JCSI
Journal of Computer Science Integration

Volume 1 | Issue 1 Article 2

2018

Lessons Learned from a District-wide Implementation of a Computer Science Initiative in the District of Columbia Public Schools

Kenneth Alonzo Anderson
Howard University

Legand L. Burge III
Howard University

Troy J. Shine
Howard University

Marlon Mejias
UNC Charlotte

Ketly Jean-Pierre
Howard University

Recommended Citation

Anderson, K. A., Burge, L. L., Shine, T. J., Mejias, M., & Jean-Pierre, K. (2018). Lessons Learned from a District-wide Implementation of a Computer Science Initiative in the District of Columbia Public Schools. *Journal of Computer Science Integration, 1* (1). https://doi.org/10.26716/jcsi.2018.01.1.2

JCSI
Journal of Computer Science Integration

Lessons Learned from a District-wide Implementation of a Computer Science Initiative in the District of Columbia Public Schools

Abstract
In this article, we use evidence to describe seven key lessons from a four-year district-wide computer science implementation project between Howard University and the District of Columbia Public Schools. These lessons are: (a) Get to know the school counselors (and other key personnel); (b) Expect personnel changes and strategic reorganization within school districts; (c) Be innovative to build and maintain community; (d) Be flexible when developing instruments and curricula; (e) Maintain a firm commitment to equity; (f) Develop tiered content and prepare to make philosophical adjustments; and (g) Identify markers of sustainability. We also include original curricula materials including the *Computer Science Course Evaluation* and the *Computational Thinking Survey*. The seven lessons and curricula materials provided in this study can be used to inform the development of future computer science researcher-practitioner partnerships.

Keywords
computer science, high school, computational thinking

DOI
10.26716/jcsi.2018.01.1.2

Corresponding Author
Kenneth Alonzo Anderson, *Howard University*, School of Education, Department of Curriculum & Instruction, 2441 4th St. NW, Washington, DC 20059. Email: kenneth.anderson@howard.edu

Funder Name
National Science Foundation

Funder or Grant ID
1240822

Cover Page Footnote
We would like to acknowledge Dr. A. Nicki Washington for her foresight and vision to start this project. We would also like to thank Ms. Qi'Anne Knox and Mr. Jarreau Moulden Tarter for their contributions.

Lessons Learned from a District-wide Implementation of a Computer Science Initiative in the District of Columbia Public Schools

In recent years, workforce demand for computer scientists and employees with strong computational skills have increased exponentially. To meet this demand, school districts in the United States have begun increasing their computer science offerings. In addition, the National Science Foundation has launched several initiatives to enhance computer science instruction and computational thinking throughout the United States. Some of these initiatives include the *Broadening Participation in Computing Alliance, Computer Science for All,* and *Computing Education for the 21st Century Programs* (National Science Foundation, 2018a; 2018b; 2018c). In this study, we describe lessons learned from a researcher-practitioner partnership (RPP) titled *The Partnership for Early Engagement in Computer Science-High School* (PEECS-HS), sponsored by the *Computing Education for the 21st Century Program.* PEECS-HS is a four-year collaboration between two primary partners: Howard University (Departments of Computer Science and Curriculum and Instruction) and the District of Columbia Public Schools (DCPS). Our project included additional critical partners, including Exploring Computer Science, a group dedicated to improving K-12 computer science, and Google, Incorporated.

Review of Literature

Computer Science and Computational Thinking

Computer science can be defined as the study of "fundamental properties of information processes, both natural and artificial" (Denning, 2009, p. 6). Computer science principles can be grouped into seven categories: computation, communication, coordination, recollection, automation, evaluation, and design. Moreover, computer science is comprised of four core practices: programming, engineering of systems, modeling, and applying (Denning, 2009). When developing computer scientists, a major aim is for learners to master a major computer science practice, computational thinking. Computational thinking, or the notion of algorithmic, or ordered and precise sequencing to solve a problem, has evolved since the 1950s and 1960s. Broader conceptualizations of computational thinking include problem solving, as well as concepts such as designing systems, understanding human behavior, efficiency, recursive thinking, abstraction, and decomposition of tasks (Wing, 2006). Although computational thinking is not limited to computer science, it is a key focus in computer science education.

K-12 Computer Science Education

In recent years, efforts to expand computer science and computational thinking at the K-12 level in the United States can be attributed to several factors. Some factors include the high financial and social costs of cyber security breaches (Krishan, 2018), large numbers of U.S. based computer scientists who were born abroad (Bound, Braga, & Khanna, 2015), job outlook and salary (Bureau of Labor Statistics, 2018), and the desire for the United States to maintain its global competitiveness (Smith, 2016). In 2016, President Obama launched a *Computer Science for All* initiative, noting the rapidly expanding computer science field and its critical relationship to the economy. Moreover, estimates have suggested that, in 2015, more than 600,000 high

paying technology jobs were unfilled, largely related to a large share of the workforce lacking the appropriate skills to fill vacancies (Smith, 2016).

Great emphasis has been placed on expanding the computer science workforce to include more individuals from traditionally underrepresented communities. In a seminal report published through a partnership between Gallup and Google, Inc., several K-12 key insights and disparities were noted. Using nationally-representative student interview data and comprehensive national survey data of principals and superintendents, the authors found that (a) computer science is valued by students, teachers, parents, and administrators, (b) administrators reported that computer science was not a high priority (relative to other core subjects), (c) disparities in access to technology for many Hispanic students was problematic, and (d) computer science courses in many lower income and Black communities lacked key components (Google Inc. & Gallup Inc., 2015). In a second report, the authors noted that computer science learning opportunities in schools were increasing rapidly, but these the learning opportunities varied widely. Specifically, the authors noted that computer science was offered in the form of clubs or classes in 76% of all the schools and 88% of the high schools (Google Inc. & Gallup Inc., 2016). Additional evidence has shown that computer science offerings are increasing as well. The Education Commission of the States published a policy brief indicating that 20 states have revised traditional graduation requirements (e.g. pre-determined number of courses), to allow computer science to fulfill a graduation requirement (Zinth, 2016). Most states allowed eligible computer science courses to partially fulfill mathematics or science requirements whereas one state allowed eligible computer science courses to meet language requirements.

Researcher-Practitioner Partnerships

As noted in the previous section, computer science education is a critical area of need in the United States. Although computer science offerings are increasing, it is critical to implement computer science in ways that are beneficial to students and other stakeholders. One approach to develop sound computer science education practices is through researcher-practitioner partnerships (RPPs). RPPs have existed for quite some time. In recent years, researchers have provided guidance for defining RPPs to distinguish them from traditional research and traditional School-University partnerships that involve placing student teachers and other pre-service professionals in schools to earn practical experience (Coburn, Penuel, & Geil, 2013). Coburn et al. argue that RPPs differ from traditional research and traditional School-University partnerships in five ways. They assert that RPPs (a) are long-term, (b) focus on problems of practice, (c) are committed to mutualism, (d) use intentional strategies to foster partnerships, and (e) produce original analyses. In addition, the authors note that there are three types of RPPs: research alliances, design research, and networked improvement communities (NIC). Research alliances are long-term partnerships between districts and independent research organizations. Within a research alliance, research questions are negotiated and carried out using feedback loops to improve policy and practice. Design research simultaneously develops and studies solutions in real contexts, usually focusing instructional activities and curricula development. NICs usually consist of clusters of groups (e.g. districts) that work together to improve capability across multiple settings. NICs usually involve trying to identify best practices for scalability.

DOI: 10.26716/jcsi.2018.01.1.2

Accordingly, we use the RPP framework of Coburn et al. to describe results and lessons learned from the Computer Science RPP between Howard University and the District of Columbia Public Schools.

About the Partnership

PEECS-HS was formed to introduce a new, introductory course, titled *Exploring Computer Science,* across DCPS high schools. The curriculum used for the PEECS-HS course is a modified version of the *Exploring Computer Science* (ECS) curriculum (Goode & Chapman, 2011), originally piloted in the Los Angeles Unified School District. Exploring Computer Science (ECS) emphasizes project-based courses that are designed to prepare high-school students to master computer science fundamentals using real-world, socially-relevant, and interdisciplinary applications. The modified ECS curriculum used in the District of Columbia consists of six units (a) Human-Computer Interaction, (b) Problem-Solving, (c) Web Design, (d) Introduction to Programming, (e) Computing and Data Analysis, and (f) Mobile Applications Development. Major project activities focused on summer and quarterly teacher professional development to address inequity in computer science outcomes, provide technical support with the software applications, support teachers with data analysis, provide in-depth support with the Mobile Applications Unit, and build professional community.

The ECS curriculum was offered in all eight District of Columbia wards (voting districts) and reached 1,851 verified students (See Figures 1-2). Student participants were primarily enrolled in grades 9-12. In 2016-2017, ECS was offered at a middle school in grades seven and eight. For school years 2013-14, 2014-15, 2015-16, and 2016-17, six DCPS teachers taught ECS in year one, nine taught in year two, nine taught in year three, and thirteen taught in year four, respectively. In addition, a total of 70 DCPS and non-DCPS teachers were trained over the project period.

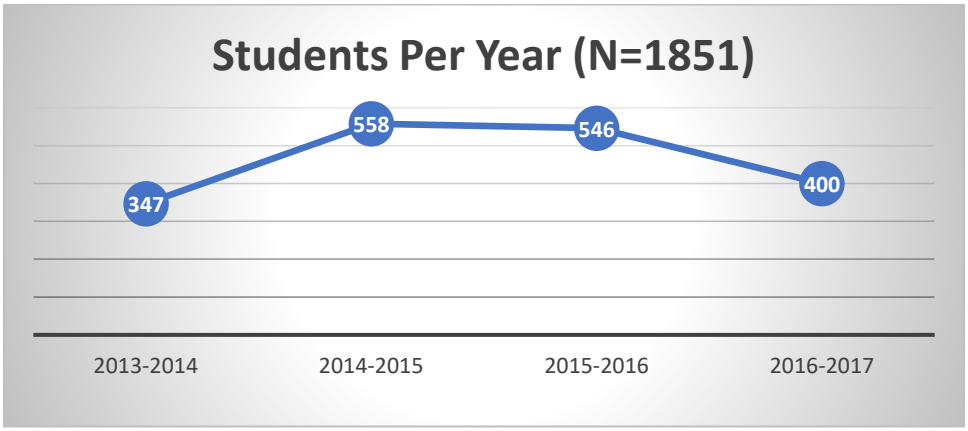


Figure 1. Enrollment trends over the project period

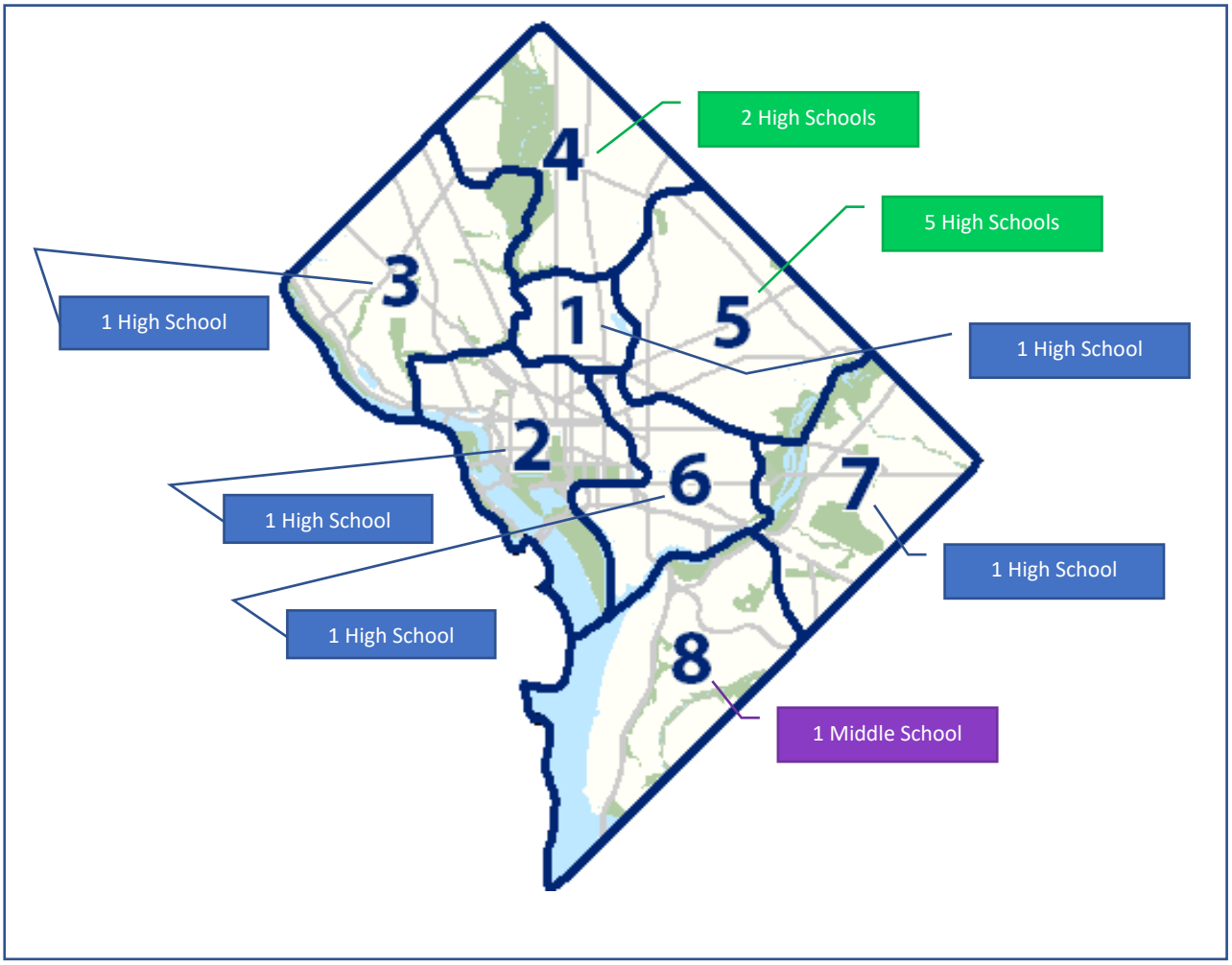


Figure 2. Map of ECS coverage in the District of Columbia from 2013-2017 by Ward (voting districts)

Method

Of the three RPP types noted, our work is most consistent with design research. The initial proposal was not set up as an RPP, but we quickly learned that our initially proposed project activities were constantly changing and the project evolved into an RPP. Consistent with the RPP literature, we describe the iterative process that we used to actualize the RPP, lessons learned from the activity, and recommendations for other districts to consider when implementing a computer science program. We organize our results by lessons learned, describing the original research and experiences that we used to frame these lessons. We conclude by providing recommendations for future research.

Lesson 1: Get to know the school counselors (and other key personnel)

Consistent with Coburn et al.'s recommendation, using intentional strategies to foster RPPs is essential. As will be described next, developing essential relationships with key

personnel was essential to the success of our RPP. During the planning (for implementation) phase of this project, we began by communicating primarily with the school principals. However, we noticed that some of the principals would regularly invite the school counselors to these meetings. School counselors often play a major role in course scheduling and assigning students to courses. As such, they can forecast enrollments from feeder schools and enroll students into innovative courses. In addition, school counselors often have a strong sense of how course credits can count towards the various academic pathways in high school. For example, whereas computer science is often thought of as a STEM credit, some of the partnering counselors understood the logistical challenges of adding another STEM credit to the curriculum. Yet, they saw opportunities to offer the *Exploring Computer Science* course as a career and technical education (CTE) credit.

Another benefit of getting to the know school counselors is that they can identify potential implementation challenges of new courses. The ECS course was designed for 50-minute blocks, Monday through Friday. However, many counselors described a host of scheduling patterns. To provide perspective, the various scheduling blocks and wide range of students that were to be accommodated at each school at the beginning of the project are shared in Table one. These data show that the counselor who had to accommodate 175 students into "A" and "B" day schedules likely encountered far greater challenges than the counselor who only had to accommodate 23 students into schedules where the same classes met daily.

Table 1
Summary of Schedule Types and Enrollment at the Beginning of the Project

DCPS School	Number of Students Enrolled	Schedule Type
School 1	31	Rotating schedule (A-B days)
School 2	23	Daily
School 3	38	Rotating schedule (A-B days)
School 4	28	Rotating schedule (A-B days)
School 5	49	Alternate on Monday's, odd, even days
School 6	175	Rotating schedule (A-B days)
Total Initial Enrollment	344	

Some of our principals also shared concerns over offering ECS as a full credit course. The inability to work out the credit issues prevented at least one school from joining the project. We discussed several different options, but accommodating ECS through a credit format did not work well for the school at the time. Although our project proposed to offer ECS as a single course throughout the entire school year, the unexpected challenges that we faced and high levels of local control made us realize that more flexibility was necessary to enhance computer science instruction throughout DCPS. Thus, we began operating more like an RPP, than a traditional research project. Reflecting on this situation, we likely could have made accommodations by utilizing a module-infusion approach with existing courses. In cases like these, school counselors (and other personnel) can advocate for creative ways to deliver the content (e.g. half credits and other creative approaches). Our findings regarding the value of school counselors are consistent with Greene and Stewart's (2016) findings in that principals and school counselors are critical to successful collaboration.

Lesson 2: Expect Personnel Changes and Strategic Reorganization within School Districts

A distinguishing component of RPPs is that they are long-term (Coburn, Penuel, & Geil, 2013). Long-term partnerships offer several benefits. However, the long-term nature of RPPs can also be challenging, especially regarding district-level personnel and strategic direction changes. In our case, central office personnel changes caused issues, ranging from minor to major. From a research perspective, we worked with three key personnel in the research division of the central office over the life of the five-year project (including the planning year). These changes were fairly smooth, but usually required additional conference calls and sending detailed emails to provide new staff with overviews, project updates, descriptions of data agreements, and data needs.

In terms of program implementation. Our primary contacts were usually directors or program specialists of divisions (technology, CTE, science) that reported to Assistant Superintendents or some similar reporting structure. These directors played a major role in encouraging teachers to consider teaching computer science courses and recruiting teachers to the professional development sessions. However, over the five-year span (including the planning year), our key contact person changed four times. At times, the transitions were not disruptive, but at times they were. At times ECS was a high priority for some central office personnel and at times, less so. For example, during one of the reorganization periods, new central office specialists were not sure to which STEM sub-discipline to re-assign our project (career, science, or technology). Considering the interdisciplinary nature of computer science, the new central office personnel had multiple conversations about where to assign responsibility for our existing project. In the final phase of the project, we ended up working with multiple specialists across the STEM spectrum.

Leadership changes and reorganization can also lead to a change in program direction. During leadership changes, it is critical to refrain from being offended if an existing project is no longer a top priority. School district needs and resources change and it is important to see how existing projects may fit within new initiatives. For example, DCPS worked with some of its partners to develop an industry certification pathway for computer science. When finalizing the

DOI: 10.26716/jcsi.2018.01.1.2

recommended course sequence, our partnering specialist was not able to fit the ECS course into the required course sequence. However, ECS was included as a recommended course. Lesson 2 is also related to Lesson 3.

Lesson 3: Be Innovative to Build and Maintain Community

As shown in the previous section, it is important to build community within RPPs and avoid becoming overly dependent on one or two key personnel. Community allows the project to thrive when personnel changes occur. When the project began, we had great momentum and teachers were excited, especially during the summer professional development sessions. Over time, teacher participation began to drop significantly. Coupling reduced participation with personnel changes led to some concern from our team and seemed to affect the morale of the faithful teachers (and the research team) who regularly participated in the summer and quarterly professional development sessions. To address this, we had to try different strategies to build community. One of the challenges was that many of the teachers taught ECS in isolation. Our efforts to build community through electronic communication and online platforms did not work as planned, so we had to continue to try to build community in other ways. DCPS promotes local control, where school-level (rather than central-level) decision-making is a common operating practice. Thus, when central office personnel made recommendations to participate in professional development workshops or complete data requests, the recommendations were optional.

One approach that we used to build community was seeking teacher perspectives and leveraging the resources afforded by our computer science industry partners. Once we began rotating the event sites and professional development session locations between industry partner workspaces and University settings, we regained the project's momentum. We learned that there is no "silver bullet" to building community, but a range of activities helped to establish community. Some of these activities included offering a Computer Science summit, where our partner teachers and teachers outside of the partnership were invited. The only requirement was to have interest in teaching or offering computer science. As a result, we were able to attract additional teachers and administrators from charter and private schools. We also found that providing opportunities for our partner teachers to see the workspaces of industry partners provided additional motivation and practical insight that they could share with their students. The industry settings naturally built community among our teacher partners. Thus, when we hosted some sessions at the University, we believe that the community that was established at industry sites motivated teachers to participate when the sessions were held at University settings.

Our design thinking exercise, which allowed teachers to help us design activities that fostered community, also enhanced community. The design thinking exercise reflects mutualism, as discussed by Coburn et al. (2013), by prompting participants and project leaders to work together to integrate solution-focused thinking with the intent of producing a constructive future result. Participants started off by brainstorming ideas, with few, or no, limits on breadth during the initial phase. The ECS teachers were split into teams and worked within their teams to encourage input and participation from a wide variety of perspectives. Eleven DCPS teachers and project staff participated in the design thinking exercise. The eleven teachers and staff were grouped into teams (2-3 per group) to address three questions (a) How might we better engage teachers to respond to assessments? (b) How might we build a computer science teacher community in the district? and (c) How might we promote/showcase student talent and teacher achievements? Within this activity we were able to incorporate a variety of recommendations, such as more incentives, peer sharing, visits to industry sites, such as Google, Inc., and guest speakers from the various sites. Table two provides a summary of some of our most successful community-building activities, which reflect Coburn et al.'s (2013) essential RPP component of using intentional strategies to foster partnerships.

Table 2
Summary of Successful Community-building Activities

Activity	Description
Computer Science Education Summit	This event was created for teachers, administrators, industry leaders, and others who had an interest in expanding computer science in DC schools. The Summit was a huge success. The agenda included an ice breaker, an overview of PEECS-HS, a summary of the year three report, a summary of feedback received from the National Science Foundation, an overview of computational thinking materials developed by SRI Education (http://pact.sri.com), initial evaluation and feedback on the proposed Computational Thinking Survey, possible next steps for PEECS-HS, a design thinking exercise to increase participation, and a tour of Google.
DC K-12 CS Teacher Mixer	Invited K-12 CS teachers and administrators within the District to a mixer event at the Google DC office. The event provided opportunities to network with colleagues and administrators and learn about resources and opportunities for schools.
CSTA Open House	Invited teachers from DC and northern Virginia to Google DC to discuss the establishment of a combined DC-NoVA CSTA Chapter. Topics of discussion included the different pathways to computer science for K-12 students and upcoming professional development opportunities for ECS and CSP.

DOI: 10.26716/jcsi.2018.01.1.2

Lesson 4: Be Flexible when Developing Instruments and Curricula

At the onset of the project, we went through a rigorous IRB process at the University and district-level to get various instruments approved. During the planning phase of the PEECS-HS project, a 27-item Likert-type course evaluation was developed. Students were given five response options: strongly agree, agree, unsure, disagree, and strongly disagree. Course evaluations were developed using the Nine Principles of Learning: Organizing for Effort, Clear Expectations, Fair and Credible Evaluations, Recognition of Accomplishments, Academic Rigor in a Thinking Curriculum, Accountable Talk, Socializing Intelligence, Self-Management of Learning, and Learning as Apprentice adopted by the ECS curriculum development team (Goode & Chapman, 2011). Approximately three items were developed for each Principle. Item responses were reversed coded from 1-5 so that higher values represent higher agreement and lower values represent lower agreement with the items.

Upon implementation, it became obvious that our instruments lacked mutualism. Given the low response rates (11% of students) from the course evaluation, we were only able to retrieve one year of useful data. We learned that the major hurdle, relative to teacher evaluations, was climate. At the time, DCPS had recently implemented a high-stakes teacher evaluation system known as IMPACT (District of Columbia Public Schools, 2009). IMPACT was used to award bonuses, rate teachers, place them on improvement plans, and even remove teachers from the classroom. IMPACT was a highly contentious topic that gained national attention. By simply having conversations with the teachers, we learned that tensions were high regarding evaluations and many were not sure who would see the data and how it might indirectly or directly influence their formal IMPACT evaluations. Moreover, teachers mentioned the union contract, an issue that we simply had not considered. Although our course evaluations were not at odds with the union contract, there seemed to be some distrust about how the data might somehow be accessed by their evaluators, despite our assurances that the evaluations were only being used for program improvement. Thus, due to the climate, we discontinued use of the course evaluation after the first year. Despite discontinuing its use after the first year, we were able to salvage some results and offer a course evaluation instrument to the computer science community (see Appendix A).

Construct validity of the course evaluation instrument was assessed using confirmatory factor analyses. Considering that the number of completed course evaluation instruments was small (N = 38), there was insufficient power to obtain a factor solution for the full course evaluation instrument. Thus, a model generation approach (Joreskog, 1993) was employed to see if any factors would emerge. First a series of exploratory factor analysis models were run to eliminate items that did not correspond to a factor. Thereafter, confirmatory factor analysis was conducted. Modification indices were examined and cross-loaded items (.30 or greater) were deleted. As a result of the model generation approach, a well-fitting, three-factor model was retained, $N=36$, X^2 (24) =21.68, p = .59; *CFI*= 1.00; *TLI*=1.06; *RMSEA*=0.00 (90% *CI*: 0.00, 0.12), pclose = .70. The three retained factors were engagement, rigor, and clarity. Table 3 shows the item descriptions and associated descriptive statistics. For items three, four, and thirteen, none of the students agreed or strongly agreed, indicating concerns regarding rigor and clarity.

Relationships between the course evaluation factors and student outcomes were examined using a series of multiple regression models. Table four shows that no course evaluation factors (engagement, rigor, or clarity) were related to course GPA. However, clarity was significantly related to overall unit average (see Table 4). Despite the small sample size, a positive relationship between clarity and unit average was detected. Findings indicate that clear directions and feedback regarding student progress towards meeting expectations were predictive of student scores on the unit assignments. Notably, students also rated item four (My teacher gave clear instructions on what was expected of me) lower than any other item (see Table 3).

Table 3
Descriptive Statistics of Course Evaluation Items

Item		*N*	Mean	Standard Deviation	Minimum	Maximum
	Engagement					
15	Classrooms discussions focused on computer science topics.	37	3.76	0.89	1	5
22	In this class, I was required to use feedback to make improvements.	38	3.00	0.84	1	4
25	In this course, I was able to learn from computer science experts other than my teacher.	37	3.62	0.98	1	5
	Rigor					
3	My teacher set high expectations for me.	38	2.42	0.60	1	3
7	Tests in this class were fair, but challenging.	38	3.02	0.79	1	4
13	The knowledge gained in this course will help me if I decide to take a more advanced computer science course.	38	2.45	0.65	1	3
	Clarity					
4	My teacher gave clear instructions on what was expected of me.	38	2.18	0.61	1	3
6	Class assignment directions and course requirements were clear.	37	3.86	0.98	1	5
11	My teacher frequently recognized my progress in the course.	38	3.97	1.00	1	5

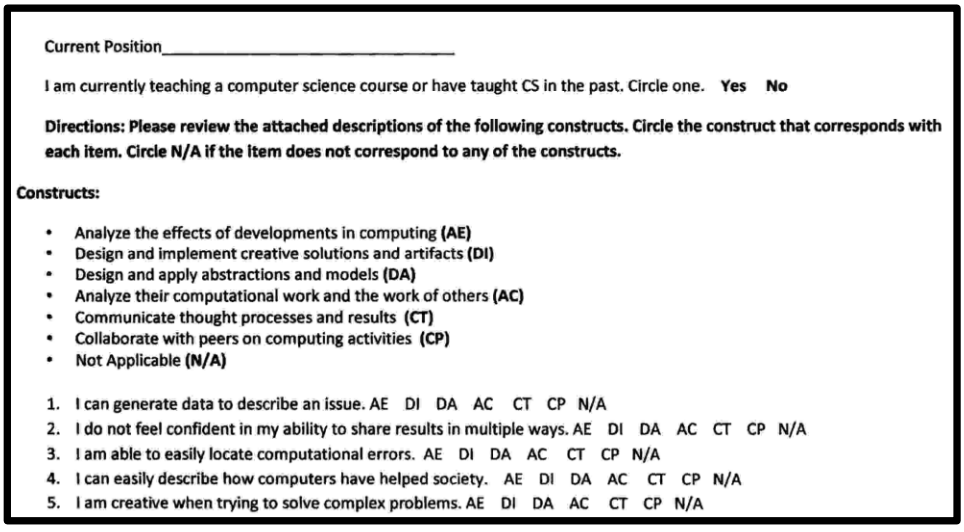
DOI: 10.26716/jcsi.2018.01.1.2

Table 4
Multiple Regression of Teacher Traits Predicting Unit Average (N = 38)

Variable	Overall *B*	95% CI
Constant	3.34	[1.15, 5.54]
Engagement	-0.02	[-0.18, 0.14]
Rigor	-0.15	[-0.43, 0.14]
Clarity	0.14*	[0.01, 0.26]
R2	0.75	
Adjusted *R2*	0.56	
F	3.93	

Note. CI = confidence interval
*p<.05

Based on the lessons learned and low response rates, we developed the Computational Thinking Survey (CTS) using mutualism. Unlike the course evaluation, we shared some suggested items with teachers that were based on the design patterns of computational thinking (See SRI International, 2018). We began by providing an overview of the computational thinking domains and asked participants to assess the content validity of our proposed items using validity prompts (See Figure 3 for sample validity prompts). We also asked teachers for suggestions and received invaluable feedback. Our mutual development approach allowed us to conduct original analysis (See Coburn et al., 2013) and offer the CTS to the broader computer science community.



Current Position_____

I am currently teaching a computer science course or have taught CS in the past. Circle one. **Yes** **No**

Directions: Please review the attached descriptions of the following constructs. Circle the construct that corresponds with each item. Circle N/A if the item does not correspond to any of the constructs.

Constructs:

- Analyze the effects of developments in computing **(AE)**
- Design and implement creative solutions and artifacts **(DI)**
- Design and apply abstractions and models **(DA)**
- Analyze their computational work and the work of others **(AC)**
- Communicate thought processes and results **(CT)**
- Collaborate with peers on computing activities **(CP)**
- Not Applicable **(N/A)**

1. I can generate data to describe an issue. AE DI DA AC CT CP N/A
2. I do not feel confident in my ability to share results in multiple ways. AE DI DA AC CT CP N/A
3. I am able to easily locate computational errors. AE DI DA AC CT CP N/A
4. I can easily describe how computers have helped society. AE DI DA AC CT CP N/A
5. I am creative when trying to solve complex problems. AE DI DA AC CT CP N/A

Figure 3. Sample content validity prompts for the Computational Thinking Survey

Our CTS validation process showed that participants agreed on the alignment of items to constructs 70% of the time. There were six items that did not garner at least majority agreement and were therefore discarded. To evaluate the degree to which participants agreed or disagreed

by chance (e.g. just randomly selecting items), kappa statistics were calculated and common guidelines for interpreting kappa statistics of interrater agreement were used. Interrater agreement ranged from moderate to substantial when accounting for chance (See Table 5). Additional qualitative feedback was gathered to make minor improvements to the language of the instrument as well. The CTS items are described in Appendix B.

Table 5
Kappa Statistics Representing Levels of Agreements for Each Construct

Construct	Kappa	z
Analyze the effects of developments in computing.	0.43	16.15*
Design and implement creative solutions and artifacts.	0.46	17.06*
Design and apply abstractions and models.	0.50	18.58*
Analyze their computational work and the work of others.	0.44	16.39*
Communicate thought processes and results.	0.59	21.98*
Collaborate with peers on computing activities.	0.68	25.11*

*p<.05, Kappa statistics of .41-.60 represent indicate moderate agreement .61 - .80 are represent substantial agreement.

When developing curricula, it is important understand the assumptions that are embedded and the audience for which the curricula is designed. One assumption of ECS is that students have basic computer knowledge regarding storing, organizing, and retrieving files. The widespread assumption that young students are highly computer literate is somewhat misleading. Many young students are highly familiar with communication and social media platforms, but these platforms often have pre-built structures that help users organize information. These platforms are also commonly used through small devices such as cellular phones. Per feedback from our partner teachers, many students lacked efficient keyboarding and basic computer operation skills (e.g. organizing information, creating folders and subfolders). Thus, teachers would spend time on these topics, which meant less time on ECS topics. Therefore, we provided some supplemental curriculum material to support teachers with the teaching of basic computing skills.

Flexibility was also necessary regarding the core ECS content. One of the original ECS units focused on robotics. However, based on lessons learned from a previous project, we understood the hardware challenges of offering robotics, such as replacement costs and proper storage. Therefore, we modified the ECS curriculum with a mobile applications unit. This modification allowed us to achieve similar goals by offering a cloud-based alternative that was highly engaging and relevant. Interestingly, the mobile applications unit is the final and generally most engaging unit. However, some teachers felt that they often ran out time to fully complete this unit. They also felt that if they could get to this unit earlier, then there would be great potential to expand the interest in computer science to more students. We worked with the teachers in the professional development sessions to promote quicker progression to the mobile applications unit; however, there is not always a “quick fix” for prerequisite knowledge. Overall, our experiences highlight the essentiality of flexibility with curricula and instrument development.

DOI: 10.26716/jcsi.2018.01.1.2

Lesson 5: Maintain a Firm Commitment to Equity

Equity in computer science is essential and multi-faceted. We found that most of our teachers generally agreed with the importance of equity in computer science. However, beliefs and passion alone are not enough to achieve equity. Our findings show that the project exposed a high number of underrepresented students to computer science. Of the 1,851 verified students that were served, approximately 80% were from underrepresented racial categories (American/Black and Hispanic/Latino, and slightly more than 40% were female. We regularly conducted equity analyses and often found equitable outcomes when comparing males to females. However, lower grades were often awarded to African American students compared to White students and to students with disabilities compared to those who did not have disabilities. For example, after year three, a random intercepts regression model (students nested within classrooms) was used to examine equity in outcomes by demographic groups. The model shows that demographic factors (race, ethnicity, special education status, and gender) explained approximately four percent of the differences in course grades earned. While there were some differences by race, differences in classroom teachers explained about 19% of the variance in grades assigned (see Table 6). This could mean that variation in classroom climates, grading practices, and teaching styles influenced outcomes.

Table 6
Random Effects Regression Estimates (students nested in classrooms) of Race, Ethnicity, and Gender predicting ECS GPA (N=451)

Parameter	Null Model	Model
Intercept	2.77 (0.17)*	2.61 (0.16)*
Level 1		
*Race**		
Asian		0.57 (0.29)
American Ind./ Alaskan Nat.		-0.41 (0.81)
Multi-racial		0.61 (0.29)
White		0.70 (0.22)*
Female		0.15 (0.10)
Disability		-0.39 (0.18)*
Hispanic		-0.36 (0.27)
Level 2		
Between	0.54	0.43
Within	1.08	0.12
P	0.19	0.14
R^2_1		0.04

$p \leq .05$. Note: Black students are the reference group.

The model suggests that despite some demographic differences, primarily race and disability differences, demographic factors are generally equitable across outcomes. Despite some differences by race and disability, our teachers generally did not request more equity

training. This may indicate an issue of reflective practice. Overall, our findings show that we achieved equitable results in many areas. Our findings also suggest that equity is multifaceted and, as project leaders, we must continue to press partnering districts on a range equity issues to achieve mutual goals. In the future, we must include more equity analysis tools to allow for greater teacher self-reflection, disaggregation of subgroup outcomes, and ensure that subgroup inequity is institutionalized during professional development sessions. Although computer science content needs are great, these findings highlight equally important computer science equity issues.

Lesson 6: Develop Tiered Content and Prepare to make Philosophical Adjustments.

Although teachers did not request additional equity training, there was clear evidence of requests for tiered content preparation. As an example, themes from our focus group data show that teachers tended to request more content training if they did not have prior computer science experience (See Table 7). We generally found that the discrepancy was between teachers with prior computer science experience and teachers with little or no computer science background. During the first summer professional development session, teachers without prior computer science experience frequently indicated that the pace was too fast, whereas experienced computer science teachers felt the pace was sufficient.

Table 7
Brief Summary of Focus Group Themes.

Theme	Summary
Collaboration	Teachers expressed appreciation for opportunities to collaborate. Collaboration was the most prevalent and positive theme throughout the focus groups.
PEECS-PD	Teachers generally enjoyed the professional development sessions and offered commendations for its structure and organization.
Computer Science Background	Comments from the focus groups were often divided between teachers with prior computer science backgrounds and teachers with no previous computer science backgrounds.
Content	Teachers with computer science backgrounds felt that content was sufficient whereas teachers with no prior computer science background felt that more content preparation was necessary.
Community of Practice	The entire cohort felt that the website would be a tremendous resource during the implementation phase of ECS.

At the outset, inquiry-based instruction permeated the ECS philosophy, thus the ECS professional development sessions reflected this philosophy. However, teachers often requested more instructor modeling of lessons, rather than having to teach demonstration lessons without any prior knowledge or little guidance. Research has shown that despite the popularity of minimally-guided instruction, strongly guided instruction is generally more effective than minimally-guided instruction (See Kirschner, Sweller, and Clark, 2006 for more information). After the first summer professional development sessions, we began providing a better balance between inquiry-based instruction and direct guided instruction. These adjustments enhanced the overall tenor of the professional development sessions and further enabled progress toward mutual goals of increasing computer science equity.

Lesson 7: Identify Markers of Sustainability.

Coburn et al. (2013) stressed the importance of having a long-term focus when establishing RPPs. It may be natural to focus on the length of the project when planning long-term partnerships. However, it is important to think beyond the length of projects when planning RPPs. Our most apparent marker of sustainability is that we were able to work with DCPS to get additional computer science courses added to the course catalog titled *Explore Computer Science* (Course V39) and Computer Science Concepts (V38). Course V39 is a one-credit course with the following course description:

> This is the introductory course to the Computer Science pathway, designed to introduce students to the breadth of computer science. The course does not focus on learning a particular software tool or programming language, but rather focuses on the conceptual ideas of computing so students understand why certain tools or languages might be utilized to solve particular problems. Students will be introduced to topics such as interface design, robotics, computers' strengths and limitations, as well as societal and ethical issues.

An additional one-half (1/2) credit course, Computer Science Concepts (V38), was also added to the catalog. The description reads:

> This course is designed to introduce students to computer science in a condensed introductory course. The course does not focus on learning a particular software tool or programming language, but rather focuses on the conceptual ideas of computing so students understand why certain tools or languages might be utilized to solve particular problems. Students will be introduced to topics such as interface design, robotics, computers' strengths and limitations, as well as societal and ethical issues.

The “V” in the course number represents career and technical education courses (e.g. vocational). Students are required to earn 24 credits and within these 24 credits, two must qualify as college level or career preparation courses. Course V39 qualifies for credit towards graduation, whereas V38 does not. Most students were enrolled into the full credit course (V39), but demand for the half credit course was noteworthy and increased to 47% of enrolled students in the final year. This offering reflects the need for flexible curricula.

In addition, we aimed to support sustainability by training master teachers. Doing so provides DCPS with additional human resources to support computer science instruction beyond the current project. We trained two master teachers using a progressive responsibility model and began a shared model of professional development delivery. In the third year, we, the project team, shared delivery of instruction with the master teachers. In the final year, the master teachers delivered a substantial portion of the professional development sessions and are prepared to deliver additional sessions without support from our team. We believe that long-term markers of success should be mutually developed in the early stages of RPP development, assessed regularly, and modified as necessary.

Conclusion

As described through seven key lessons, we provide a summary of key insight that can be used in future development of computer science related RPPs. In our descriptions of the seven key lessons, we demonstrate how the lessons reflect Coburn et al.'s (2013) five distinguishing factors of RPPs. We were able to address issues of inequity in access to computer science by offering the ECS course in all eight District of Columbia wards or voting districts. When the project began, we operated with two major goals: (a) increase the number of high-school students exposed to computer science earlier in their academic careers in an effort to prepare them for computer science courses, undergraduate programs, and careers; and (b) increase the number of in-service DCPS teachers implementing the ECS course. Results presented in this paper provide convincing evidence towards meeting these project goals. Our project, which operated more like an RPP, evolved significantly and allowed us to share insights gained therein.

In closing, we reiterate our seven lessons:

1. Get to know the School Counselors (and other key personnel).
2. Expect Personnel Changes and Strategic Reorganization within School Districts.
3. Be Innovative to Build and Maintain Community.
4. Be Flexible when Developing Instruments and Curricula.
5. Maintain a Firm Commitment to Equity.
6. Develop Tiered Content and Prepare to make Philosophical Adjustments.
7. Identify Markers of Sustainability.

It is our hope that future computer science projects will incorporate some of the insight from the seven lessons noted in this paper and the resources provided in the Appendices.

DOI: 10.26716/jcsi.2018.01.1.2

References

Bound, J., Braga, B., Golden, J. M., & Khanna, G. (2015). Recruitment of Foreigners in the Market for Computer Scientists in the United States. *Journal of Labor Economics, 33*(3), S187-223. https://doi.org/10.1086/679020

Bureau of Labor Statistics (2018). U.S. Department of Labor, *Occupational Outlook Handbook*, Computer and Information Research Scientists. Retrieved from https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm

Coburn, C.E., Penuel, W.R., & Geil, K.E. (2013). *Research-practice partnerships: A strategy for leveraging research for educational improvement in school districts.* William T. Grant Foundation: New York, NY. Retrieved from http://wtgrantfoundation.org/library/uploads/2015/10/Research-Practice-Partnerships-at-the-District-Level.pdf.

District of Columbia Public Schools (2009). IMPACT: The DCPS evaluation and feedback system for school-based personnel. Retrieved from https://dcps.dc.gov/page/impact-dcps-evaluation-and-feedback-system-school-based-personnel

Denning, P. J. (2009, June). Beyond computational thinking. *Communications of the ACM*, 52(6), 28-30. Retrieved from https://cacm.acm.org/magazines/2009/6/28490-beyond-computational-thinking https://doi.org/10.1145/1516046.1516054

Google Inc. & Gallup Inc. (2015). *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*. Retrieved from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

Google Inc. & Gallup Inc. (2016). Trends in the State of Computer Science in U.S. K-12 Schools. Retrieved from http://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf

Goode, J., & Chapman, G. (2011). *Exploring Computer Science Curriculum (Version 4)*. Retrieved from http://www.exploringcs.org/wp-content/uploads/2010/08/ExploringComputerScience-v4.0.pdf

Greene, N. & Stewart, P. (2016). The school counselor and the principal: Keys to successful collaboration. *New Hampshire Journal of Education*, Practitioners Speak. Retrieved from http://nhje.plymouth.edu/?article=the-school-counselor-and-the-principal-keys-to-successful-collaboration

Kirschner, Sweller, & Clark (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist*, 41(2), 75-86. https://doi.org/10.1207/s15326985ep4102_1

Krishan, R. (2018). Corporate solutions to minimize expenses from cyber security attacks in the United states. *Journal of Internet Law*, *21*(11), 16-19.

National Science Foundation (2018a). *Broadening Participation in Computing Alliance Program*. Retrieved from https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503593&org=CISE&from_org=CISE

National Science Foundation (2018b). *Computer Science for All*. Retrieved from https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505359&org=CISE&from_org=CISE

National Science Foundation (2018c). *Computing Education for the 21st Century*. Retrieved from https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=505359&org=CISE&from_org=CISE

Smith, M. (2016). *Computer Science for All*. Retrieved from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all#Need

SRI International (2018). SRI International and Principled Assessment of Computational thinking (PACT). Retrieved from https://pact.sri.com/

Wing, J. (2016, March). Computational thinking. *Communications of the ACM*, 49(3), 33-35. Retrieved from https://cacm.acm.org/magazines/2006/3/5977-computational-thinking https://doi.org/10.1145/1118178.1118215

Zinth, J. (2016). *Computer science in high school graduation requirements* (September 2016 Update). Retrieved from https://www.ecs.org/wp-content/uploads/09.13.2016_Computer-Science-in-High-School-Graduation-Requirements.pdf

DOI: 10.26716/jcsi.2018.01.1.2

Appendix A

Computer Science Course Evaluation Instrument

Directions: Please read the statements below. After reading each statement, circle the letters that show how much you agree or disagree.

SD = Strongly Disagree D = Disagree U = Unsure A = Agree SA = Strongly Agree

1.	My teacher set high expectations for me.	**SD**	**D**	**U**	**A**	**SA**
2.	In this course, I was able to learn from computer science experts other than my teacher.	**SD**	**D**	**U**	**A**	**SA**
3.	My teacher gave clear instructions on what was expected of me.	**SD**	**D**	**U**	**A**	**SA**
4.	In this class, I was required to use feedback to make improvements.	**SD**	**D**	**U**	**A**	**SA**
5.	Class assignment directions and course requirements were clear.	**SD**	**D**	**U**	**A**	**SA**
6.	Tests in this class were fair, but challenging.	**SD**	**D**	**U**	**A**	**SA**
7.	My teacher frequently recognized my progress in the course.	**SD**	**D**	**U**	**A**	**SA**
8.	The knowledge gained in this course will help me if I decide to take a more advanced computer science course.	**SD**	**D**	**U**	**A**	**SA**
9.	Classrooms discussions focused on computer science topics.	**SD**	**D**	**U**	**A**	**SA**

Scoring Procedures for the Computer Science Course Evaluation

	SD	D	U	A	SA
Add up the total number of items circled for Items 2, 4, and 9					
Multiply by:	×1	×2	×3	×4	×5
=					
Add up the previous row			÷ 3		
Engagement Score =					

	SD	D	U	A	SA
Add up the total number of items circled for Items 1, 6, and 8					
Multiply by:	×1	×2	×3	×4	×5
=					
Add up the previous row			÷ 3		
Rigor Score =					

	SD	D	U	A	SA
Add up the total number of items circled for Items 3, 5, and 7					
	×1	×2	×3	×4	×5
=					
Add up the previous row			÷ 3		
Clarity Score =					

Add up Engagement, Rigor, and Clarity Score		÷ 3
Total Score =		

https://inspire.redlands.edu/jcsi/vol1/iss1/2
DOI: 10.26716/jcsi.2018.01.1.2

Appendix B

Computational Thinking Survey

Please respond to all items in this survey by placing a check in the circle that represents your best answer. Ignore the letters in parentheses below for now. These will be used for scoring later.

1. I can:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
use computers to communicate my ideas. (cr)	○	○	○	○
easily describe how computers have helped society. (ae)	○	○	○	○
design solutions to a stated problem. (di)	○	○	○	○
easily evaluate the work of others. (ac)	○	○	○	○

2. I am:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
able to easily locate computational errors. (ac)	○	○	○	○
creative when trying to solve complex problems. (di)	○	○	○	○
able to easily communicate results using oral communication. (cr)	○	○	○	○
confident in my ability to select the best computing application to solve a problem. (di)	○	○	○	○
willing to compromise when there is no one right answer. (cp)	○	○	○	○

https://inspire.redlands.edu/jcsi/vol1/iss1/2
DOI: 10.26716/jcsi.2018.01.1.2

3. I:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
understand data structures. (da)	○	○	○	○
collaborate well in small groups. (cp)	○	○	○	○
know how to develop models to generate new questions. (da)	○	○	○	○
usually develop creative solutions when solving problems. (di)	○	○	○	○

4. I am:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
able to develop models to answer questions. (da)	○	○	○	○
able to easily communicate results through writing. (cr)	○	○	○	○
familiar with applying effective teamwork practices. (cp)	○	○	○	○
able to summarize the performance of software applications. (cr)	○	○	○	○
able to use algorithms. (di)	○	○	○	○

5. I can:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
develop models to answer questions. (da)	○	○	○	○
easily describe major themes and ideas (cr)	○	○	○	○
recognize the potential impact of a computer application. (ae)	○	○	○	○

6. I believe:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
people will use a computing application more if it helps to solve a problem. (ae)	○	○	○	○
that I can evaluate computing applications that others have created. (ac)	○	○	○	○
that I can reach a decision in groups when I disagree with my peers. (cp)	○	○	○	○

https://inspire.redlands.edu/jcsi/vol1/iss1/2
DOI: 10.26716/jcsi.2018.01.1.2

7. I feel confident:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly Agree (4)
in my ability to identify strengths and weaknesses of various solutions. (ac)	○	○	○	○
in my ability to easily explain results using computer output. (cp)	○	○	○	○

8. I understand:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Agree (4)
privacy and security concerns for computing applications. (ae)	○	○	○	○
how to evaluate computing applications that I have created. (ac)	○	○	○	○
different levels of abstraction. (da)	○	○	○	○
why people pick some applications over others. (ae)	○	○	○	○

End of Survey: Please refer to Scoring Guide for additional instructions.

Scoring Guide for the Computational Thinking Survey (CTS)

Directions: Record the number that corresponds to your response for each item in the boxes below. All boxes should contain a number.

Example:

	Strongly Disagree (1)	Disagree (2)	Agree (3)	Strongly agree (4)
use computers to communicate my ideas. (cr)	○	○	✓	○
easily describe how computers have helped society. (ae)	✓	○	○	○

Enter your responses in the boxes below.						**Total**
ae	1					
di						
da						
ac						
cr	3					
cp						

Enter your responses in the boxes below						Total
ae						
di						
da						
ac						
cr						
cp						

ae - **Analyze** the **effects** of developments in computing.
di - **Design** and **implement** creative solutions and artifacts.
da - **Design** and **apply** abstractions and models.
ac - **Analyze** your own **computational** work and the work of others.
cr - **Communicate** thought processes and **results**.
cp - **Collaborate** with **peers** on computing activities.

Your highest two scores represent your strongest areas. You second two highest scores represent your second highest categories. The lowest two scores represent areas in which you may need the most development.

The CTS was developed, with support from the National Science Foundation (Award Number: 1240822), at Howard University by Kenneth Alonzo Anderson, Legand L. Burge III, Troy J. Shine, Marlon Mejias, and Ketly Jean-Pierre. Paper copies may be freely duplicated in its original form. Please contact Dr. Kenneth Anderson at kenneth.anderson@howard.edu for assistance with electronic administration of this survey.

Computational Thinking Survey (CTS)

Overview

What is Computational Thinking?

Computational thinking is a universal 21st century skill that dates back to beginning of time. Computational thinking can be used in almost any field of study. At the most basic level, computational thinking involves the mental processes used to solve problems.

Computational thinking is a key practice of computer science and many other disciplines. Computational thinking is for everyone and involves reasoning, communication, considering multiple perspectives, identifying patterns, and much more[1].

How is this the COMPUTATIONAL THINKING SURVEY useful to you?

The COMPUTATIONAL THINKING SURVEY results identify your strengths, interests, and areas that may need more development across six (6) computational thinking practices[2&3].

1. Analyze the effects of developments in computing.
2. Design and implement creative solutions and artifacts.
3. Design and apply abstractions and models.
4. Analyze your own computational work and the work of others.
5. Communicate thought processes and results.
6. Collaborate with peers on computing activities.

Building on your strengths and developing your areas for improvement will help you to become more effective in solving problems, dealing with abstractions, computational tasks, and more.

[1] Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35. Retrieved from https://cacm.acm.org/magazines/2006/3/5977-computational-thinking.

[2] Arpaci-Dusseau, A., Astrachan, O., Barnett, D., Bauer, M., Carrell, M., Dovi, R., et al. (2013). Computer science principles: Analysis of a proposed advanced placement course. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (pp. 251–256). doi:10.1145/2445196.2445273

[3] Snow, E., Tate, C., Rutstein, D., & Bienkowski, M. (2017). *Assessment Design Patterns for Computational Thinking Practices in Exploring Computer Science*. Retrieved from https://pact.sri.com/downloads/AssessmentDesignPatternsforComputationalThinking%20PracticesinECS.pdf.