# Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?

VALERIE BARR
Union College

and
CHRIS STEPHENSON
Computer Science Teachers Association

The process of increasing student exposure to computational thinking in K-12 is complex, requiring systemic change, teacher engagement, and development of significant resources. Collaboration with the computer science education community is vital to this effort.

Categories and Subject Descriptors: K.3.2 [**Computer and Information Science Education**]: Computer Science Education; K.3.1 [**Computer Uses in Education**]: General

General Terms: Education, Curriculum

## 1. INTRODUCTION

When Jeanette Wing [13] launched a discussion regarding the role of "computational thinking" across all disciplines, she ignited a profound engagement with the core questions of what computer science is and what it might contribute to solving problems across the spectrum of human inquiry. Wing argued that advances in computing allow researchers across all disciplines to envision new problem-solving strategies and to test new solutions in both the virtual and real world. Computing has made possible profound leaps of innovation and imagination as it facilitates our efforts to solve pressing problems (for example, the prevention or cure of diseases, the elimination of world hunger), and expands our understanding of ourselves as biological systems and our relationship to the world around us. These advances, in turn, drive the need for educated individuals who can bring the power of computing-supported problem solving to an increasingly expanded field of endeavors.

It is no longer sufficient to wait until students are in college to introduce these

Author's address: V.Barr, Computer Science Department, Union College, 807 Union Street, Schenectady, NY 12308
C.Stephenson, Computer Science Teachers Association, 2 Penn Plaza, Suite 701, New York, NY 10121-00701

concepts. All of today's students will go on to live a life heavily influenced by computing, and many will work in fields that involve or are influenced by computing. They must begin to work with algorithmic problem solving and computational methods and tools in K-12. The successful embedding of computational thinking concepts into the K-12 curriculum requires efforts in two directions. Educational policy must be changed, overcoming significant infrastructure hurdles, and K-12 teachers need resources, starting with a cogent definition and relevant age-appropriate examples. In this paper we report on the first part of a multiphase project aimed at developing an operational definition of computational thinking for K-12 along with suitable resources for policy and curricular change. In addition to explaining the issues involved in the K-12 arena, this paper, following Ezer and Stephenson [11], is intended to help bridge the gap between the K-12 and the CS education communities.

The computer science education community can play an important role in highlighting algorithmic problem solving practices and applications of computing across disciplines, and help integrate the application of computational methods and tools across diverse areas of learning. At the same time, CS educators must understand the complexities of the K-12 educational setting, incorporating that knowledge into their outreach activities and their support for K-12 changes.

Developing a definition of or approach to computational thinking that is suitable for K-12 is especially challenging in light of the fact that there is, as yet, no widely agreed upon definition of computational thinking. Certainly K-12 students already learn how to think and problem solve, but computer scientists can help teachers understand these processes as algorithmic, and identify where actual computation and manipulation of data with a computer may fit in. Many disciplines require, promote, and teach problem solving skills, logical thinking, or algorithmic thinking. Computer scientists can promote understanding of how to bring computational processes to bear on problems in other fields and on problems that lie at the intersection of disciplines. For example, bioinformatics and computational biology are different, but both benefit from the combination of biology and computer science. The former involves collecting and analyzing biological information. The latter involves simulating biological systems and processes. Presenting both bioinformatics and computational biology in algorithmic form helps scientists exchange information [5].

## 2. MULTIPLE DEFINITIONS OF COMPUTER SCIENCE AND COMPUTATIONAL THINKING

Questions of the nature and educational value of computer science are as old as the discipline itself. In 1985, Abelson and Sussman argued that computer science is "a discipline of constructing appropriate descriptive languages" [1]. Denning [2], however, posited that computer science consists of mechanics (computation, communication, coordination, automation, and recollection), design principles (simplicity, performance, reliability, evolvability, and security) and practices (programming, engineering systems, modeling and validation, innovating, and applying). The ACM Model Curriculum for K-12 Computer Science [12] provides a definition of computer science specifically for K-12 educators. Computer science, it argues, is nei-

ther programming nor computer literacy. Rather, it is "the study of computers and algorithmic processes including their principles, their hardware and software design, their applications, and their impact on society" (pg.1). Computer science therefore includes:

—programming,
—hardware design,
—networks,
—graphics,
—databases and information retrieval,
—computer security,
—software design,
—programming languages,
—logic,
—programming paradigms,
—translation between levels of abstraction,
—artificial intelligence,
—the limits of computations (what computers cannot do),
—applications in information technology and information systems, and
—social issues (Internet security, privacy, intellectual property, etc.).

More recently, Felleisen and Krishnamurthy [3] have argued that "imaginative programming" is the most crucial element of computing because it closely aligns mathematics with computing and in this way brings mathematics to life.

In framing the conceptual and educational importance of computational thinking, as distinct from computer science, Wing [13] suggested that computational thinking includes: seeking algorithmic approaches to problem domains; a readiness to move between differing levels of abstraction and representation; familiarity with decomposition; separation of concerns; and modularity. More recently, Isbell et al. [6] have argued for "computationalist thinking", a focus on providing services, interfaces, and behaviors that involves a more central role for modeling as a means of formulating relationships and identifying relevant agencies that are sources of change.

As the ITEST Working Group on Computational Thinking [8] pointed out, however, computational thinking "shares elements with various other types of thinking such as algorithmic thinking, engineering thinking, and mathematical thinking". Perkovic et al. [9] similarly focus on the intellectual skills necessary to "apply computational techniques or computer applications to ... problems and projects" in any discipline. Hemmendinger [4] notes that we must be aware of the risks of arrogance and overreaching when discussing the role of computational thinking, especially across disciplines. He argues that the elements of computational thinking that computer scientists tend to claim for their own (constructing models, finding and correcting errors, creating representations, and analyzing) are shared across many disciplines and that the appearance of grand territorial claims risks provoking adverse reactions. Hemmendinger concludes that the ultimate goal should not

be to teach everyone to think like a computer scientist, but rather to teach them to apply these common elements to solve problems and discover new questions that can be explored within and across all disciplines.

## 3. CREATING A DEFINITION FOR COMPUTATIONAL THINKING IN K-12

K-12 education today is a highly complex, highly-politicized environment where multiple competing priorities, ideologies, pedagogies, and ontologies all vie for dominance. It is simultaneously subject to wildly diverse expectations, intense scrutiny, and diminishing resources. Any effort to achieve systemic change in this environment requires a deep understanding of the realities of the system. Passionate debate about the nature of computer science or computational thinking may provide intellectual stimulation for those in the computing fields. However, embedding computational thinking in K-12 requires a practical approach, grounded in an operational definition. It requires that we begin with a set of questions focused specifically on K-12 implementation:

—What would computational thinking look like in the classroom?

—What are the skills that students would demonstrate?

—What would a teacher need in order to put computational thinking into practice?

—What are teachers already doing that could be modified and extended?

To be useful, a definition must ultimately be coupled with examples that demonstrate how computational thinking can be incorporated in the classroom. Research regarding the implementation of computational thinking skills in informal education also provides valuable insights. Lee [7] for example, points to several successful projects that use simulation and modeling, robotics, and computer game design to teach abstraction, automation, and analysis. As the ITEST Working Group on Computational Thinking notes, these kinds of activities also involve an iterative design, refinement, and reflection process that Resnick [10] argues is central to creative as well as computational thinking.

In the summer of 2009, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE) began a multi-phase project aimed at developing an operational definition of computational thinking for K-12. These two organizations (see Appendix A for more information about CSTA and ISTE) were particularly suited for this undertaking because of their extensive involvement in K-12 and their expertise in developing educational standards, curriculum materials, and professional development for educators. This project would bring together computational thinking and K-12 curriculum thought leaders committed to focusing on definitions and implementation of computational thinking in the context of real K-12 curriculum outcomes, standards, and artifacts. The project began with the selection of a small steering committee that met to:

—identify criteria for and names of potential invitees for a Thought Leaders meeting; and

—develop an agenda for a two-day Thought Leaders meeting designed to create a framework/lexicon to better facilitate discussions of key elements of computational thinking across diverse disciplines.

The steering committee identified a group of educators and administrators who

—had clearly demonstrated interest in computational thinking for K-12 or expertise in curriculum development and implementation

—would provide representation from a broad spectrum of backgrounds and perspectives (higher education faculty and researchers, K-12 professional associations, school-based leaders, teachers, the corporate community),

—had experience with or demonstrated interest in K-12 issues, and

—demonstrated leadership, particularly in STEM discipline areas.

The steering committee eventually selected 26 Thought Leaders and charged them with developing a shared vision and set of strategies for embedding computational thinking across the K-12 curriculum, most especially in the STEM subject areas. The purpose of the meeting, held over two days in April 2010, was not to craft a formal or definitive definition of computational thinking to be debated by academics. Rather, the goal of the meeting was to reach a consensus of what computational thinking means in K-12, as well as explain the particularities of K-12 education to the CS education representatives. Specifically, for any K-16 collaboration to be successful, college faculty must understand the complexities of teaching in and making changes in the K-12 setting. The computer scientists participating, in particular, noted that educational change was considerably more complex than they suspected and that working with educators from multiple diverse disciplines meant learning to "disconnect computational thinking from computer science".

## 4. WAYS OF ENVISIONING COMPUTATIONAL THINKING IN K-12 CLASSROOM

The participants identified many ideas about what computational thinking is and what it could be in classrooms. When challenged with the task of describing what makes computational thinking unique from other kinds of thinking, participants tended to focus on the centrality of the computer and a set of concepts that computational thinking and doing encompass:

> *CT is an approach to solving problems in a way that can be implemented with a computer. Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. CT is a problem solving methodology that can be automated and transferred and applied across subjects.*

and its generation from, and potential use in, a wide variety of disciplines: *The power of computational thinking is that it applies to every other type of reasoning. It enables all kinds of things to get done: quantum physics, advanced biology, human-computer systems, development of useful computational tools.*

The participants envisioned computational thinking manifesting in the classroom through active problem-solving. They saw students: "enngaged in using tools to solve problems", "comfortable with trial and error", and working in "an atmosphere of figuring things out together". They also saw students using key concepts, so that "you will hear them talk about sequences, inputs, outputs, saved value,

how complex the solution is". The meeting participants also predicted that students whose learning abounded with opportunities for "computational doing" would evidence a more fluid kind of problem solving. These students would understand that "problems can be solved in multiple ways", have "a tolerance for ambiguity and flexibility" and have "reasonable expectations about the prospect of producing a working solution".





One structured model that emerged focused on identifying core computational thinking concepts and capabilities and providing examples of how they might be embedded in activities across multiple disciplines. Table 1 shows the results of these efforts.

| CT Concept, Capability | CS | Math | Science | Social Studies | Language Arts |
|---|---|---|---|---|---|
| data collection | find a data source for a problem area | source for a problem area doing probability exercises, for example, flipping coins or throwing dice | collect data from an experiment | study battle statistics, or population data | do linguistic analysis of sentences |
| data analysis | write a program to do basic statistical calculations on a set of data | count # occurrences of flips, dice throws and analyzing results | analyze data from an experiment | identify trends in the data from the statistics | identify patterns for different sentence types |
| Data representation and analysis | use data structures such as array, linked list, stack, queue, graph, hash table, etc. | use a histogram, pie chart, bar chart, etc. to represent data; use sets, lists, graphs, etc. to contain data | summarize data from an experiment | summarize and represent the trends | represent patterns of different sentence types |
| abstraction | use procedures to encapsulate a set of often repeated commands that perform a function | use variables in Algebra; identifying essential facts in a word problem | build a model of a physical entity | summarize facts; deuced conclusions from facts | use of simile and metaphor |
| analysis and model validation | validate random number generator | curve fitting | validate that the model is correct | | |
| automation | | use tools such as: Geometer Sketch Pad; Star Logo; Python code snippets | use Prove ware | use Excel | use a spell checker |
| testing and verification | debug a program; write unit tests; formal program verification | do guess and check | validate and clean data | | |
| algorithms & procedures | study classic algorithms; implement an algorithm for a problem area | do long division, factoring; do carries in addition/subtraction | do an experimental procedure | | write instructions |
| problem decomposition | define objects and methods; define main and functions | apply order of operations in an expression | do a species classification | | write an outline |
| control structures | use conditionals, loops, recursion, etc. | study functions in algebra compared to functions in programming; use iteration to solve word problems | | write a story with branches | |
| parallelization | threading, pipelining, dividing up data or task in such a way to be processed in parallel | solve linear systems; do matrix multiplication | run experiments simultaneously with different parameters | | |
| simulation | algorithm animation, parameter sweeping | graph a function in a Cartesian plane and modify values of the variables | simulate movement of the solar system | play Age of Empires; Oregon Trail | do a re-enactment from a story |

Participants also discussed the core concepts in the context of capabilities, dispositions and pre-dispositions, and classroom culture. In many ways the capabilities category is a reiteration of the core concepts, focused on what students would actually do. These capabilities include:

—Design solutions to problems (using abstraction, automation, creating algorithms, data collection and analysis);
—Implement designs (programming as appropriate);
—Test and debug;
—Model, run simulations, do systems analysis;
—Reflect on practice and communicating;
—Use the vocabulary;
—Recognize abstractions and move between levels of abstractions;
—Innovation, exploration, and creativity across disciplines;
—Group problem solving; and
—Employ diverse learning strategies.

The dispositions and pre-dispositions category arose from an attempt to capture the "areas of values, motivations, feelings, stereotypes and attitudes" applicable to computational thinking. These included:

—Confidence in dealing with complexity,
—Persistence in working with difficult problems,
—The ability to handle ambiguity,
—The ability to deal with open-ended problems,
—Setting aside differences to work with others to achieve a common goal or solution, and
—Knowing one's strengths and weaknesses when working with others.

In attempting to define a classroom culture that would be most conducive to computational thinking, the participants identified strategies or characteristics that could be considered broadly beneficial to any learning experience. These included:

—Increased use by both teachers and students of computational vocabulary where appropriate to describe problems and solutions;
—Acceptance by both teachers and students of failed solution attempts, recognizing that early failure can often put you on the path to a successful outcome;
—Team work by students, with explicit use of:
  —decomposition - breaking problems down into smaller parts that may be more easily solved,
  —abstraction - simplifying from the concrete to the general as solutions are developed,
  —negotiation - groups within the team working together to merge parts of the solution into the whole, and
  —consensus building - working to build group solidarity behind one idea or solution.

While further detail and synthesis work is clearly required (and planned for in the next phase of the project) these models provide a way to begin embedding computational thinking within K-12 formal education. This counters the potential claim that computational thinking can only be addressed in informal education experiences where discipline based-learning and classroom constraints are not major encumbrances. However, there are still considerable barriers that must be considered in any attempt at systemic and sustained change.

## 5. STRATEGIES FOR ACHIEVING SYSTEMIC CHANGE

The kind of systemic and sustained educational change proposed necessitates two sets of resources. The first is resources that will help inform educational policy makers about the nature and importance of computational thinking, its connections to learning goals that may have already been set for students (for example national and state standards), and ways it can best be integrated within the larger framework for student learning and success. The second set of resources are those that teachers need to most appropriately and effectively integrate these new concepts, first into their own sphere of content and pedagogical knowledge and then into their classroom content and practice.

In order to articulate and expand on these two set of resources, the Thought Leaders identified several strategic areas that would have to be addressed in order to successfully embed computational thinking within K-12. For each strategic area they developed a set of requirements and suggestions that would support that element of systemic and sustained change.

### 5.1 Educational policies that include computational thinking as a part of every student's education:

—Present a single message at federal, state, and local levels about the importance of computational thinking in K-12 education.

—Encourage computer science professional organizations to advocate at the federal and state levels and work with groups that are active on state K-12 standards.

—Incorporate computational thinking throughout the entire K-12 experience with outcomes that demonstrate incremental steps.

—Attach computational thinking, where possible, to existing policies. For example, it could be included as an explicit outcome of state level technology tests.

—Include in all teacher pre-service preparation programs a class on computational thinking across disciplines.

### 5.2 Shared Vision and Common Language

—Improve the relationships and communication between K-12 educators (faculty and administrators), college CS faculty, computer science professionals, and others in industry. Develop a clear statement of computational thinking as a core competency in K-12. Demystify terminology about computational thinking, give clear examples of ways it applies to and can be integrated into a range of curricular areas.

### 5.3 School and District Level Leadership Inspired to Change

—Provide materials that will help school administrators understand computational thinking so that they can see why this knowledge and skills are important for today's students. The larger CS community can help by providing suitable materials and taking advantage of opportunities to work with K-12 administrators.

### 5.4 Inspiring and Preparing Teachers to Change

—Professional development is critical to successful educational change. CS faculty can help by providing summer institutes, demonstrating the role of computational thinking in non-CS disciplines and providing relevant curricular materials.

—Encourage school administrators to provide incentives for K-12 teachers to change courses and curricula. The NSF RET grants awarded to CPATH grantees are one model that provides incentives for K-12 teachers to adopt curricular or pedagogic changes that have been piloted first at the college level.

—Provide teachers with resources to support change, including curricular materials, models and simulations, model activities, and web sites for independent student activities.

—Provide teachers with professional development and support in the form of learning communities, summer institutes, peer learning offered by teachers with computational thinking experience, exposure to industry applications where CT skills are utilized, and help identifying where computational thinking is already included in teaching.

—Make available to school districts open-source tools (blogs, wikis, forums) and web-based social networks and content delivery systems for use by teachers and students (vetted so that districts are not likely to block them).

—Encourage current professional education associations to show how computational thinking fits into their current standards/work.

—Ask professional education associations to include a focus on computational thinking in their conferences, workshops, and professional development events.

These represent strategic areas that would support the long-term goal of embedding computational thinking in K-12. They clearly demonstrate the myriad issues and obstacles involved when trying to achieve educational change in K-12. They also illustrate the critical importance of engaging knowledgeable K-12 educators in projects that purport to improve student learning, and the extent to which a successful effort will require the expertise, resources, and dedication of educators and policy makers at all educational levels.

## 6. NEXT STEPS

The next phase of this project will involve a Practitioners Workshop that will begin to develop the resources and strategies identified in the Thought Leaders meeting. The challenge will be to determine the best possible artifacts to promote the implementation of computational thinking concepts in K-12. We expect that the Practitioners Workshop will therefore include development of various resource sets. For example, a toolkit might be developed to guide high level policy work (e.g school, district, state). It might include items such as press releases, executive

summaries, and definitions for school boards. A second toolkit might consist of materials for classroom teachers, such as concept maps or a flow chart to guide planning based on existing or model curricula. While the precise set of resources and their content have not yet been determined, it is clear that the Practitioners Workshop will be focused on formulating new materials both for implementing CT concepts into the curriculum and for advocating for computational thinking as a key educational component for all students. Given efforts already under way at the college level, including the development of new curricula and resources, we expect the computer science education community will have much to contribute to this effort.

## Appendix A

CSTA is a membership organization of more than 7000 computing educators at the K-12 and post-secondary level. Its mission is to support and promote the teaching of computer science and other computing disciplines at the K-12 level by providing opportunities for teachers and students to better understand the computing disciplines and to more successfully prepare themselves to teach and to learn. Since its inception five years ago, CSTA has become the primary voice for K-12 computer science education, advocating for the importance of computer science as part of the educational canon and its centrality to all of the STEM (science, technology, engineering, mathematics) disciplines. Through its development and publication of the ACM Model Curriculum for K-12 Computer Science and supporting curriculum implementation documents, CSTA has provided the de facto national standards for computer science in K-12. CSTA also conducts ground-breaking research and has published several germinal white papers on key computer science education issues. It provides multiple levels of professional development (through workshops and annual conferences) that have helped educators improve their technical knowledge and pedagogical skills.

ISTE is recognized for its leadership to improve learning and teaching through effective integration of technology across the curriculum and throughout the education enterprise. ISTE's commitment to educational transformation is best represented by its work to develop the National Educational Technology Standards (NETS) for Students, Teachers, and Administrators. By convening K-12 educators, teacher educators, curriculum and education associations, government, business, and private foundations, ISTE built consensus for the framework and momentum for using the standards. ISTE is a also a leader in convening educators and school leaders, best illustrated by its annual conference which showcases emerging technology and innovative and effective use of technology in the K-12 classroom.

## REFERENCES

H. Abelson and G. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, 1985.

P. Denning. Great principles of computing. *Communications of the ACM*, 46(11):15–20, 2003.

M. Felleisen and S. Krishnamurthi. Viewpoint - why computer science doesn't matter. *Communications of the ACM*, 52(7):37, 2009.

D. Hemmendinger. A Plea for Modesty. *ACM Inroads*, 1(2):4–7, 2010.

T. Hey, S. Tansley, and K. Tolle, editors. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Redmond,WA, 2009.

C. Isbell, A. Stein, R. Cutler, J. Forber, L. Fraser, J. Impagliazzo, V. Proulx, S. Russ, R. Thomas, and Y. Xu. (re)defining computing curricula by (re)defining computing. *ACM SIGCSE Bulletin*, 41(4):195–207, 2009.

I. Lee. don't know yet. waiting for information, 2010.

I. W. G. on Computational Thinking. Computational thinking for youth. Technical report, Education Development Center, Inc., Newton, MA, May 2010.

L. Perkovic, A. Settle, sungsoon Hwang, and J. Jones. A Framework for Computational Thinking across the Curriculum. In *ITiCSE 2010 Conference Proceedings*, pages 123–127. ACM, June 2010.

M. Resnick. All i really need to know (about creative thinking) i learned (by studying how children learn (in kindergarten). In *ACM Creativity and Cognition Conference*, Washington, DC, 2007.

J. G.-E. . C. Stephenson. Computer science teacher preparation is critical. *ACM Inroads*, 1(1):61–66, 2010.

A. Tucker, D. McCowan, F. Deek, C. Stephenson, J. Jones, and A. Verno. A model curriculum for k-12 computer science: Report of the acm k-12 task force computer science curriculum committee. Technical report, Association for Computing Machinery, New York, NY, 2006.

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