Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective

As Published: https://doi.org/10.1007/s10956-019-09802-x

Publisher: Springer Netherlands

Persistent URL: https://hdl.handle.net/1721.1/131786

Version: Author's final manuscript: final author's manuscript post peer review, without publisher's formatting or copy editing

Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective

Cite this article as: Irene Lee, Joyce Malyn-Smith, Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective, *Journal of Science Education and Technology*, doi: 10.1007/s10956-019-09802-x

Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective

Irene Lee
Massachusetts Institute of Technology
ialee@mit.edu

Joyce Malyn-Smith
Education Development Center
jmsmith@edc.org

Acknowledgements
The authors would like to thank the participants in the two workshops on "Developing an Interdisciplinary Framework for Integrating Computational Thinking in K–12 Science, Mathematics, Technology, and Engineering Education" for their contributions of activities and discussion surrounding CT integration. This work would not have been possible without their willingness to share activities and insights. This project was supported by the National Science Foundation's STEM+C program, grant number 1647018. Any opinions, findings, conclusions, or recommendations reported here are those of the authors and do not necessarily reflect the views of the Foundation.

Compliance with Ethical Standards
Funding: The study was funded by the National Science Foundation (grant number 1647018).
Conflict of Interest: The authors individually declare that each has no conflict of interest.
Research Involving Human Participants: All procedures performed in studies involving human participants were in accordance with the ethical standards as articulated in the 1979 Belmont Report and as reviewed by the Institutional Review Board at Education Development Center, Inc. (Registration No. 00000865). This research was determined to be exempt by the IRB under Section 101(b), paragraph 1.

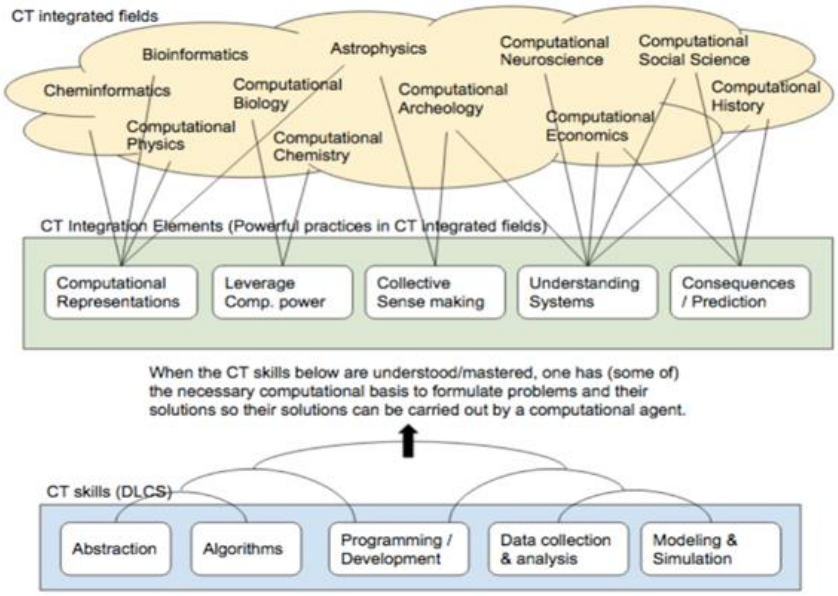

Figure 1. CTIEs as a bridge between traditional CT integration in K–12 education and CT as a powerful practice used in CT-integrated STEM fields.

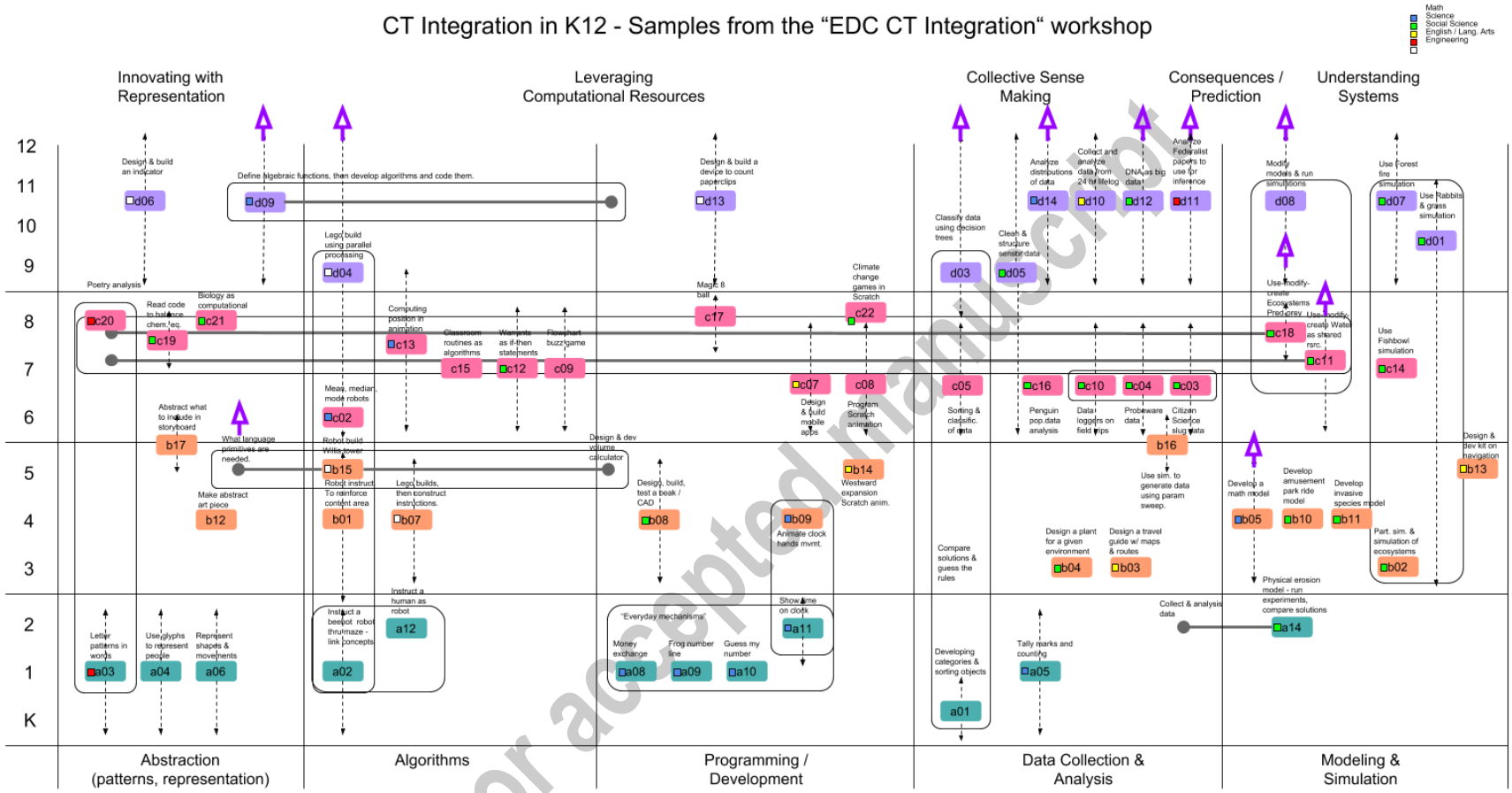

Figure 2. CT integration in K–12: Samples from the EDC CT Integration workshop

Computational Thinking Integration Patterns Along the Framework Defining Computational Thinking from a Disciplinary Perspective

Abstract

This paper describes analyses of the K–12 computational thinking (CT) integration activities collected at two NSF-funded workshops, "Developing an Interdisciplinary Framework for Integrating Computational Thinking in K–12 Science, Mathematics, Technology, and Engineering Education," held in August and November of 2017 at Education Development Center, Inc., in Waltham, Massachusetts. The workshops convened a working group of principal investigators, researchers, and educators from the National Science Foundation (NSF)-funded ITEST (Innovative Technology Experiences for Students and Teachers) and STEM+C (STEM+Computing) projects to draft an interdisciplinary Framework for Integrating CT into K–12 Education. The goal of this paper is to share that framework and our findings on promising learning progressions, gaps that exist in the collected set of activities, specific advances in STEM fields that were made possible through CT, and suggested ways that CT integration in K-12 can evolve to reach what the CT Integration Framework proposes as five "Computational Thinking Integration Elements." or "CTIEs". This framework is designed to help educators see ways to engage students in CT within disciplinary learning. The analyses and findings may benefit STEM and computing education fields by elucidating the target of CT as used within CT-integrated STEM fields.

Background

CT is widely described as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an

information-processing agent" (Wing, 2010, p. 1). Since noted computer scientist Jeannette Wing first proposed CT as a new "core skill" in 2006, various groups (e.g., Grover & Pea, 2013, 2018) have attempted to define CT for education and training purposes. CT was given prominence as a strand of the K–12 Standards for Computer Science developed by the Computer Science Teachers Association (CSTA) (2011), and the K–12 Computer Science Framework Steering Committee (2016) positioned CT at the heart of the computer science practices in the *K–12 Computer Science Framework*. CT, seen as the "connective tissue" between computer science and science (Martin, 2018), also has relevance in scientific practices; the *Next Generation Science Standards* (NGSS Lead States, 2013) include CT as one of eight scientific practice standards.

CT integration is the embedding of CT within fields in the service of STEM learning and innovation. Numerous examples exist of CT fundamentally changing STEM fields and producing new integrated fields, such as computational biology, computational chemistry, and computational social sciences. In these fields, once laborious tasks are now made easier when utilizing computers' processing power. More importantly, new techniques and tools have enabled scientists to ask new questions and expand the knowledge base in these fields. Many innovations in these rapidly changing fields are due to CT; what is being accomplished now would not have been possible without computational power, CT, and computational tools.

Too often in today's STEM classrooms, CT is taught in a manner disconnected from the STEM content it serves. CT is dissected into its component parts, and activities focus only on developing understanding of a single component of CT (such as algorithms, or data analysis) without connection between components. In this paper, we argue that CT in today's STEM classrooms needs to be more than an introduction to computer science. Instead of being taught

in a manner disconnected from disciplinary learning and understanding, rich opportunities exist to link CT to the practices of CT-enabled scientists and engineers.

Prior Work and Foundational Understandings

Three "Profiles" of computational thinking-enabled STEM professionals working in America's workplaces were developed and validated nationally between 2011 and 2014: the CT-Enabled STEM Professional/Research Scientist (Malyn-Smith & Ippolito, 2011), the CT-Enabled Product Engineer (Malyn-Smith & Ippolito, 2012), and the Big-Data-Enabled Specialist (Malyn-Smith & Ippolito, 2014). Each profile was collaboratively developed by a panel of professionals in these jobs who shared their first-hand knowledge of the CT tasks and functions they perform on a daily basis. The resulting profiles clearly defined the role and described the job functions (duties and tasks) of professionals in STEM fields whose work integrated and was dependent on CT. These profiles informed the Framework defining CT from a Disciplinary Perspective (Malyn-Smith et al., 2018).

A fourth profile, of the CT Integration Specialist (Malyn-Smith, Lee, & Ippolito), was developed in 2017. The panel who created this profile consisted of K-12 educators whose job it was to integrate CT in K-12 curriculum within school settings. The panel defined a *CT integration specialist* as an educator who recognizes that CT is integral to learning; integrates CT across academic disciplines and/or out-of-school activities by establishing an inclusive culture, modeling the use of CT, creating new CT activities or modifying existing ones, and assessing students' CT learning. Through this work and our connections to the panel members, our understanding of how CT is integrated into K–12 education and professional STEM fields was formed and has continued to develop through our subsequent work and as these fields evolve.

The Framework Defining Computational Thinking from a Disciplinary Perspective

To guide the teaching and learning of CT within the STEM disciplines, a new kind of CT framework was needed—one that captured and clarified what students were able to do in a STEM context using CT, and were unable to do without CT. The workshops on "Developing an Interdisciplinary Framework for Integrating Computational Thinking in K–12 Science, Mathematics, Technology, and Engineering Education" were our first step toward developing a CT integration framework.

The 54 workshop participants comprised a balance of researchers and practitioners. They represented four grade spans (K–2, 3–5, 6–8, and 9–12) and a number of disciplines, including science, mathematics, engineering, social science, computer science, and the humanities. In total there were 31 researchers, 18 teachers and practitioners, 3 participant observers, and 2 staff members. Thirteen participants were from colleges or universities, 15 from schools, 15 from nonprofits, 1 from business, and 3 from foundations, including NSF. Others were guests. The development of a framework for CT from a disciplinary perspective was guided by some of the foremost CT thought leaders in the United States, including Irene Lee of Massachusetts Institute of Technology, Shuchi Grover, Fred Martin of University of Massachusetts/Lowell and CSTA, and Michael Evans of North Carolina State University.

Educators, researchers and practitioners across K–12 grade levels shared curricular lessons and activities that illustrated CT in action in classrooms. Together, the group explored these examples and found that a number of common "CT integration elements" emerged that forged connections between these CT activities and the powerful practices used by professionals in CT-integrated STEM fields. Participants were then asked to provide additional examples of CT integration by grade level and discipline and how their lesson or activity aligned with CTIEs.

These examples were subsequently reviewed and discussed within the emerging framework of common CT integration elements.

Sixty-one descriptions of CT integration activities were collected and refined by the participants during the workshops. Each activity was described in a paragraph and then annotated by the submitter with (a) an identification of the CT involved in the activity, (b) how CT enhanced the disciplinary learning, and, if relevant, (3) how the disciplinary learning illuminated CT.

The resulting framework (Malyn-Smith et al., 2018) lists five CT Integration Elements (CTIEs) that span the professional uses of CT in STEM fields and can be integrated into K–12 STEM education:

- **Understanding complex systems.** Modeling how interactions of many individuals or components in a system lead to aggregate-level emergent patterns is difficult to do without CT, as complex systems are often hard to predict, due to having a multitude of interrelated factors and levels. Indeed, our understanding of complex adaptive systems prior to the advent of computational tools for modeling these systems was limited. Modeling the rules governing such systems, then simulating a system's change over time and gathering real-time feedback in the form of simulations help scientists understand the dynamics of complex systems. In K–12 education, computer modeling and simulation of complex systems offers students a way to see test their hypotheses about the generators of pheonomena, and visualize how systems behave under different circumstances and with different inputs (Lee & Martin, 2016). Students are able to pose "what if?" questions and conduct scientific experiments to seek answers.
- **Innovating with computational representations.** The design and development of innovations is made possible through CT. New ideas, conceptualizations,

representations, and processes can be thought of and developed as computations. For example, thinking of the brain as a signal processor and creating neural networks as artificial brains has led to advances in artificial intelligence and cognitive science. In K–12 settings, educators can encourage students to think of processes as computations that can be combined and/or reordered to produce different outcomes.

- **Designing solutions that leverage computational power and resources.** Scientists working with large data sets or on computationally intensive calculations design solutions that leverage the efficient use of resources and computational power to optimize their time. In some cases, distal collaborators can pool and share computational resources; in other cases, co-located collaborators can access distributed resources to achieve their goal. Some speedups are achieved by decomposing data sets and/or processes to run in parallel. In K–12 settings, educators can challenge students to think about how they would solve a problem differently if the input set was larger in size. For example, rather than sorting 10 items, how would students go about sorting 10,000 items?
- **Engaging in collective sense-making around data.** Data sets can be amassed through crowd-sourcing or collection by multiple individuals or sensors. These data can then be analyzed to uncover patterns. Visualization of multi-dimensional data enables students to see patterns that might not otherwise be apparent. In K–12 settings, teachers can ask small groups of students to run simulations on a subset of the inputs, then share their output data and analyses. Gathering and analyzing the combined data illustrates how each part of the data set contributes to an understanding of the whole.
- **Understanding potential consequences of actions.** Scientists envision the future through simulation and use machine learning to make predictions. Using parameter sweeping, the space of all possible combinations of inputs can be tested to see the

variety and probability of outcomes. In K–12 classrooms, students can learn how cause-and-effect relationships can be used to predict outcomes. Students can also begin to understand the space of inputs created by parameterizing models.

These elements are foundational to gaining new understanding and developing processes that enabled innovation and scientific discovery in these new fields.

Notably, the CTIEs go beyond the mechanics of learning to program a computer. They form a bridge between CT as it has traditionally integrated into K–12 classrooms and professional practices in CT-integrated STEM fields.

[INSERT FIGURE 1]

CT Learning Progressions Aligned to the CTIEs

To better understand if and how existing CT integration activities in grades K–12 prepared students for the work described in the framework, we conducted three analyses of these activities. In the first analysis, to identify promising CT integrations and any existing gaps, the activities were sorted and arranged in a matrix, with grade band along the vertical axis and the primary CT component addressed (using the five components of the Massachusetts *Digital Literacy and Computer Science* [MA DLCS] *Standards* ([Massachusetts Department of Elementary and Secondary Education (MA DESE), 2019]) along the horizontal axis.

Further annotation (horizontal bars) was added to indicate when an activity spanned several CT areas, and vertical lozenges were used to indicate potential progressions comprising multiple activities. These progressions were not inherent in the activity descriptions but were drawn after

the activities were arranged to suggest potential linkages between activities. This landscape view (see Figure 2) was intended to expose gaps and potential links between activities that could form the basis for progressions in the sample of activities and potential progressions built out of existing activities.

[INSERT FIGURE 2.]

A second analysis aimed to assess if, when, and how various activities reached the five CTIEs. Each activity was also assessed for its coherence with the five CTIEs.

Activities were arranged by grade level within each DLCS component: abstraction, algorithms, programming and software development, data collection and analysis, and modeling and simulation. Our findings, arranged by the primary MA DLCS CT component addressed, are described below.

Abstraction

The MA DLCS defines *abstraction* as “a process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the problem” (MA DESE, 2019, p. 16). As the grade level increased in K–12 settings, abstraction progressed from “What is an abstraction?” to identifying and using abstractions and then creating abstractions.

In grades K–2, students learned about abstraction by identifying and matching patterns. Activities included finding lettter patterns in words and movement patterns in dances.

Abstraction was also taught through representing fellow students by their attributes, such as eye color, hairstyle, and height. These representations were then connected to data studies by having students count and analyze the number of classmates who had each characteristic. In grades 3–5, abstraction was used to answer the question "What key features do I need to include?" in a representation of a story, artistic creation, or building, presaging the use of abstraction in developing computer models. In middle school, examples of how abstraction was taught were somewhat limited. Activities ranged from identifying patterns in poetry to representing probablistic outcomes of reproduction via Punnett squares. Seventh and eighth graders were engaged in involved moving through a series of representations stressing the utility of models at varying levels of abstraction. Students in grades 9–12 combined engineering design and representation in which they were tasked with drawing a circuit diagram for the indicator. Students traded their diagrams with another group and then had to recreate the indicator using only the diagram.

Abstraction extends up to **innovating with computational representations**. Three activities in this grouping lend themselves to moving from abstraction to innovating with representations as seen in the practices of STEM professionals:

- The Willis Tower activity required students to decide on and then develop the language primitives or basic set of commands that a robot would need to construct the Willis Tower as a set of layers. This provided students with the opportunity to create and refine a reusable set of moves that can be used to encode a building.
- Creation of decision trees provided high school students an opportunity to develop innovative representations as they developed classification schemes for objects based on key characteristics. The path taken to the item from the root to a leaf node could then become the shorthand representation of the item.

- The creation of an algebraic function as an object to represent a process, pattern, or phenomenon is key to the work of STEM professionals.

Gap analysis: Only a quarter of the examples of abstraction provided K-12 students with the opportunity to gemerate new representations rather than simply using provided representations.

Based on this finding, we make the following suggestions on moving from abstraction to innovating with representations in classroom activities:

- Link abstraction explicitly to condensed representations that a computer could use to process operations on a set. For example, human visual systems are capable of differentiating between 6 million colors. In computer systems, color is commonly represented by bytes in RGB (red-green-blue) that enables a compact digital representation of those colors.
- Focus attention on developing and assessing representations that can be encoded digitally. Compare and contrast alternate representations when possible.
- Highlight abstraction in the context of modeling and simulation by identifying which entities were chosen to be included in a model and which were left out.

Algorithms

In K–2, algorithms were introduced as a sequence of instructions for completing a task. Initially, single instructions are given to update from the current state (for example, position in a maze) to the next state. As students' age and sophistication advance, longer sequences of instructions were developed before "running and checking" the algorithm for correctness. Across grades K–6, the addition of algorithmically driven robot movement was used to reinforce content learning. For example, on a mat made up of alphabet tiles, the robot could be instructed to follow a path that connects life stages of a butterfly in order. In these activities, the robots were extraneous to the primary content area learning objectives but may have reinforced a progression or process.

In grades 3–5, students were tasked with the decomposition of objects, either LEGO brick constructions or famous buildings, prior to developing algorithms to build the target structure. In middle school, the concept of an algorithm as a set of instructions to perform a task was reinforced by eliciting algorithms for everyday classroom routines. Students also developed and used algorithms to construct animations. Students in a math class used interpolation to position characters at intermediary positions between the starting and ending animation frames. Scientific reasoning and argumentation from evidence were reinforced by encoding warrants as if-then statements, such as "If ___ happens, then I know that ____" or "If ___ happens, then I know that my hypothesis was correct." Program logic, internal to an algorithm, was introduced to middle school students who created flowcharts of a mathematical game for learning about divisors and multiples. In a high school activity, the issue of scale (or processing at scale) was addressed in an engineering activity in which students developed parallel processing algorithms to mass-produce a LEGO brick construction. Students conducted experiments with different pipelining and parallelization schemes to determine which would be optimal under certain constraints.

Gap Analysis: In terms of moving from algorithms to **designing solutions that leverage computational power and resources**, only one activity (at the high school level) moved students toward thinking of algorithms for more than one processor. This type of algorithmic thinking and optimization is critical in the work of STEM professionals who use vast computing resources.

Based on this finding, we make the following suggestions on moving from algorithms to leveraging computational power and resources in classroom activities:

- Emphasize that the benefit of computational power is the processing speed and capability. With this greater speed and capability, one can test multiple solutions quickly, search parameter spaces to find best solutions, and process large quantities of data to inform decision making.
- Provide real-world examples from CT-integrated fields, such as modeling of weather systems where the increased capabilities enable longer term forecasts. Additionally multiple models can be executed in parallel to produce landscapes of potential outcomes.
- Minimize engaging students in creating algorithms that simply reinforce content and have no alignment with the use of CT in integrated fields.
- Provide age-appropriate versions of professional high performance computing practices such as task decomposition or domain decomposition. One way to accomplish this is to have students play the role of processors and develop strategies to carry out processes in parallel.

Programming and Software Development

In grades K–2, programming and software development began with block-based languages in which students linked blocks to create a program. Several examples were situated in everyday math activities that students can automate such as money exchanging and number line traversal. In grades 3–5, programming activities included making Scratch animations to reinforce content learning, programming a mechanism in a Makerspace context, and programming clock hands in a math context. At the middle school level, programming to create Scratch animations and/or games was common theme, but the degree to which these activities included abstraction varied. At the high school level, an activity featured designing and building an automated device to count paperclips.

Gap Analysis: The activities presented engaged students in using arithmetic operators, conditional, and looping, and creating, testing, and modifying a simple program. They did not extend students' work to selecting a "best" algorithm based on a given criteria or using functions to hide details. No evidence was found of students being taught to use a software development process that leads to a minimal viable product, selecting appropriate data structures, and analyzing tradeoffs among multiple approaches.

Both algorithms and programming lead up to the ability to **design solutions that leverage computational power and resources** at scale. There is a need for students to think about and develop automations beyond a single computer as the processor and to consider sharing and allocating resources to complete a large task—all of which are important precursors to the professional tasks of using the cloud, networked resources, and parallel processors. The issue of scale (processing at scale) is not often addressed in K–12, though it is critical in the work of STEM professionals who use computing resources.

Based on the findings, we make the following suggestions on moving from programming and software development to leveraging computational power and resources in classroom activities:

- Extend students work in programming to finding or selecting the "best" algorithm based on a given critera.
- Introduce functions as a way to encapsulate task specific solutions and hide detail in programs.
- Consider solutions that leverage more than one processor.

Data Collection and Analysis

In grades K–2, data collection and analysis took the form of categorizing objects and sorting them into groups, using the pre-defined categories. In some cases, students also needed to be able to explain the reasoning for the groupings they created. In a math context, tally marks were used to track collected items in categories; the tally marks were then counted to produce a total for a given category. In grades 3–5, various project based learning activities necessitated the collection and visualization of data. In these projects, developing solutions to the open ended problems often entailed collecting, monitoring, and analyzing data. An activity in grades 5–6 focused on multiple sets of experimental data to be generated, collected, and analyzed, using simulations. For a given research question, students worked in small groups to design one run of the experiment, identifying the experiment setting, steps, and data to be gathered. As a class, students reviewed and refined the design and agreed on the experiment settings for which each group had responsibility. This method of dividing a multiple-dimension parameter sweep and assigning different simulations to different groups is a common way to mimic multiprocessor simulation experiments. Later the teacher gave a copy of the output data as a spreadsheet to students for additional analyses that included reorganization, formatting, and sorting to reveal patterns. Formulas and charts were also introduced. These practices mimic the professional practices of crowdsourcing and **engaging in collective sense-making around data**.

Students in grades 6–8 collected real-world data in a variety of ways. Data collection methods included using data loggers on field trips; using probeware to collect data while running classroom experiments; and conducting snail and slug counts as part of a citizen science project. Additionally, students used readily available data sets found online, such as penguin population data from Antarctica. These data were subsequently cleaned, organized, and analyzed. In citizen science projects, collecting local data supported discussion about both the advantages of this type of data (e.g., having many people collect data allows you to get data

from a larger area) and the disadvantages (e.g., an absence of data from a particular location could mean that no one made observations at that location).

At the high school level, data collection and analysis were integrated into a variety of content areas. In one mathematics class, a sampling device (a calculator, computer, etc.) was used to conduct thousands of iterations of data sampling based on a particular distribution. Students graphed the outcomes at various sample sizes and observed how the plotted outcomes approached the shape of the distribution. In a high school science context, data collection and analysis were used to identify which everyday processes could be automated. Students logged their activity over a 24-hour period then searched for patterns in the data and discussed which activities could be automated or enhanced through computation. Other sources of data used in high school data analysis activities included DNA substrand data, sensor data, and historical texts.

One of the most innovative CT integration examples came from a combined social studies and ELA context. Given access to 73 Federalist Papers (essays) and the attribution of 61 of them to one of three authors, students were tasked with inferring the authors of the remaining papers. Using collaborative sampling and analysis of word size, students were able to infer the specific authors of the remaining 12. This activity was designed for an AP Statistics course and links to the professional practices of **engaging in collective sense-making around data** and predicting or making inferences from data.

Gap Analysis: Many rich examples of activities incorporating data collection and analysis were presented in the the K-12 grade bands. In terms of moving from data collection and analysis to collective sense making, several activities pooled data from different collectors to perform

analyses of the combined data set thus elucidating the impacts of sample size and sampling variability.

Based on these findings, we make the following suggestions on moving from data collection and analysis to collective sense-making and understanding consequences and making predictions in classroom activities:

- Consider the impact of sample size, variability, and missing data on analyses and findings.
- When appropriate, discuss the data representation and formats used to prepare the data for processing by a computational agent.
- Use project based learning scenarios to frame data collection and analysis.
- Provide opportunities for students to make inferences from data sets through data analytics.

Modeling and Simulation

Modeling and simulation activities took many different forms across K–12 education. In the K–2 activities contributed, students learned about modeling by creating physical models to be used as a test beds for running simulation experiments, but the link to computer models and simulation was absent. In grades 3–5, stronger links between a physical or participatory simulation and an analogous computer model were seen. In a math activity, students engaged in drawing diagrams to break down math problems (decomposition), then used an interactive computer-based math simulation to help explain an algorithm to solve an algebra problem through balancing equations. Project-based learning activities that integrated CT through abstraction (answering “What are the key elements to include?”) were seen in grade 4, but

students were generally not expected to follow the abstraction exercises with automation or programming to create a working model and run simulations. Other activities featured readily available simulations used by students to run experiments, generate data to be analyzed, or "get a sense of" a scientific process through an engaging interactive visualization.

In the middle and high school grade bands, the use-modify-create trajectory (Lee et al., 2011; GUTS, 2014) was a pedagogical tool to progressively engage students in deeper CT. In these lessons students were tasked with using, decoding, and then modifying a computer model (through programming) to answer a question about a system's behavior under different circumstances or reflect a local condition. Subsequently, parameter sweeping experiments were conducted using the model as an experimental testbed, data from the experiments were collected and analyzed in order to assess the impact of the modifications. The use of CT—in particular, automation—allowed multiple runs to be conducted. Subsequent analysis of the data generated by the simulation enabled students to test their hypotheses about the generators of the system's behavior.

Gap Analysis: Overall, modeling and simulation activities that follow the use-modify-create trajectory mimic professional practice in STEM fields and, together with data collection and analysis, link to the professional practices of **designing solutions that leverage computational power and resources, engaging in collective sense-making around data**, **understanding potential consequences of actions**, and **understanding complex systems**.

Based on these findings, we offer the following suggestions on moving from modeling and simulation to understanding complex systems in classroom activities:

- Use computer modeling and simulation tools that enable students to “look under the hood” and decode the models they use to run simulations. Avoid “black box” simulations that cannot be inspected and evaluated.
- Discuss the abstractions and assumptions embedded in models.
- Run parameter sweeping experiments to understand the behavior of the system under different conditions. Data from these experiments can be gathered in a spreadsheet and used to generate a landscape of outcomes.
- Use a jigsaw method to distribute parameter sweeping experiments among student pairs acting as processors.
- Investigate sources of randomness in models of complex systems and discuss the impact of that randomness on the simulation outcomes.
- Discuss the need to run multiple trials at each parameter setting during experimentation due to randomness factors in the model’s behavior.
- Consider which behaviors are encoded in the model and which are emergent (generated through the interaction of the entities in the model but not specifically programmed).

CT Learning Progressions that Combine and Connect DLCS Components

Several interesting examples of linked activities that cut across DLCS components were seen. (These progressions are displayed as horizontal bars in Figure 2.) One example was the fifth grade “Willis Tower” lesson in which students were tasked with programming a robot to build a tower (the DLCS component **programming and software development**). This required students to develop a language of primitives that would be necessary to complete this task (the DLCS component **abstraction**, and the CTIE **innovating with computational**

representations). The lesson concluded with having students build a volume calculator (the DLCS components **abstraction**, **algorithms**, and **programming and software development**).

A second example was the high school algebra lesson in which students defined functions (the DLCS component **abstraction**), designed algorithms that encapsulated the functions (the DLCS component **algorithms**), and then built computer programs that executed the functions (the DLCS component **programming and software development**).

A third example was a population dynamics lesson; students engaged in a participatory simulation in which they acted out the part of a species in an ecosystem, based on the rules of interaction, and tracked their population size over time. They viewed a computer model of a similar ecosystem and saw patterns of population dynamics over many generations (the DLCS component **modeling and simulation**, and the CTIE **engaging in collective sensemaking around data**). This activity could be followed by students modifying and creating their own models of ecosystems to uncover dynamics in other ecosystems (the DLCS components **abstraction**, **algorithms**, **programming and software development**, **modeling and simulation**, and **data collection and analysis**, and the CTIE **understanding complex systems**).

A fourth example was a "water as a shared resource" module, in which students participated in a participatory simulation of sharing (or not) a finite water source among local stakeholders with different water needs. In each round of the "game," students collected data on how much water they took and if they had enough to meet their needs (the DLCS component **data collection and analysis**, and the CTIE **engaging in collective sensemaking around data**). Discussion ensued about the "tragedy of the commons" phenomenon (the CTIE **understanding complex systems**), and students developed strategies to attempt to share the water equitably. This

activity was followed by using the model to collect data (the DLCS components **modeling and simulation** and **data collection and analysis**, and the CTIE **understanding complex systems**) and examining a model of a single in-ground water pump fed by an aquifer. Students were tasked with modifying the model (the DLCS components **abstraction**, **algorithms**, and **programming and software development**) to more closely match the scenario with multiple stakeholders (i.e., water users). The new model was then used as an experimental testbed to gain an understanding of the ramifications of multiple users of the same aquifer (the DLCS components **modeling and simulation** and **data collection and analysis**, and the CTIEs **engaging in collective sensemaking around data** and **understanding complex systems**). In their final activity, students designed and built innovative storage systems for community water.

Presaging the Future

"Developing a Framework for Computational Thinking from a Disciplinary Perspective" (Malyn-Smith et al., 2018) presented a new framework that aimed to bridge between the CT practices used when learning in STEM content and the professional practices of those in CT-integrated STEM fields. In particular, the framework identified five CT elements that reflect the work of CT-enabled STEM professionals and can be integrated into schools—the CTIEs further explored in this paper.

We also found that it is possible to map from the DLCS components to the CTIEs in a number of meaningful ways. For example:

- Gaining experience with abstraction and understanding representations can be expanded to developing more powerful abstractions and innovating with representations.

- Developing and analyzing algorithms and then instantiating them as computer programs can be extended to an understanding of how tasks can be decomposed and then parallelized for running on multiprocessors.
- Collecting data and analyzing small local data sets can be expanded to crowdsourcing data (either collecting data from different regions or generating data by running simulations with different inputs or variable settings) and collective sense-making as the data are pooled, helping students understand a system on a larger scale.
- Modeling and simulation experiences along a single variable or a set of variables can be extended to running parameter sweeping experiments to gain an understanding of systems behavior as a landscape of potential outcomes.
- Modeling and simulation experiences in which students are asked to modify a model to reflect a change in a system, then run the simulation to generate data, and analyze the data to see the consequence of that change help students more deeply understand the potential consequences of changes to a complex system.

The new CT integration framework and the results of this analysis raise new questions around where we should focus our teaching, learning and assessment efforts as we help students move along a CT progression in K–12.

- Recognizing that enacting age-appropriate versions of professional practices may not be a reasonable target for students at younger ages, what is the appropriate grade band to start introducing these practices?
- If our goal to help students bridge school to work at the human technology frontier—what curriculum changes need to be made to get there most effectively and efficiently?
- How can our existing assessment tools be revised to adequately capture students' learning of concepts and practices in CT-integrated STEM fields?

- How do we best provide educators with insights into the work of professionals in CT-integrated STEM fields?
- As technological advances associated with artificial intelligence (AI) and machine learning (ML) impact both our daily lives and nearly all STEM fields, how can CT and CT integration best address the burgeoning need to prepare students for work in AI and ML?
- As literally all industry sectors are impacted by technology innovations, is the novel application of computing technologies to STEM fields a new critical skill needed for future career success?

These questions point to the importance of ongoing research on computational thinking and the integration of CT in STEM education.

Acknowledgments

We'd like to thank the participants in the two workshops on "Developing an Interdisciplinary Framework for Integrating Computational Thinking in K–12 Science, Mathematics, Technology, and Engineering Education" for their contributions of activities and discussion surrounding CT integration. This work would not have been possible without their willingness to share activities and insights.

References

Computer Science Teachers Association. (2011). *K–12 Computer Science Standards.* Retrieved from http://csta.acm.org/Curriculum/sub/k12standards.html

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning* (pp. 19–38). London: Bloomsbury Academic.

K–12 Computer Science Framework Steering Committee. (2016). *K–12 Computer Science Framework.* Retrieved from https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Lee, I., Martin, F. Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. 2011. Computational Thinking for Youth in Practice. ACM Inroads 2(1): 32-37.

Lee, I. & Martin, F. 2016. Computational Thinking. In Peppler, K. (Ed.), Encyclopedia for Out-of-School Learning. Thousand Oaks, CA: Sage.

Malyn-Smith, J., & Ippolito, J. (2011). *Profile of a Computational Thinking Enabled STEM Professional in America's Workplaces: Research Scientist* (Unpublished manuscript). Waltham, MA: Education Development Center, Inc.

Malyn-Smith, J., & Ippolito, J. (2012). *Profile of a Computational Thinking Enabled Product Engineer* (Unpublished manuscript). Waltham, MA: Education Develoment Center, Inc.

Malyn-Smith, J., & Ippolito, J. (2014). *Profile of a Big-Data-Enabled Specialist* [White paper]. Retrieved from http://oceansofdata.org/system/files/Big%20Data%20Enabled%20Specialist%20Profile_0.pdf

Malyn-Smith, J., & Ippolito, J. (2017). *Profile of the Data Practitioner* [White paper]. Retrieved from http://www.edc.org/sites/default/files/uploads/HumanTechnologyFrontierWhitePaper.pdf

Malyn-Smith, J., Lee, I. A., & Ippolito, J. (2017). Profile of a CT Integration Specialist. In S. C. Kong, J. Sheldon, & R. K. Y. Li (Eds.), *Conference Proceedings of the International*

Conference on Computational Thinking Education 2017 (pp. 60–63). Hong Kong: The Education University of Hong Kong.

Malyn-Smith, J., Lee, I., Martin, F. G., Grover, S., Evans, M. A., & Pillai, S. (2018). Developing a Framework for Computational Thinking from a Disciplinary Perspective. In S. C. Kong, D. Andone, G. Biswas, T. Crick. H. U. Hoppe, T. C. Hsu, J. Vahrenhold (Eds.), *Conference Proceedings of the International Conference on Computational Thinking Education 2018* (pp. 182–186). Hong Kong: The Education University of Hong Kong.

Martin, F. (2018, February 17). Rethinking computational thinking. *The CSTA Advocate Blog.* Retrieved from http://advocate.csteachers.org/2018/02/17/rethinking-computational-thinking/

Massachusetts Department of Elementary and Secondary Education. (2019). *Digital Literacy and Computer Science Curriculum Framework: Grades Kindergarten to 12.* Retrieved from http://www.doe.mass.edu/frameworks/dlcs.pdf

NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. Washington, DC: The National Academies Press.

Wing, J. (2006, March). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

Wing, J. (2010, November 17). *Computational thinking: What and why?* Pittsburgh, PA: Carnegie Mellon University, School of Computer Science. Retrieved from https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf