Infusing Computational Thinking across Disciplines: Reflections & Lessons Learned

Lori Pollock, Chrystalla Mouza, Kevin R. Guidry, Kathleen Pusecker

University of Delaware
Newark, DE 19716, USA
{pollock, cmouza, krguidry, klp}@udel.edu

ABSTRACT

In this work, we describe our effort to develop, pilot, and evaluate a model for infusing computational thinking into undergraduate curricula across a variety of disciplines using multiple methods that previously have been individually tried and tested, including: (1) multiple pathways of computational thinking, (2) faculty professional development, (3) undergraduate peer mentors, and (4) formative assessment. We present pilot instantiations of computational thinking integration in three different disciplines including sociology, mathematics and music. We also present our professional development approach, which is based on faculty support rather than a co-teaching model. Further, we discuss formative assessment during the pilot implementation, including data focusing on undergraduate students' understanding and dispositions towards computational thinking. Finally, we reflect on what worked, what did not work and why, and identify lessons learned. Our work is relevant to higher education institutions across the nation interested in preparing students who can utilize computational principles to address discipline-specific problems.

Keywords

Undergraduate CS education; computational thinking; CS principles

ACM Reference format:
YoLori Pollock, Chrystalla Mouza, Kevin R. Guidry, and Kathleen Pusecker. 2019. Infusing Computational Thinking Across Disciplines: Reflections & Lessons Learned. In *Proceedings of 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19), February 27-Mar. 2, 2019, Minneapolis, MN, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3287324.3287469

1. INTRODUCTION

As a result of policy efforts seeking to improve science, technology, engineering and mathematics (STEM) learning as well as industry initiatives aimed at promoting a more technology-savvy workforce, the need to help all students acquire computational thinking (CT) skills has gained increased attention.

CT is a problem-solving methodology implemented with a computer that can be automated, transferred and applied across subjects [1]. Jeanette Wing [2] suggested that CT is a fundamental skill of analytical thinking for everyone, which influences all aspects of our lives, from protecting our personal privacy and understanding our legal rights to deciding how to efficiently find and analyze information. A report on CT by the National Research Council (NRC) [3] advanced a similar idea indicating that CT is a cognitive skill that the "average person is expected to possess" (p.13).

CT includes a broad range of mental concepts and tools from computer science that help students analyze and develop solutions to problems within their own disciplines, including: problem decomposition, abstraction, algorithms, data representation and analysis, and automation [1, 3]. For example, business students can use CT to collect, classify, and leverage social media data for marketing and sales (data representation and analysis). Exercise science and physical therapy students can use CT to collect, classify, and learn from fitness and health records data for customized health regimes for patients (data representation and analysis). Journalism students can use CT to decompose articles and identify patterns, determine missing information to design interviews and surveys, develop sound logical arguments in their stories, and produce videos (problem decomposition, abstraction, automation). Music students can use CT to create and manipulate music through various technologies using concepts from programming (problem decomposition, abstraction, data representation, automation). Engineering students can analyze traffic patterns and algorithms to control traffic lights (problem decompositions, algorithms, and data analysis). Finally, art students can use CT to transform biochemical data into geometric images with meaning and beauty (abstraction and algorithms).

Although the ability to think computationally is essential to productive disciplinary engagement, most university curricula are currently not designed to provide such knowledge to a broad student population [4]. Even when efforts are made to teach every student about computing as a general education requirement, the discussion of CT does not always reach out to the entire university community [3]. Through support from the National Science Foundation (NSF), institutions such as DePaul University [5], Union College [6], and Towson University [4] have pioneered ways of integrating CT into existing, discipline-specific undergraduate courses in the sciences, humanities, and social sciences by creating flavors of introductory computer science courses for broad audiences. Recognizing the need for these interventions to be contextual so that students appreciate and use CT in their own career paths, these institutions advanced frameworks and models for promoting the development of CT among a diverse body of undergraduate students. Previous work, however, did not provide recommendations for establishing such initiatives university-wide to reach all students within a variety of majors and disciplines. For instance, some efforts focused on honors students [4], others only on students in the humanities [5], and others only on students within STEM fields [7].

SIGCSE'19, February 27-March 2, 2019, Minneapolis, MN, USA.

ACM ISBN 978-1-4503-5890-3/19/02...$15.00.
DOI: http://dx.doi.org/10.1145/3287324.3287469

In this work, we present our effort to develop, pilot, and evaluate a model for infusing CT into undergraduate curricula across a variety of disciplines using multiple methods that have been previously tried and tested individually, including: (1) multiple pathways of CT, (2) faculty professional development (PD), (3) undergraduate peer mentors, and (4) formative assessment [4, 6, 8]. Locally, this work is motivated by the recent decision of our institution to add CT to its education objectives for all undergraduate students. Nationally, this project is motivated by the need to identify models and practices that help institutions across the nation infuse CT across undergraduate curricula in a variety of disciplines, to ensure that all students acquire computational competencies required to maintain the economic competitiveness of the U.S. in a 21st-century workforce [3].

2. MULTIPLE PATHWAYS TO CT

Creating multiple pathways to the development of CT knowledge and skills is not new. For example, Carroll University [9] replaced general education mathematics and computer science requirements with new courses focused specifically on CT for all Bachelor of Science students at the University (80% of the population). It also developed interdisciplinary programs in computational science and trained faculty to teach new courses as well as integrate CT throughout individual curricula. While this effort targeted developing distinct CT courses, other prior NSF-supported efforts focused explicitly on the integration of CT across a variety of disciplines. For instance, one effort addressed infusing CT in liberal arts and journalism by having students from diverse backgrounds work in groups to find technology solutions to real world problems [10]. Similarly, another effort brought together faculty leaders inside the CT community to collaborate with faculty and students from outside the CT community [6]. The goal was to attract non-traditional students to take introductory computer science courses as well as encourage faculty outside computer science to develop discipline-specific courses that built on the introductory computational course. DePaul University, collaborated with other Chicago area institutions to work to incorporate CT across liberal studies [5]. Finally, Towson University, developed an introductory course for freshmen called *Everyday CT* as well as offered a discipline-specific CT seminar [4].

Collectively, these efforts provided a set of curricular examples and syllabi that illustrate ways in which CT can be infused into a variety of disciplines. Most of these efforts, however, were primarily "bottom up" and did not lead to sustainable change. Our approach differs in that our institution has already mandated that CT be incorporated across undergraduate curricula across disciplines as a General Education objective for all students. Therefore, we are tasked with the objective of identifying implementation strategies that respond to this new mandate. In this work, we present *pilot instantiations* of CT integration in three different disciplines: sociology, mathematics, and music. We also present our faculty support model and discuss formative assessment during the pilot implementations. There are two unique elements in our approach compared to prior efforts. First, rather than developing new courses, our faculty integrated CT into existing courses in their discipline. Second, none of the faculty had a background in computer science. Rather, they were disciplinary experts with interests in identifying strategies for CT integration that were relevant to courses in their own field.

3. FACULTY PROFESSIONAL DEVELOPMENT

PD is key to facilitating ongoing course preparation and implementation particularly for faculty with minimal CT background. One-shot workshops, however, are rarely effective [11]. Rather, effective PD that leads to improved student outcomes is ongoing and is connected to content and specific teacher needs [11]. Earlier faculty PD models on CT consisted of short-term workshops [4] or co-teaching approaches where CS faculty were paired with faculty in a specific discipline to ensure accurate content instruction. While there are a number of benefits to co-teaching, it makes scaling up challenging.

Our PD model consists of participation in a multiday institute during the summer and bi-weekly meetings throughout the academic year with the faculty teaching the course(s) and a CT integration team. The team includes a faculty from computer science who brings CT-related expertise, a learning scientist, and professionals from our university's Center for Teaching & Assessment of Learning. The institute provides an introduction to CT and CT classroom activity ideas while the CT integration teams and undergraduate students with computer science background provide assistance and resources during the planning and implementation of the courses.

3.1. Summer Institute

Every year at the end of the spring semester, our university provides a Faculty Institute that focuses on teaching, learning, and assessment. To infuse CT across undergraduate curricula for all majors, the Center for Teaching & Assessment of Learning at our institution in concert with faculty in computer science and education established a CT-related PD track within the Summer Institute. This track focused specifically on helping faculty learn about CT, including approaches and strategies for integrating CT in their specific disciplines. The Summer Institute provided hands-on opportunities for faculty to learn more about CT and how to create classroom activities and assessments that promote CT-related skills among students. We began by introducing CT with a "CS Unplugged" activity - an activity that introduces a CT concept without technology. We then shared our draft rubric focusing on assessing artifacts that integrate CT with disciplinary content. The rubric, modeled after the Association American of Colleges and Universities VALUE rubrics, draws heavily on the literature on CT and has been vetted by CT experts (see Appendix A). Faculty participants used the rubric as a scaffold for developing CT related activities within their discipline. Faculty with whom we have worked to infuse CT into their courses also shared specific examples and activities from their courses. In addition to these sessions at the summer institute, we also piloted with one faculty an alternative PD model that consisted of a 3-day intensive workshop focused on course design rooted in backward design principles. Both of these models are adaptations of existing faculty support programs at our institution and models that are commonly used at many U.S. colleges and universities.

3.2. Year-Long Support

To ensure that participating faculty were supported in their efforts to integrate CT in their respective disciplines, we provided year-long support in bi-weekly meetings that included both time for collaborative discussions and sharing of practices as well as one-to-one consultation. As noted, the team meetings included the disciplinary expert professors, a learning scientist, a computer

science professor, and two professionals from the Center for Teaching Assessment & Learning. Our discussion topics focused on identifying opportunities for CT integration into existing course modules, developing new modules that integrate CT in the specific discipline, providing feedback on module development, as well as developing instruments (e.g., surveys and rubrics) for assessing student CT knowledge and skills. We used the criteria in the draft rubric to guide many of our discussions and keep them focused on specific CT constructs that may be more fruitful in each discipline.

3.3. Undergraduate CT Fellows

Our work is uniquely characterized by the role of undergraduate peer mentors, called CT Fellows, who are responsible for ongoing faculty support. Specifically, CT-knowledgeable undergraduates not only assist participating faculty in the design and implementation of instructional activities that integrate CT with specific disciplinary content, but they also serve as peer mentors during the course. Our institution has a rich tradition of utilizing peer mentors in a variety of disciplines and contexts. This model is effective because the mentors are close enough to their peers to recognize confusion and perspectives that they themselves experienced while acquiring new knowledge. This "near peer" closeness helps mentors build rapport with students, encourage and provide emotional support, and set expectations with nondirective enabling [12].

To prepare CT Fellows, we draw on a model developed to prepare undergraduate students to assist K-12 teachers in teaching computing [13]. In this model, students interested in becoming Fellows participate in a university course called Field Experiences in Teaching Computing. The course, which is open to students who have completed at least one prior computer science course, is based on prior NSF-supported service-learning models and fulfills requirements for a computer science major, computer science minor, and discovery learning experience (a graduation requirement for undergraduate students at our institution). It has been offered continuously since spring 2013. Participants meet for 75-minutes on-campus once a week to identify and implement CT related teaching resources targeted toward specific disciplines, model classroom lessons that build on research-driven strategies, discuss teaching pedagogy, prepare and analyze lesson plans, and reflect on their experiences.

4. DESCRIPTION OF CT-INFUSED COURSES

In this section, we describe efforts to integrate CT across disciplines including sociology, mathematics, and music.

4.1. Sociology

In sociology, a faculty member infused CT into the *Introduction to Sociology* course for honors students in their freshman year. In this course, the instructor met with the CT Fellow and decided to focus primarily on the use of algorithms in relation to gender in children's products. Specifically, working in small groups, students researched and selected websites aimed at promoting and selling toys to children. Their task was to identify patterns such as slogans, colors, music, and activities associated with gender. Subsequently, their task was to develop algorithms, which is a key component of our rubric (see Appendix A), using terms such as AND, OR and NOT that helped a search engine identify gender on children's websites by specifying the salient patterns found in their analysis. Students were free to create an algorithm in a variety of formats (that did not include programming). For instance, some students provided lists of items while others wrote programs using pseudo-code.

Using the construct of algorithms in this context provided a means for gender socialization. Participants learned that websites selling toys to children could easily be analyzed using an algorithm and that these algorithms represent society's understanding of socially salient elements that help identify and recognize gender. In the process, students had an opportunity to recognize how rigid gender differences must be for an algorithm to identify gender. Thus, in this course, algorithms highlighted the process of socializing children into differences that are meaningful and how to identify and embody those differences. In essence, this assignment helped highlight the connections among CT and disciplinary content related to gender socialization.

4.2. Mathematics

In mathematics, CT was integrated into a course for non-majors titled *Contemporary Mathematics*. Specifically, the instructor consulted with the CT fellow and the PIs and integrated CT into an exercise related to data analysis which is a key component of CT and our rubric (see Appendix A). For example, the instructor asked students to identify a dataset from an online portal (https://www.statcrunch.com/). Subsequently, students were asked to identify a question that could be answered based on the dataset, analyze the data, display the findings using graphs, and write an interpretation of the findings.

Working in groups of 2-3, participants selected a variety of datasets from cereal (e.g., where cereal is stored in a supermarket, nutritional information, etc.) to social media to sports (e.g., NFL statistics) to music to presidential ratings. In the process, students learned about the data they were using and realized the value of using automation in the analysis as the data sets were too large to identify patterns manually. Further, students realized that meaningful representation of data was frequently dependent on the size of the dataset, which in turn influenced their choice of graphs and their interpretation of the findings.

4.3. Music

In music, a faculty member developed and taught a course called *Computational Thinking in Music*. This was the only instance where a new course was created as opposed to integrating CT into existing materials. This course was developed in collaboration with a music faculty who had previously integrated CT into a music appreciation course for non-majors. Although the course was successful, both faculty wanted to take a deeper dive into the connections among music and CT and thus established a pilot course focusing explicitly at the intersection of CT and music. Specifically, a key component of the course was the integration of programming in music and the use of online data (e.g., HookTheory.com) and Python code to explore music released by a popular artist and algorithmically write a portion of a song in the style of that artist.

The course was open to all students independent of prior background in music. As a result, there was variability in students' understanding of music from those with no background at all to amateur musicians. The first four weeks of the course were devoted to helping students fill in the gaps in terms of their understanding of music vocabulary and terminology. In Week 5, students downloaded pre-existing transcriptions of artists' music to form a corpus. Subsequently, they created harmonic

progressions for a verse (chords and rhythms) based on a computer-based analysis of corpus data; this analysis was performed using Python programs developed by the CT Fellow assigned to support this course. Following the development of the progression, participants created melody for verse (pitches and rhythms) again using Python code. The basic string of pitches and layout for segmentations was suggested by computer analysis of their specific corpus that created a Markov model. Finally, participants repeated the harmonic progression and melody tasks for the chorus and revised as needed. To conclude the course, the instructor brought in a band of musicians who gave a live performance of students' work.

5. METHODS

In this paper, we report on formative evaluation of our approach to CT integration across the disciplines of sociology, mathematics, and music. Relatedly, we answer the question: *How does participation in a CT-infused discipline specific course influence undergraduate students' understanding and attitudes towards CT?*

Data were collected from all undergraduates enrolled in one of the pilot courses in sociology (N=22), mathematics (N=35), and music (N=10) through a tested pre-and-post course survey [14] that was slightly modified for our work. In addition to other questions, the survey included one open-ended question of interest to our work: *What do you think the term computational thinking means?* This question tested students' understanding of CT. The survey also used 20 Likert-type items (see Table 1) to assess student understanding and attitudes in four categories: *definition* (e.g., see items 1-4), *comfort* (see items 5-10), *interest* (see items 11-14), *career/future use* (see items 15-20). Likert-type responses were scored on a scale of 1- 4 where 1=*Strongly Disagree* and 4=*Strongly Agree*. Subsequently scores were entered into a spreadsheet and means were calculated. The pre and post open-ended survey response was analyzed qualitatively using the constant comparative method [15]. This approach helped identify common themes that cut across participants' responses.

6. RESULTS

Overall, results indicated positive improvements from the pre to the post administration of the survey in some categories but not others. Specifically, participants reported better understanding (items 1-4) and comfort (items 5-10) with CT. However, participants did not express greater interest in CT after their participation in the course (items 11-14). Further, there was some ambivalence regarding the importance and value of CT in participants' future careers, specifically within their discipline (items 15-20). The exception was item 20, where participants clearly indicated improvements in their understanding of tools that can be used in their discipline. It is important to note, however, that given the small sample size and the fact that not all students responded to each question we were not able to conduct a statistical analysis to test for the significance of the gain score from the pre to the post administration of the survey.

Table 1. Pre/Post Scores on Understanding and Attitudes Toward CT

	Sociology* (N=22)		Mathematics* (N=35)		Music* (N=10)	
	Pre	Post	Pre	Post	Pre	Post
1. CT focuses solely on understanding how computers work.	3.7	3.7	3.0	3.7	2.7	2.8
2. CT involves thinking logically to solve problems.	3.6	3.4	3.9	3.6	3.8	3.7
3. CT requires using computers to solve problems.	2.9	3.6	2.9	3.1	1.9	2.4
4. CT involves abstracting general principles and applying them to other situations.	3.3	3.4	3.5	3.5	3.6	3.9
5. It is possible to apply CT skills to solve problems in my major.	3.6	3.9	3.1	3.2	3.6	3.3
6. I am comfortable with CT concepts.	2.1	3.5	2.7	3.0	2.4	3.3
7. I can achieve good grades in courses that require CT.	3.1	3.7	3.2	3.1	2.8	3.4
8. I can learn to understand CT concepts.	3.7	3.6	3.6	3.5	3.4	3.8
9. I use CT skills in my daily life.	3.2	3.3	3.0	3.1	3.0	3.2
10. I can solve problems by using CT skills.	3.4	3.0	3.3	3.3	3.2	3.3
11. I think using CT skills is boring**.	1.7	1.7	2.4	2.2	2.1	2.1
12. The challenge of solving problems using CT appeals to me.	3.1	2.7	2.7	2.7	3.0	3.2
13. I am interested in using CT skills.	3.3	3.0	2.8	2.7	3.1	3.2
14. I will voluntarily take courses that use CT if I am given the opportunity.	3.2	3.2	2.3	2.4	3.0	3.1
15. Knowledge of CT skills will help me secure a better job.	3.4	3.7	3.1	3.1	3.6	3.4
16. My career goals require that I learn CT skills.	3.2	3.8	2.6	2.9	2.9	2.9
17. I expect that learning CT skills will help me to achieve my career goals.	3.4	3.4	2.8	2.9	3.3	3.3
18. I hope that my future career will require the use of CT skills.	2.9	3.0	2.6	2.5	2.8	3.2
19. Having background knowledge and understanding of CT skills is valuable in and of itself.	3.5	3.1	3.1	3.2	3.6	3.7
20. I know what CT tools are available to solve problems in my major.	2.2	3.1	2.7	2.9	2.6	3.3

* Not all participants responded to all items on the pre/post survey

** This item is a reverse statement and thus the lower score indicates disagreement that CT is boring.

Findings from the open-ended question also noted improvements in participants' understanding of CT related concepts before and after participation in a CT infused course. When asked to respond to the question *"What do you think the term CT means?"* initial responses focused primarily on the following:

- Logical thinking: "*Using logic and common sense when thinking", "Using logic and structured thinking to organize given information", "Using common sense and logic in daily activities."*
- Non-creative thinking: "*Thinking in a math-oriented kind of way, thinking in a strict way/not creatively."*

- Thinking like a computer: *"Thinking through tasks like an AI computer program to find the fastest and most efficient option"*.

Further some participants simply noted that CT is about "using a computer", "computer knowledge" or "completing mathematical calculations". Only a small number of students recognized CT as a problem-solving methodology that can be implemented with a computer. One participant, for instance, explained: *"CT is defining a problem and trying to find a solution to it using a computer or machine"*.

In contrast, post-survey data demonstrated an improved understanding of CT through more detailed and conveying responses. Most participants recognized CT as a problem-solving methodology, while others used specific CT concepts as they defined CT, including "decomposition" and "abstraction". One participant explained, *"CT is a thought process in which tasks are broken down into easier, more manageable steps. CT uses problem decomposition, algorithms, abstraction, and data."* Similarly, another student noted: *"CT involves considering a problem and breaking it down through decomposition in order to better carry out the task or solve the problem."* Further, participants recognized that CT is not a mechanistic approach to thinking but rather involves creativity and problem solving through multiple perspectives. One student explained, *"CT involves thinking abstractly and being able to decompose an issue at hand in order to see all possible solutions"* while another student noted that *"CT is being able to look at things or issues from multiple perspectives or lenses and come up with various solutions or answers"*. Finally, more participants understood the relationship between CT and computers. One student explained, *"CT is being able to think or put ideas in terms that a computer could follow"*. Overall, in the post-survey, a greater number of participants were able to articulate the relationship between CT, problem-solving, and computers.

7. DISCUSSION AND LESSONS LEARNED

In this section, we identify lessons learned from the pilot implementations in sociology, mathematics, and music.

- Faculty development is critical in learning about CT and honing an understanding of discipline-specific approaches to CT. Further, regular discussions between a CT integration team that includes both CT and disciplinary experts as well as experts on teaching and assessment are critical in developing course objectives and instructional activities that indeed provide a path to the design and implementation of CT into disciplinary courses without the need for co-teaching. The draft VALUE-style rubric that presents a succinct definition of CT with measurable criteria was tremendously helpful in not only facilitating discussions, but also in helping faculty modify their courses, creating assignments, and assessing students' work.
- Peer mentors in the form of CT knowledgeable undergraduates can support the faculty in a way that increases faculty confidence in the classroom as they enact CT activities. However, the role of CT Fellows varied among participants. For example, the music faculty required substantial technical support from a CT Fellow who wrote computer code and helped the faculty and students adapt and run that code on their computers. In sociology and mathematics, however, the CT Fellow served primarily as a brainstorming partner for the faculty members in developing and polishing course assignments that integrated CT with disciplinary content. As faculty gain confidence they are more likely to continue to build on their successes with additional CT-related activities and modules. The peer mentor for a given course may not require more than four hours a week including class time, thus the same CT Fellow may be able to support more than one faculty in a given year. This helps with scaling the model to support more faculty in CT integration each year. However, the logistics of coordinating, supervising, and supporting these students who also have many other responsibilities – their own coursework, enrollment in the CT fieldwork course, and attending some classes with their CT faculty member – can be formidable and cannot be underestimated. These challenges make it likely that we will make significant changes to this aspect of our support model as we continue infusing CT into additional courses at our institution.
- Students sometimes do not appreciate why they are learning CT within a disciplinary-specific course. This suggests that faculty need to begin the course and its advertising with strong motivation for integrating CT transparent learning objectives into the course, and reiterating the importance of those learning objectives throughout the course. This can be done by providing examples of how the CT-related activities are relevant to their discipline and the real world outside the classroom; in fact, situating CT within students' and faculty members' selected disciplines is why we have elected to infuse CT into many different courses instead of assigning this responsibility to computer science faculty or creating CT-specific courses co-taught by computer scientists.
- Formative assessment needs to account for the fact that artifacts created for assessment are different across disciplines. Thus, a common rubric for assessing CT knowledge and skills may need to be adapted when implemented on various kinds of student artifacts.

In conclusion, our findings indicate that a model for CT integration that applies multiple proven methods together holds promise for successfully meeting university goals of CT for all, without a co-teaching model or separate discipline-oriented computer science courses. In addition, the model provides service learning opportunities for CT knowledgeable undergraduates and creation of a network of faculty who can share their ideas, challenges, and experiences in integrating CT within their discipline. Our next challenge is scaling up the model to reach a greater number of faculty and students

Appendix 1. Draft Computational Thinking Rubric adapted from [1]. See also: http://www.udel.edu/005415

	Capstone	Milestones		Benchmark
	4	3	2	1
Decomposition *Breaks a problem into its constituent subproblems*	Creates a problem decomposition that breaks a complex problem into clearly described, well-defined, and distinct-but-related subproblems that are easier to solve than the original problem but when combined efficiently solves the original problem.	Creates a problem decomposition that breaks a complex problem into clearly described subproblems that are distinct-but-related but lack efficiency, although they solve the original problem.	Creates a problem decomposition that breaks a complex problem into subproblems that lack efficiency, fail to have sufficient descriptions, and overlap, although they solve the original problem.	Creates a problem decomposition that breaks a complex problem into subproblems that are inefficient, described poorly, overlap or closely related, and fail to completely solve the original problem.
Data *Analyze (or create) a data set that facilitates discovery of patterns and relationships*	Analyzes[1] a data set to ensure it is sufficiently comprehensive, efficiently organized, meaningfully labeled, and thoroughly described so that it can be analyzed to discover meaningful patterns and relationships.	Analyzes a data set to ensure it is sufficiently comprehensive, meaningfully labeled, and thoroughly described but fail to ensure that it is efficiently organized.	Analyzes a data set to ensure it is sufficiently comprehensive and meaningfully labeled but fail to ensure that it is thoroughly described and efficiently organized.	Analyzes a data set but fail to ensure that it is sufficiently comprehensive, efficiently organized, meaningfully labelled, and thoroughly described so patterns and relationships are obscured.
Algorithms *Creates a series of ordered steps to solve a problem or achieve a goal*	Creates an accurate, logical, efficient, and well-described sequence of steps or instructions to solve a problem or achieve a goal.	Creates accurate steps that are logical and well-described and solve a problem or achieve a goal but the steps are inefficient e.g., not in an optimal sequence, overlapping or duplicative.	Creates logical steps that solve a problem or achieve a goal but the steps are poorly described.	Creates a sequence of steps that do not solve a problem or achieve a goal. The steps lack efficiency, sufficient descriptions, and are not described or documented.
Abstraction *Reduces complexity to create a general representation of a process or group of objects so it is not only appropriate for the immediate purpose or goal but can also be used in different contexts*	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Selects essential characteristics by filtering out unnecessary information. Can be used to solve other problems or goals.	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Selects essential characteristics by filtering out unnecessary information. Cannot be used to solve other problems or goals.	Creates an accurate-but-simplified representation of a process or group of objects to solve the problem or meet the goal. Fails to select all essential characteristics by filtering out unnecessary information. Cannot be used to solve other problems or goals.	Creates an representation of a process or group of objects that is not accurate, not sufficiently simplified, or fails to solve the problem or meet the goal.

[1] Advanced courses or disciplines with a strong focus on CT may instead require students to create a data set instead of analyzing an existing one.

8. ACKNOWLEDGMENTS

This work is supported by a grant from the National Science Foundation (Award # 1611959). All opinions are the authors and do not necessarily represent those of the funding agency.

9. REFERENCES

[1] Barr, V., & Stephenson, C. 2011. Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, 2(1): 48-54.

[2] Wing, J. 2006. Computational thinking. *Communications of the ACM, 49*(3). 33-35.

[3] National Research Council. 2010. *Report of a Workshop on the Scope and Nature of Computational Thinking*. Washington, DC: National Academy of Sciences.

[4] Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., Kleinsasser, W., Dehlinger, J., & Kaza, S. 2011. A model for piloting pathways for computational thinking in a general education curriculum. In *Proceedings of the 42nd ACM Technical Symposium on Computer science education*, 257-262. doi: 10.1145/1953163.1953243.

[5] Perković, L., Settle, A., Hwang, S., & Jones, J. 2010. A framework for computational thinking across the curriculum. In *Proceedings of the 15th annual conference on Innovation and technology in computer science education*, 123-127. doi: 10.1145/1822090.1822126.

[6] Barr, V., Motahar, E., Liew, C. W., & Stewart-Gambling, H. 2009. Building a campus wide computation initiative: Panel discussion. *Journal of Computing Sciences in Colleges, 24*(3), 109-110.

[7] Alvarado, C., Dodds, Z., Kuenning, G., Hadas, R., & Shelton, C. 2010. *Modular CS1 from the inside out: Computational thinking for all STEM students*. CPATH PI Meeting, March 25-26, Arlington, VA.

[8] Pollock, L., Mouza, C., Atlas, J., & Harvey, T. 2015. Field experiences in teaching computer science: Course organization and reflections. *Proceedings of the 47th ACM technical symposium on Computer science education*, 374-379.

[9] Kuster, C., & Hu, C. 2012. CAPTH-1: Developing computational thinking skills across the Undergraduate curriculum. Retrieved from: http://grantome.com/grant/NSF/CNS-0939032.

[10] Pulimood, S. M., Pearson, K., & Bates, D. C. 2014. CABECT: collaborating across boundaries to engage undergraduates in computational thinking (Workshop – abstract only). In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, Atlanta, Georgia, USA, March 05 – 08, 2014.

[11] Zepeda, S.J. 2011. Professional development: What works (2nd ed.). Larchmont, NY: Eye on Education, Inc.

[12] Harrington, K., O'Neill, P., & Bakhshi, S. 2008. *Making time and space for writing: Student writing mentors and the writing centre*. Presentation at Writing Development in Higher Education Conference, University of Strathclyde, Glasgow, Scotland.

[13] Pollock, L., Mouza, C., Atlas, J., & Harvey, T. 2015. Field experience in teaching computer science: Course organization and reflections. In *Proceedings of Special Interest Group in Computer Science Education*, March 4-7, Kansas City, MO.

[14] Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. 2014. Computational thinking in elementary and secondary teacher education. *ACM Trans. Comput. Educ. 14*, 1, Article 5 (March 2014), 16 pages. DOI:http://dx.doi.org/10.1145/2576872

[15] Hatch, J. A.(2002. Doing qualitative research in education settings . New York, NY: Suny University Press.