors, providing indentation, or combining blocks. In inter-problem adaptation the difficulty of the next problem is modified based on the learner's performance on the last problem.

Figure 2: An example Parsons problem with distractor blocks on the left and the correct solution on the right. Note that the blocks are both in the correct order and indented correctly.

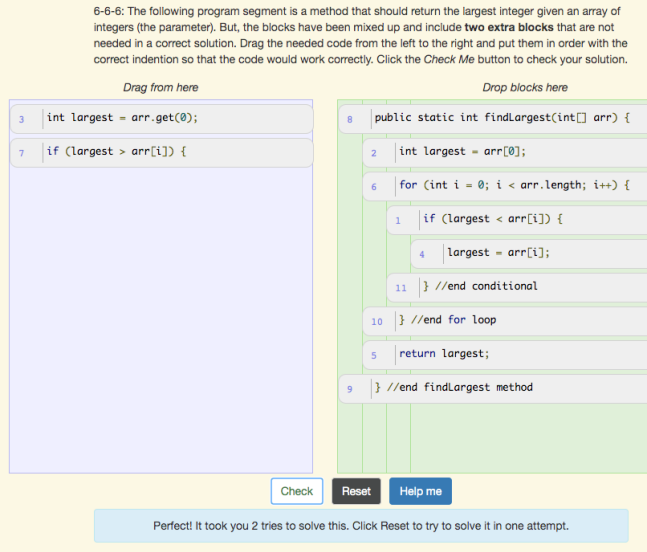
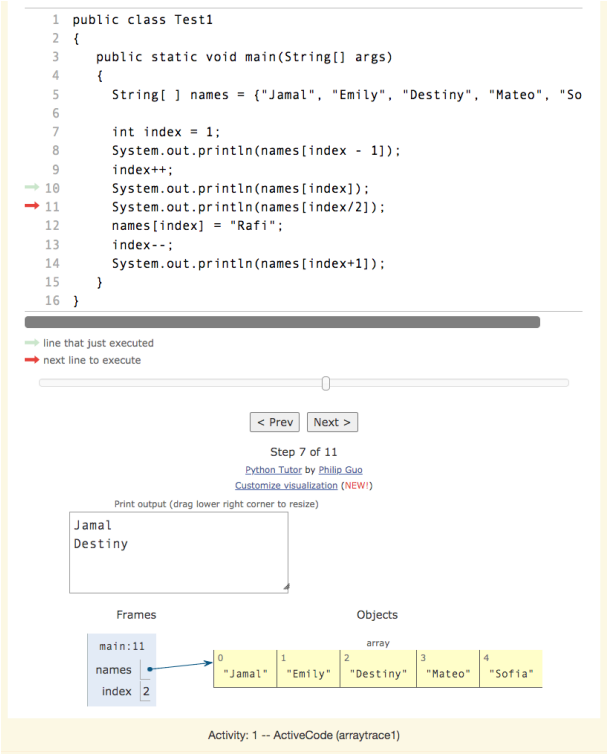


Figure 3: The Java code visualizer/stepper



There is evidence that most learners enjoy solving Parsons problems [29], many attempt to solve them [16], solving Parsons problems is more efficient for learning than writing the equivalent code or than fixing code with errors [15, 17], and that learners are twice as likely to correctly solve adaptive versus non-adaptive Parsons problems [14].

Another technique to reduce cognitive load is using worked examples plus interleaved practice problems [1, 35, 36]. A worked example is an expert's solution to a problem. The ebook contains worked examples in the form of executable and modifiable Java code followed by practice problems. Learners can also use a code visualizer/stepper based on the Python Tutor [22] to step through code examples as shown in Figure 3 [22]. All active code problems in the ebook now have a Code Lens button which shows the Java visualizer for that code to allow the student to step through the code.

2.1.3 Desirable Difficulties. Desirable difficulties are those that help learners store and recall information in multiple contexts [3]. One key idea of this work is that improving the learner's performance while learning can actually decrease long-term learning, and conversely techniques that reduce the learner's performance while learning can actually lead to long-term retention and better recall. Some techniques that result in desirable difficulties are spaced practice over time rather than massed practice [9, 30], interleaving different subjects, and using tests for learning rather than just for assessment. The open-source ebook platform, Runestone, has a practice tool that supports spaced practice [40]. Instructors can require that students correctly answer some number of questions a day over some number of days. The questions are selected from previously covered chapters based on a algorithm that predicts when the learner is likely to forget a topic. Learners can postpone a question if they do not know the correct answer. In a study with undergraduate students, each hour of use of the tool was associated with a 1.04% increase in the final exam grade [40].

2.1.4 Social Constructionism. Constructionism is similar to constructivism in that they both agree that learners need to actively construct their own knowledge. Social Constructionism adds that this works best when working in social groups and when creating something that you care about and can share with others [28].

To encourage social constructionism the ebook supports pair programming, collaborative work, and creative open-ended projects which have been shown to improve student retention, confidence, and success in CS, especially among underrepresented students [24, 25]. For example, students are asked to design a turtle house working in pairs using Turtle graphics with loops and a text adventure game when working with if statements. The ebook uses Turtle graphics to introduce objects and classes and to allow learners to use their body knowledge to understand programs [28].

2.2 Teacher Resources

The CSAwesome curriculum also supports new to CSA teachers with comprehensive teacher resources. Each lesson in the book has a corresponding lesson plan and unplugged activities and worksheets developed by experienced CSA teachers. The lesson plans include the CSA learning objectives and a detailed guide with timings for the learning activities with a hook, direct instruction, guided practice, collaborative work, and reflection sections. They also contain

inclusive teaching strategies and support for differentiation and teaching tips to support the transition from CSP to CSA.

The curriculum also provides support for the AP CSA College Board Labs and AP free response questions (FRQs). The College Board provides labs which are not required, but can be used to meet the minimum requirement of 20 hours of hands-on lab experiences. Each lab provides programming practice in a context, such as the Picture Lab, which covers two-dimensional arrays by having students write code to modify pictures. The AP CSA exam has 40 multiple-choice questions and four free response questions. In a free response question (FRQ) the student is given an English description of a problem and must write code to solve that problem. FRQs are challenging for many new-to-Java teachers and students. The curriculum scaffolds FRQs though practice problems with immediate feedback to help the student translate from English to algorithm to code.

Table 5: Online Extended PD Schedule

Week	Java Content	Inclusive Pedagogy
1	Unit 1 Primitive Types	Growth Mindset, Debugging strategies
2	Unit 2 Using Objects	Pair Programming, unplugged objects activities
3	Unit 3 Conditionals	POGIL groupwork, TLO teaching practice
4	Unit 4 Iteration	Creating welcoming environments, FRQ strategies, TLO
5	Unit 5 Writing Classes	Unconscious bias and microaggressions activities, OOP activities, TLO
6	Unit 6 Arrays	Active recruiting strategies, unplugged activities, TLO

3 CSAWESOME PROFESSIONAL DEVELOPMENT

There is a shortage of teachers who are trained to teach CSA, and a demand among new-to-CS CSP teachers for additional professional development to develop the skills necessary to teach AP CSA. Currently, many of the CSP teachers are new to CS and come from other areas such as Math or Business; their only CS experience comes from participating in a CSP PD which is usually using a block-based programming environment (MIT App Inventor, Snap!, Scratch, etc.). They require further PD to transition to teaching AP CSA and more complex text-based programming using Java, as well as continued support to use equity-based pedagogy in AP CSA.

CSAwesome held an online pilot PD for 24 CSP teachers during summer 2019 and a hybrid pilot PD for 30 CSP teachers transitioning to CSA at the Pathfinders Winter Institute in February 2020. Demand for CSAwesome PD has been substantial this past year, with over 100 teachers submitting an interest form and 78 teachers applying for 54 spots in our first two PD opportunities (summer 2019 and winter 2020), and over 100 teachers who have registered for PD this summer (2020).

The CSAwesome PD is based on effective professional development and learning theory by [12] which identifies seven key areas for effective professional development: content-focused, incorporates active learning, supports collaboration, uses models of effective practice, provides coaching and expert support, offers feedback and reflection, and sustained duration. We use a master teacher model where the PD facilitators are experienced CSA teachers who model teaching, pedagogy, and work to build a professional learning community. New-to-Java teachers are encouraged to take a 65 hour summer PD and are supported throughout the academic year.

The CSAwesome PD is offered in two differentiated formats: extended PD (65 hours) for new-to-Java teachers and immersion PD (45 hours) for teachers with more experience in text-based coding. Teachers who are new to Java in the extended PD are supported by PD facilitators in an initial 20 hours of online PD to build up their Java coding and debugging skills covering the first units of the curriculum. This part is optional for teachers who have some Java coding experience. Then, the immersion PD teachers join the extended PD teachers in a 45 hour PD covering the CSAwesome curriculum content, AP CSA materials, and inclusive teaching and pedagogy. The PD can be in-person or online with a blend of synchronous and asynchronous instruction. The PD schedule in Table 5 shows the 65 hour extended PD covering Units 1-6 in 6 weeks with approximately 10 hours of work each week followed by academic year support that will cover Units 7-10.

The PD facilitators, who are experienced CSA teachers trained by the CSAwesome team, model teaching lessons, lead group practice, support teachers through individual practice, and discuss teaching pedagogy, equity and inclusive teaching. PD facilitators also support the participants during the academic year with monthly meetings to form a community of practice and provide just in time support while the participants are teaching the course for the first time. PD participants also practice teaching during the Teacher-Learner-Observer (TLO) sessions of the PD, which provide teachers with both an opportunity to gain confidence in a supportive community as well as reflect on their practice [7, 21].

The CSAwesome PD is innovative in incorporating inclusive teaching practices and pedagogical content knowledge (PCK) in teaching CSA with the goal to broaden participation in CS. In CSAwesome PD, teachers discuss inclusive teaching pedagogy and practice mitigating stereotype threat, unconscious bias, and microaggressions in order to provide equitable access to CS learning and increase the sense of belonging for underrepresented students [23, 26, 27]. Including equity issues through discussions and culturally responsive computing lessons helps teachers recognize their belief systems and how they impact students [21]. Teachers also create student recruitment action plans and work on inclusive teaching strategies based on their interests and school demographics.

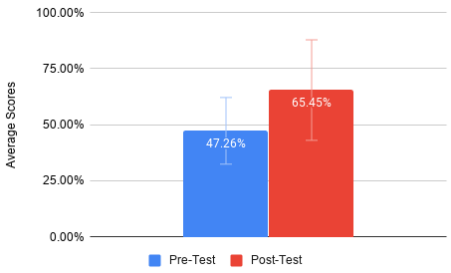
The CSAwesome curriculum and pedagogy is structured around inclusive teaching practices and PCK that support student engagement [26] such as pair programming, collaborative group work, process-oriented guided inquiry learning (POGIL), growth mindset, active recruitment, and creating a welcoming physical environment. The Tapestry Workshops [8] and partners emphasized using pair programming, projects that connect to student interests and

other subject areas, as well as interventions from educational psychology such as growth mindset, normalizing struggle, providing feedback, and self-affirmations. These interventions are helpful for all students, but have been shown to narrow achievement gaps for students underrepresented in STEM who are impacted negatively by stereotypes.

4 CSAWESOME PD RESULTS

Initial results from the Summer 2019 and Winter Pathfinders 2020 pilot PD cohorts indicate that teachers' CS content knowledge and ability to teach AP CSA Java were positively impacted by the PD. Most (n=31) of the 54 teachers in the two PD cohorts took the pre/post test that was developed and used in the Runestone Java Review ebook for the past six years. This is a difficult test of 20 questions modeled on the AP CSA exam where a passing score is about 50%. For the PD participants who were mostly new to Java, the average Java post-test score taken after the PD was 65%, 18% higher than the average pretest score taken before the PD which was 47%, see Figure 4. The distribution of the scores varied widely and did not correlate with self-identification as novice/expert or years of teaching CS. However, all scores exhibited a significant increase after the PD (p<=0.01, n=31).

Figure 4: Pre and Post Test Averages (n=31)

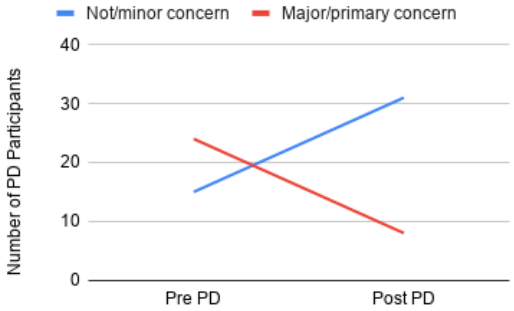


We also gave a retrospective post-PD survey to teachers (n=39 out of the 54 PD participants) which was based on validated CS education instruments including the BASICS Questionnaires and Interviews [19], CS4HS Teacher Professional Development [31], and WeTeach_CS motivation survey [20] instruments. In post-PD surveys, 92% of PD participants (n=39) agreed or strongly agreed that the PD sessions for CSAwesome significantly improved their ability to teach CS content and 64% agreed or strongly agreed that the CS Awesome PD was helpful in making the transition from AP CSP to AP CS A.

On a scale of 0 to 100 (beginner to expert), teachers on average self-rated their CS experience level as 19 points higher after the PD (p<=0.00, n=39), with an average of 48% before the PD and 67% after the PD. After the PD, 67% (n=39) of teachers strongly agreed that teaching AP CS A Java will be rewarding and worth the effort compared to only 41% before the PD.

Teacher confidence and ability to teach CS content was also positively impacted by the PD. Of teachers who completed the PD, 93% (n=39) agree or strongly agree that CSAwesome has significantly improved their ability to teach CS content. Before the PD, 62% (24 out of 39) of teachers indicated they had major or primary concerns about teaching Java. After the PD, 79% indicated that teaching Java was not a concern or a minor concern, and only 21% still indicated it was a major or primary concern (p<=0.00), see Figure 5.

Figure 5: Post-PD retrospective survey (n=39): I am concerned about teaching Java (before and after the PD)



During the pilot PD, we ran into challenges and learned from these to adapt the PD. The first PD included new and experienced teachers which made organizing and pacing the PD content difficult. The second PD was just for teachers who were new to CSA, but we found that they still had varying knowledge of programming and Java. We found that it is essential to gather more information about the knowledge levels of the teachers and to concentrate the PD on effectively teaching CSA, rather than helping teachers become proficient Java programmers. Differentiated PD that meets teachers at their knowledge levels, particularly with the variety of backgrounds that teachers bring to CS, is essential in CS PD. Another difficulty that we ran into was retention of the participants during the academic year and during the Covid-19 pandemic. We hope to have better retention in the summer.

5 SUMMARY AND FUTURE WORK

This paper presents CSAwesome, a new curriculum and professional development provider for the AP CSA exam, as well as initial results from the curriculum and pilot PD during the first year of use. The goals of CSAwesome are to broaden participation in the AP CSA course and to support new-to-CS students and teachers as they transition from the AP CSP course to the AP CSA course by using inclusive teaching practices and curriculum design. This new curriculum supports student engagement and scaffolded learning through an interactive ebook with embedded executable and modifiable code (Active Code), a variety of practice types with immediate feedback, and adaptable mixed-up code (Parsons) problems. The ebook design is based on the theories of constructivism, cognitive load, desirable difficulties, and social constructionism. The CSAwesome PD follows effective professional development and learning theory and incorporates inclusive teaching, equity, and active recruitment strategies with the goal of broadening participation in CSA. Results from the first two PD cohorts show improvements in teachers' content knowledge, confidence, and ability to teach CSA.

During the summer of 2020, CSAwesome ran differentiated PD for over 200 teachers in three different formats: Intro to Java (20 hours) for teachers who want to explore learning Java, Extended PD (65 hours) for CSP teachers who are new to Java and will be teaching CSA for the first time in 2020, and Immersion PD (45

hours) for teachers who are more experienced in Java or coding and want to learn how to teach AP CSA with CSAwesome. The teachers in the extended PD will take part in our research study to gather data on their progress and their students' progress with pre/post tests and surveys.

The CSAwesome curriculum will also be improved using teacher input and participation. In one large project completed this summer, teachers volunteered to write JUnit tests for all of the embedded active code in the ebook to make them auto-gradable and to provide immediate feedback to students. We will also analyze the anonymous clickstream data from CSAwesome to investigate how learners actually use the ebook and to identify areas for improvement.

6 ACKNOWLEDGEMENTS

We would like to acknowledge the support of a Google Computer Science Education Research grant during this project in 2019-2020. We would also like to thank the CSAwesome team, especially Pauline Lake and Cassandra Beckworth, the CSAwesome PD facilitators and teachers, the National Math and Science Initiative for supporting development of teacher resources, and Brad Miller for creating the open-source Runestone platform and adding new capability to support Java to host the CSAwesome course.

REFERENCES

[1] Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research* 70, 2 (2000), 181–214.

[2] Laura E Berk and Adam Winsler. 1995. *Scaffolding Children's Learning: Vygotsky and Early Childhood Education. NAEYC Research into Practice Series. Volume 7.* ERIC.

[3] Elizabeth L Bjork, Robert A Bjork, et al. 2011. Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. *Psychology and the real world: Essays illustrating fundamental contributions to society* 2, 59-68 (2011).

[4] College Board. 2019. *Participation in AP Computer Science Principles More Than Doubles 3 Years After Launch.* https://www.collegeboard.orgreleases/2019/participation-csp-nearly-doubles

[5] College Board. 2020. *AP Computer Science A Course and Exam Description.* https://apcentral.collegeboard.org/pdf/ap-computer-science-a-course-and-exam-description.pdf

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

[7] Code.org. 2019. *TLO Guide 2019-2020.* https://docs.google.com/document/d/1IJV3A9F6nTuUQ3hRPH8a2KfeEiPPPkr6exbCY7E3osM/edit

[8] James Cohoon, Joanne M Cohoon, and Mary Lou Soffa. 2011. Focusing high school teachers on attracting diverse students to computer science and engineering. In *2011 Frontiers in Education Conference (FIE)*. IEEE, F2H–1.

[9] National Research Council et al. 1992. *In the mind's eye: Enhancing human performance.* National Academies Press.

[10] CSAwesome. 2020. *CSAwesome curriculum and PD provider for AP CSA.* csawesome.org(lessonplans)andhttps://runestone.academy/runestone/static/csawesome/index.html(curriculum)

[11] Wanda Dann, Dennis Cosgrove, Don Slater, Dave Culyba, and Steve Cooper. 2012. Mediated Transfer: Alice 3 to Java. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (Raleigh, North Carolina, USA) *(SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 141–146. https://doi.org/10.1145/2157136.2157180

[12] Linda Darling-Hammond, Maria Hyler, and Marilyn Gardner. 2017. *Effective teacher professional development.* Palo Alto, CA. https://learningpolicyinstitute.org/sites/default/files/product-files/Effective_Teacher_Professional_Development_REPORT.pdf

[13] Barbara Ericson. 2020. *Slides on AP CSA and CSP Data.* Retrieved 2020 from https://cs4all.home.blog/2020/01/13/slides-on-ap-csa-and-csp-data

[14] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[15] Barbara J Ericson, James D Foley, and Jochen Rick. 2018. Evaluating the efficiency and effectiveness of adaptive parsons problems. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 60–68.

[16] Barbara J Ericson, Mark J Guzdial, and Briana B Morrison. 2015. Analysis of interactive features designed to enhance learning in an ebook. In *Proceedings of the eleventh annual International Conference on International Computing Education Research*. 169–178.

[17] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 20–29.

[18] Barbara J. Ericson and Bradley N. Miller. 2020. Runestone: A Platform for Free, On-Line, and Interactive Ebooks. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 1012–1018. https://doi.org/10.1145/3328778.3366950

[19] Outlier Research & Evaluation. 2017. *BASICS Study ECS Student Implementation and Contextual Factor Questionnaire Measures [Measurement scales].* http://outlier.uchicago.edu/basics/resources/Measures-StudentImplementation/

[20] C. Fletcher, L. Garbrecht, W. Monroe, C. Ramsey, and J. Warner. 2017. Analyzing the impact of teacher perceptions of computer science on computer science certification success. In *American Educational Research Association Proceedings*.

[21] Joanna Goode, Jane Margolis, and Gail Chapman. 2014. Curriculum is not enough: The educational theory and research foundation of the exploring computer science professional development model. In *Proceedings of the 45th ACM technical symposium on Computer science education*. 493–498.

[22] Philip J Guo. 2013. Online python tutor: embeddable web-based program visualization for cs education. In *Proceeding of the 44th ACM technical symposium on Computer science education*. 579–584.

[23] Colleen Lewis. [n.d.]. *CS Teaching Tips.* https://www.csteachingtips.org/

[24] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald. 2002. The effects of pair-programming on performance in an introductory programming course. *ACM SIGCSE Bulletin* 34, 1 (2002), 38–42.

[25] Charlie McDowell, Linda Werner, Heather E Bullock, and Julian Fernald. 2006. Pair programming improves student retention, confidence, and program quality. *Commun. ACM* 49, 8 (2006), 90–95.

[26] NCWIT. [n.d.]. *Engage CS Edu.* https://www.engage-csedu.org/engagement/make-it-matter

[27] NCWIT. 2020. *By the Numbers.* https://www.ncwit.org/resources/numbers

[28] Seymour Papert. 1990. Children, computers and powerful ideas.

[29] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 157–163.

[30] Doug Rohrer and Kelli Taylor. 2006. The effects of overlearning and distributed practise on the retention of mathematics knowledge. *Applied Cognitive Psychology: The Official Journal of the Society for Applied Research in Memory and Cognition* 20, 9 (2006), 1209–1224.

[31] J. Blazevski S. Davis, J. Ravitz. 2018. Evaluating computer science professional development models and educator outcomes to ensure equity. In *2018 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT). IEEE.*

[32] 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. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (Portland, OR, USA) *(SIGCSE '20)*. 542–548.

[33] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[34] John Sweller. 2010. Cognitive load theory: Recent theoretical advances. (2010).

[35] John Sweller and Graham A Cooper. 1985. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction* 2, 1 (1985), 59–89.

[36] John Gregory Trafton and Brian J Reiser. 1994. *The contributions of studying examples and solving problems to skill acquisition.* Ph.D. Dissertation. Citeseer.

[37] Jeroen JG Van Merriënboer and Marcel BM De Croock. 1992. Strategies for computer-based programming instruction: Program completion vs. program generation. *Journal of Educational Computing Research* 8, 3 (1992), 365–394.

[38] Jeroen JG Van Merriënboer, Paul A Kirschner, and Liesbeth Kester. 2003. Taking the load off a learner's mind: Instructional design for complex learning. *Educational psychologist* 38, 1 (2003), 5–13.

[39] Barry J Wadsworth. 1996. *Piaget's theory of cognitive and affective development: Foundations of constructivism.* Longman Publishing.

[40] Iman YeckehZaare, Paul Resnick, and Barbara Ericson. 2019. A Spaced, Interleaved Retrieval Practice Tool That is Motivating and Effective. In *Proceedings of the 2019 ACM Conference on International Computing Education Research* (Toronto ON, Canada) *(ICER '19)*. Association for Computing Machinery, New York, NY, USA, 71–79. https://doi.org/10.1145/3291279.3339411