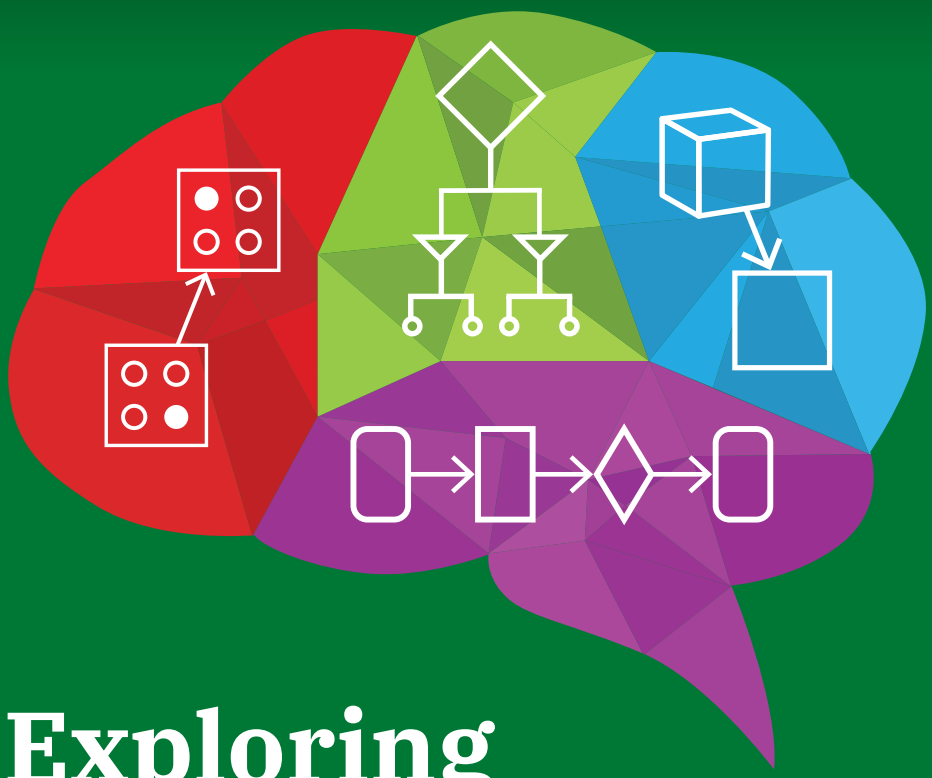
TERC
Because math and science build futures



Exploring Computational Thinking

A look into the different definitions and understandings of computational thinking

by Teon Edwards and Michael Cassidy

The *Exploring Computational Thinking* eBook is based on a blog series developed by Teon Edwards and Michael Cassidy with funding from TERC. Read more at blog.terc.edu/

Authors

Teon Edwards is co-founder of and a lead designer for the Educational Gaming Environments group (EdGE) at TERC. With a background in astrophysics, mathematics, and education with a focus on the use of technology and multimedia for learning, she has spent over 20 years developing science curricula, experiences, tools, and games for both formal and informal settings. She was production manager for the re-release of the award-winning computational thinking game *Zoombinis*, and she has since worked on multiple NSF, Department of Education, and TERC-funded projects researching CT learning across grades 3-8, including the nation-wide *Zoombinis Implementation Study*, the Research-Practice Partnership (RPP) *CodePlay* with the Braintree, MA school district, and *INFACT: Include Neurodiversity in Foundational and Applied Computational Thinking*, as well as the pilot study addressed in this article.

Michael Cassidy, Ph.D,. is a senior researcher and member of the STEM Education Evaluation Center at TERC. His research and evaluation work draw on professional experiences as a middle and elementary school science and English language arts teacher in Title I schools in Mobile, AL. His current work focuses on computational thinking, engineering education, robotics, and evaluation of mathematics and science intervention programs. He is especially interested in teachers' perspectives about their professional learning, the impact of STEM educational programs on learning opportunities, particularly for members of underrepresented groups, and application of computational thinking across content areas.

Because Math and Science Build Futures

TERC is a Cambridge, MA based nonprofit made up of teams of forward-thinking STEM experts dedicated to innovation and creative problem solving. TERC believes great futures are built through math and science.

For more than 50 years, TERC has taken a learner-centered approach to education in both formal and informal settings. At the frontier of theory and practice, TERC's work encompasses research, content and curriculum development, technology innovation, professional development, and program evaluation. Through TERC's curricula and programs, students develop the knowledge and skills needed to ask questions, solve problems, and expand their opportunities. TERC's professional development helps teachers and facilitators utilize new tools, materials, and inquiry-based strategies to enrich the learners' experience.

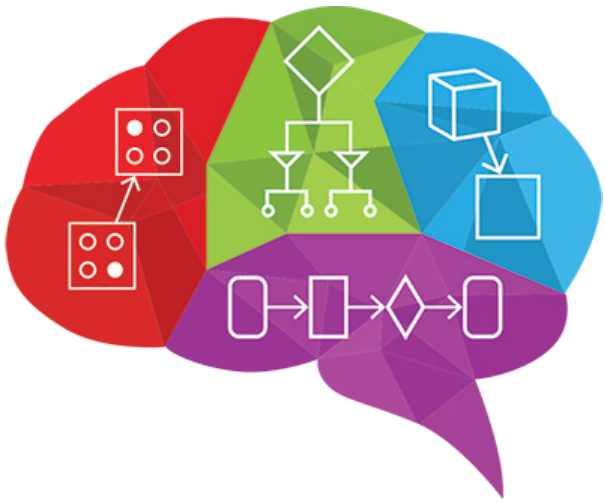
With a passion for social justice, TERC strives to create level playing fields for all learners, reaching more than three million students each year. TERC imagines a future where teachers and students are members of vibrant communities engaged in creative, rigorous, and reflective inquiry.

To learn more, please visit www.terc.edu.

Table of Contents

The Purpose of this Document

Understandings of Computational Thinking



About four years ago in May 2016, a group of TERC staff interested in Computational Thinking (CT) education started meeting to help inform and support each other in a community of practice. These meetings identified some common issues and areas of interest, including a serious, shared challenge—a general lack of agreement around the definition of CT.

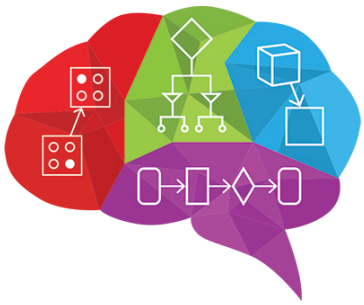
This sparked our interest, so the two of us—Mike Cassidy and Teon Edwards—have spent part of the last few years exploring different definitions and various peoples' understandings of computational thinking. In this document, we will delve into just a few of the ways teachers and people at TERC have been defining … or at least thinking about … CT in education.

During interviews, you will read how Jodi Asbell-Clarke discusses the ways she defines computational thinking as well as the ways in which her team, EdGE, think about using CT in their work. Andee Rubin will discuss computational thinking and how it is and is not related to data science. We will also look at how Jodi Asbell-Clarke connects Computational Thinking and Executive Function. And we'll be exploring some TERC projects that incorporate CT.

This document will *not* provide a singular definition for Computational Thinking. Instead, it will explore the many ways people define, think about, and use CT. At the end you might conclude that a singular definition for CT is not needed at this point in time or it might help you develop your own definition. Read on to learn and find out!

Teachers' Understandings of Computational Thinking

Computational thinking is a hot topic in education. The idea of computational thinking in education can be traced back to the work of Papert in 1980, with the term most often associated with Wing from 2006. But only over the last five years or so has computational thinking become a common focus in education ... and at TERC, where multiple projects continue to conduct research on computational thinking learning.



Four years ago, in May 2016, a group of TERC staff interested in computational thinking started meeting together monthly. The idea was for the people in the group to help educate, inform, and support each other in a community of practice. In these meetings, some common issues and areas of interest were identified, including a serious shared challenge—a general lack of agreement around the definition of computational thinking within the research field, amongst ourselves, and with our teachers. Indeed, despite efforts within the field over the years, there is still no unanimous definition of computational thinking or agreement how to best apply it in K-12 classrooms (Malyn-Smith et al., 2018).

Teachers as Part of the Discussion

In Wing's (2006) seminal piece, she stated computational thinking is "a fundamental skill for everyone, not just for computer scientists. To reading, writing and arithmetic, we should add computational thinking to every child's analytical ability" (p. 33).

Her comments sparked a debate among computer scientists, educational researchers, and other academics about what computational thinking is and what it is not, as well as how to best integrate it into education. However, classroom teachers are typically not part of the discussion.

We believe that teachers need to be represented more in this computational thinking conversation. As practitioners, they are actually bringing the computational practices, terminology, and

experiences into the classroom, as well as noticing students' ability to take up these practices and concepts. In addition, teachers are the ones most impacted by the resulting definitions, changes to and development of related standards and curricular materials, and the research directions.

Thus, the two of us decided to go beyond the monthly discussions to gain a better understanding of how teachers are thinking about computational thinking. We wanted to scope out the landscape of computational thinking education, especially as it relates to clear communication between educators and researchers.

As part of this, we sought teacher input in various ways, including via a survey. This survey included three ways of eliciting the teachers' understandings of the definition of computational thinking:

1. An open-end text box,
2. A select-up-to-5 list of central terms, and
3. A pick one definition.

We distributed the survey over multiple National Science Teacher Association listservs, via research colleagues, and through TERC's communication department. Overall, 202 teachers responded enough to be included in our analysis, with an approximately equal number from each school level (elementary, middle, and high school). Here's some of what they had to say.

Open-Ended Text Box

Early in the survey, we asked respondents, "If a parent asked you to explain what computational thinking is, what would you say?" We provided an open-ended text box for their answers.

Not too surprising to us, many teachers said they did not know (n = 21 or 9%), even though they were responding to a survey specifically about computational thinking in education. Also not surprisingly, **problem solving** was the most commonly referenced idea (n = 42 or 18%). Problem solving is core

to most computational thinking definitions, as you'll see later. It was also core to a number of the teachers' responses:

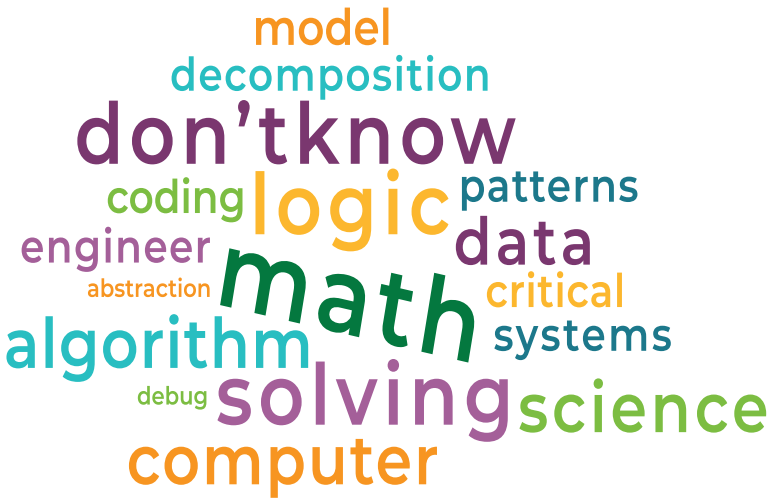
> *To think using Algorithms and **solve problems**.*
>
> *Creating and then using feedback from a system to **problem solve** using logical steps to come up with a working solution.*
>
> *How to **solve problems** using algorithms and logic.*
>
> *Computational thinking is a mindset that has to do with developing **problem-solving skills** where you are logically interweaving data analysis to develop solutions.*
>
> *Computational thinking is the process of identifying a **problem**, thinking of a solution, and ensuring that **solution** can be carried out and repeated by another.*

However, there were also a few surprises. For example, we found it interesting so many **science** teachers noted computational thinking as related to **mathematics** (n = 30 or 13%). We were also surprised that coding and computers weren't more prominent, with only n = 9 or 4% referencing coding or programming and only n = 19 or 8% referencing use of a computer.

There's a lot of debate around how central coding is to computational thinking, especially when dealing with computational thinking assessments, but a connection to computers is pretty standard. For example, teacher math- and computer-related responses included the following:

> *Mathematical and logical thinking.*
>
> *Computational thinking is understanding how computers and mathematical tools are used to analyze data and do simulations.*
>
> *Thinking like a mathematician, problem solving.*
>
> *Being able to express your ideas in a way that a computer could understand.*
>
> *All I would know to say is it is similar to activities that are done on Code.org.*
>
> *Trying to think logically like a computer would, or in a way that you can communicate with a computer.*

Select-Up-To-5 List of Central Terms



Word cloud of the terms most used by teachers in the open-ended "explain what computational thinking is" survey question.

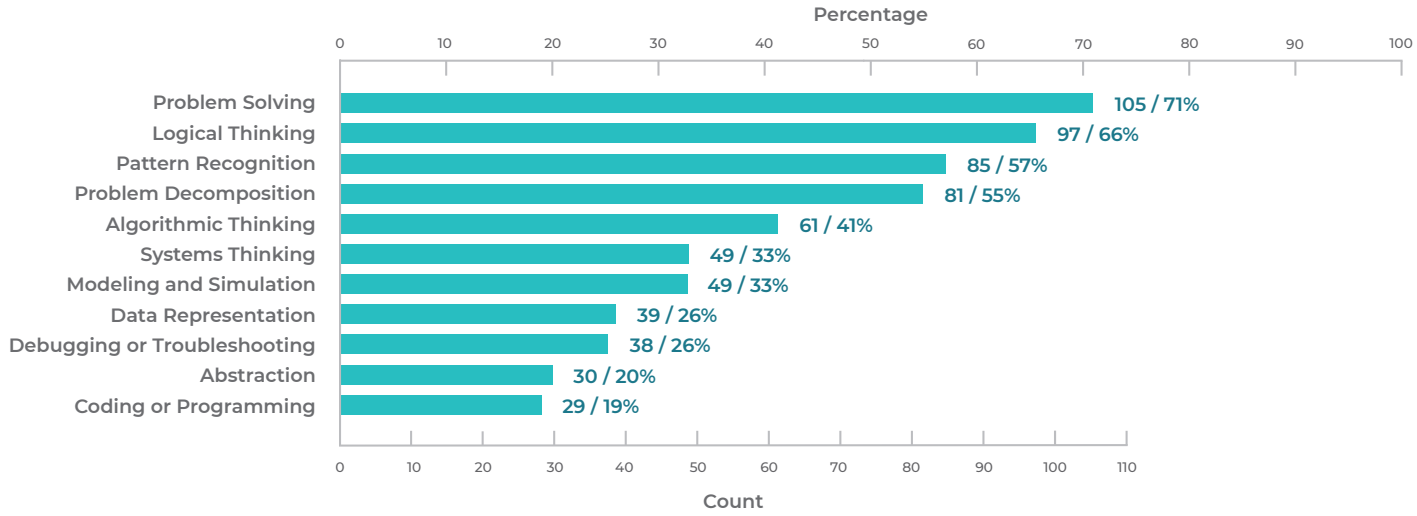
Of course, open-ended answers, while rich, are also hard to analyze, so we asked a series of subsequent questions, while not allowing the respondents to backtrack to their written answers. For example, we asked, "Which of the following terms do you consider most central to computational thinking?" with a select-up-to-five list of eleven terms commonly used in computational thinking literature:

1. Abstraction
2. Algorithmic Thinking
3. Coding or Programming
4. Data Representation
5. Debugging or Troubleshooting
6. Logical Thinking
7. Modeling and Simulation
8. Pattern Recognition
9. Problem Decomposition
10. Problem Solving
11. Systems Thinking

Within the 148 respondents to this question, the terms selected ranged widely, with problem solving again quite common (71%) and coding or programming less common (20%) than we anticipated,

based on our interactions with teachers through our projects. [A comparison of the teachers' and researchers' selections might prove an interesting area of additional exploration.]

Table 1: Terms teachers (n = 148) selected as most central to CT



OTHER INTERESTING FINDINGS:

Ability Ratings

The survey addressed an array of questions, not all related to just the definition of computational thinking. We also asked teachers to rate which common computational thinking techniques they can teach and their beliefs about what their students can do. We found all teachers were much more confident in their teaching abilities than in their beliefs of their students' skills. For example, 97% of respondents indicated they had at least an "adequate" ability to teach problem solving, while 80% of them indicated they believed their students' had at least an "adequate" ability with it. Contrast this with 52% and 38%, respectively, for coding or programming.

Our results showed that grade level did not matter across these skills, except in two areas: algorithmic thinking and programming. In both areas, elementary school teachers were more confident in both their teaching ability and their students' ability. We hypothesize the elementary teachers were more confident in both areas because of differing complexity of these terms at each grade level, as well as the tools that are used (e.g., block-based vs. text-based programming).

Pick One Definition

The survey finally offered a select set of commonly used definitions and asked respondents (n=147) to pick the one they identified with most.

- Wikipedia citing Wing (2014): "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out."
- ISTE (International Society for Technology in Education): "Computational thinking is a problem-solving process that includes (but is not limited to) formulating problems, analyzing and representing data, and algorithmic thinking."
- Wing (2006): "Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."
- CSTA (Computer Science Teachers Association): "Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer."
- Created by Authors: "Computational thinking is what you do when you use a computer."

Respondents picked the ISTE description most at 50%, with Wing (2014) from Wikipedia at 26%, Wing (2006) at 14%, CSTA at 9%, and—thank goodness—our made-up definition at only 1%. The high selection of the ISTE description is not surprising, as many of the respondents mentioned ISTE when asked which conference(s) related to computational thinking they had attended. Additionally, during interviews (conducted separately from the survey), teachers mentioned ISTE as the most common place they saw computational thinking referenced.

What's This All Mean?

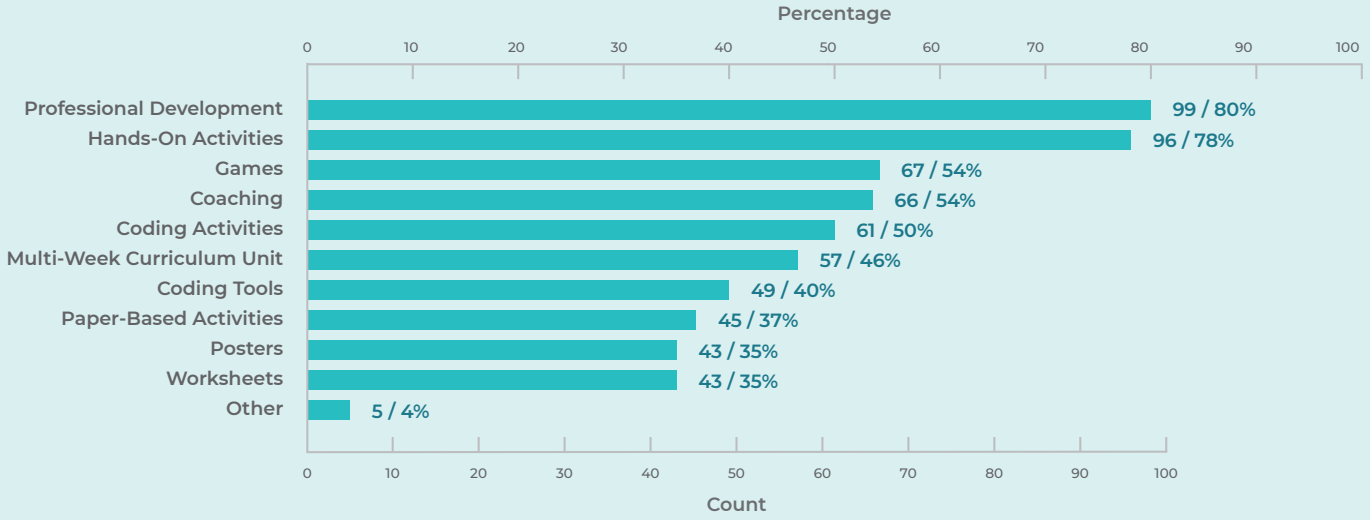
We believe an understanding of computational thinking and its roles that is shared by researchers and teachers is vital to furthering the field of computational thinking education research and development. By this, we do NOT mean a single definition; instead, we are striving for shared understandings about different ways we think and talk about computational thinking as well-focused, shared understandings within the scope of an individual project or effort. Such shared understandings are important to clear communication, to logical research findings, to appropriate assessments, and so much more.

OTHER INTERESTING FINDINGS:

Desired Supports or Materials

The survey also asked teachers what types of supports or materials they wanted to help integrate computational thinking practices into their teaching.

Table 2: Supports and materials teachers (n = 123) selected as most desirable to help them integrate CT.



Our survey and this article are very small attempts at building toward these shared understandings. For us, a main take-away is simply a refinement of where we were when we started this work: *When we, as education researchers, are talking to and working with teachers around computational thinking, we need to remember we aren't all necessarily speaking the same language.* For any project, early sharing of all parties' perspectives and reaching a shared agreement on what will be meant for the work together are always important. These principles are even more important in computational thinking education.

As the debates over computational thinking continue, let's all try to have teachers, and the realities of their classrooms, be part of the conversation.

Acknowledgements

Thank you to NSTA for the use of their listservs and to the teachers who participated in the survey. We would also like to thank TERC for funding this project.

References

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. CTE 2018 conference proceedings.

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

Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*.

Defining Computational Thinking

An Interview with Jodi Asbell-Clarke

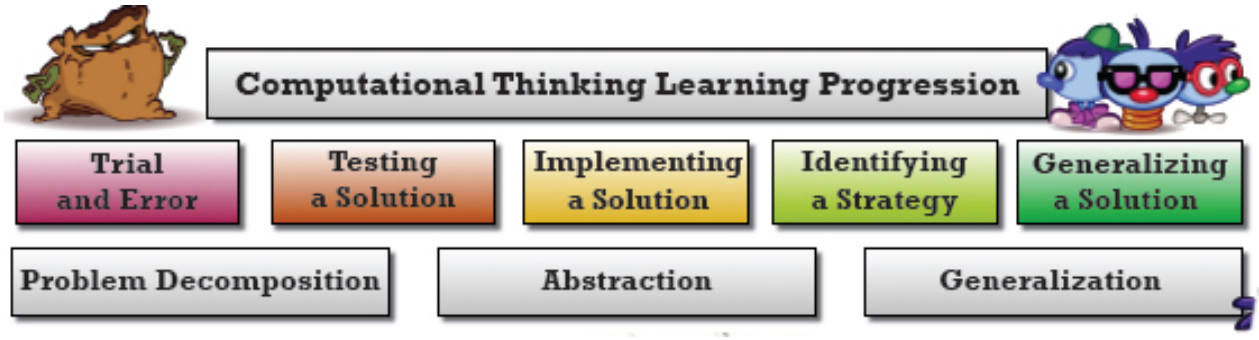
Jodi is the director of the **Educational Gaming Environments group (EdGE)** at TERC, where for the past five or so years, she has led a variety of projects that address computational thinking in education.

Teon: Thank you for agreeing to talk with us, Jodi. Can you start by telling us a bit about your definition of computational thinking?

Jodi: Thanks for having me. **A paper I co-authored with Val Shute and Chen Sun** defines computational thinking as the conceptual foundation required to effectively and efficiently solve replicable and scalable problems. Honestly, however, I don't think the field is mature enough to have a solid definition yet. Indeed, I don't know that it's advantageous for the field to hone in on an exact definition yet. But I think for our work, these four fundamental practices—problem decomposition, pattern recognition, abstraction, and algorithm design—underlie whatever definition of computational thinking emerges in education.

If learners understand how to break down a problem into smaller components, problem decomposition. If they recognize patterns. If they're able to abstract those patterns into generalizable ideas. And if they can then build algorithms around those ideas, they have the tools they need to be able to do coding, debugging, modeling, and other applications of CT.

Mike: Alright, so one of our main purposes in conducting these interviews was to get at various peoples' definitions of computational thinking. And now you don't have one for us!



Actually, though, this isn't surprising. Conversations around TERC, as well as communications with teachers, suggested that understandings of CT were many and varied, with different projects thinking about CT in different ways. What I'm wondering now is whether you think it's important to have a clear definition or understanding of CT within a particular project.

Jodi: When we consider computational thinking as a type of problem solving, and if we consider the four fundamental practices as underlying those, then I wouldn't say you define it for your specific project. However, I do think that every researcher has to operationalize it for their specific context.

So for example, we took a lot of time to look at ***Zoombinis*** gameplay, which was not designed for this particular task [of addressing CT practices], but we knew it was in there. We knew those practices were embedded in the problem solving one must do to be successful at Zoombinis. So our task was to operationalize those practices in the context of Zoombinis enough that we could seek evidence of those practices within the game play.

Teon: Ah, *Zoombinis*. We'll come back to that in a moment. [And this will be the topic of our next blog post.] Before we move on from the definition of CT, do you have any final thoughts to share.

Jodi: To me, computational thinking reminds me of where scientific inquiry was several decades ago. We know it's important. We know it's happening in schools. We're not exactly sure what that means, and we're not exactly sure how to measure it yet.

Computational Thinking and Game-based Learning

One of Jodi Asbell-Clarke's recent Computational Thinking (CT)-related projects was a national implementation study around the game *Zoombinis*. *Zoombinis* is an award-winning puzzle game developed by TERC in the 1990s and rereleased for modern computers and tablets in 2015. In the game, players guide groups of little blue characters, called *Zoombinis*, through increasingly difficult logic puzzles, as they flee the evil Bloats and journey to a new homeland.

Teon: Jodi, as you mentioned earlier, some of your CT work has focused around the game Zoombinis. Can you talk to us a bit about how computational thinking, or the four fundamental practices you mention, are shaped or even defined by your work with *Zoombinis*?

Jodi: It's complicated, but basically, we watched kids solve the puzzles in *Zoombinis*—many, many kids and many, many iterations of the puzzles—and we had researchers identify those four practices [problem decomposition, pattern recognition, abstraction, and algorithm design] within the kids' game play.

Mike: Can you talk to us a bit more about how this research actually worked?

Jodi: We used a process in *Zoombinis* that we've used with other games to study what we call **implicit learning**—learning that can't be necessarily articulated on a test or in a question, but that manifests itself as behaviors or practices within a game.

What we do first is watch players either in real time or video, on Screenflow, and listen to what they're saying as they solve the problems.

After that, we collect the back-end data, meaning the events corresponding to all the clicks or drags in the game, as well as the events that make up the game play and related data, such as timestamps. Together, this creates a data log for each player. And we can use these data logs to re-create a video of the game play, with a simulated playback tool we've developed.

We then have two researchers examine the video replay of many players (around a hundred for this study) and human code the data for the four CT practices and evidence of those practices. And we do that until we reach high inter-rater reliability between those two researchers. Once we have that, we have the evidence we're looking for in the data logs and we can build automated detectors to build that data.

Mike: So let me get this straight. After what sounds like a lot of hard work and analysis, you're now able to collect data from players of *Zoombinis*, and through the use of detectors, identify when players demonstrate your four fundamental practices of CT: problem decomposition, pattern recognition, algorithm design, and abstraction … at least as they appear within the *Zoombinis* puzzles?

Jodi: Yes. We knew those practices were embedded in the problem solving that one must do to be successful at *Zoombinis*. So our task was to operationalize those practices in the context of *Zoombinis* enough that we could seek evidence of those practices within the game play.

Now that we have those detectors, we can identify the practices in an unlimited number of players without having to go and spend time or resources on site collecting that data.

Mike: That's amazing.

Teon: Jodi, this work was part of a nationwide NSF-funded implementation study of the computational thinking learning within the game *Zoombinis*. Can you tell us more about that broader study?

Jodi: In our study, we recruited classes who would have the kids play the game and bridging activities—classroom activities that bring the game alive into the classroom and connect gameplay with classroom content. And we studied the effect of the game, as well as the effect of game plus bridging on student learning of computational thinking.

Mike: And what did you find?

Jodi: Well, here is the main takeaway ... Overall, the more kids play *Zoombinis* and the more they demonstrate computational thinking practices in their gameplay, the better they did on outside measures of CT. Neurodiverse learners appeared to particularly excel. In fact, in a small sub-study, we found we couldn't see a difference in outcomes, after the use of *Zoombinis* and related activities, between kids who had IEPs (Individual Education Plans for neurodiversity) and those who didn't. The numbers were low in that sub-study, so we want to do further research, but we're still really excited by the result. *Zoombinis*, with bridging, seems to have a positive impact on computational thinking learning.

Computational Thinking and Neurodiverse Students

This section wraps up the interview with Jodi and discusses how her work with computational thinking has turned her attention toward neurodiversity and executive functions.

Teon: Jodi, you and I co-founded EdGE together, along with Jamie Larsen, about ten years ago. And one of our main objectives was to try to broaden participation in STEM, using games to get to disengaged learners—kids who don't consider school their primary focus but who are spending time playing games.

Jodi: Yes. We've done a lot of research in **STEM game-based learning** and assessment, and throughout that work, we knew we were engaging a different set of students. It's the kids who are often disengaged in school who thrive in a game-based learning setting.

Teon: Can you talk to us a bit about how this has played out around CT?

Jodi: When we turned our attention to computational thinking, we saw that not only was the game-based learning of help, but computational thinking itself was a way to tap into some of the cognitive strengths of learners who don't often succeed in other settings.

Mike: You say you saw this potential in CT. What did that actually look like?

Jodi: So this came from teachers in the first place, because when we were using *Zoombinis*, even in the early days in classes, we

were finding teachers saying, hey, it's my kids who usually struggle academically who are becoming leaders, not just doing fine at *Zoombinis*. They're leading the class in *Zoombinis*. And that's changing other students' perception of them and the students' perceptions of their own skills and capabilities.

Starting with this, and based on our subsequent research and learning, we see computational thinking as an avenue for inclusion, not just a need to support other students.

Teon: Indeed, this has turned EdGE's attention toward neurodiversity.

Jodi: Yes. We've learned that computational thinking in and around games is a way to heighten the strengths and reveal the strengths of learners who may not be able to demonstrate those strengths in other ways. Many neurodiverse students in classes are very capable. They're great problem solvers. They have wonderful ideas. It just doesn't come out in typical schoolwork.

We're looking at how to support students with autism, ADD, dyslexia, and other learning differences. Our recent research has raised our awareness that computational thinking may be a field that actually includes cognitive assets of some neurodiverse learners. What I mean by that is that kids with autism may be particularly good at some aspects of pattern recognition, and learners with ADHD may be particularly good at thinking outside the box and finding creative solutions. We want to capitalize on those cognitive assets, while also providing support for learners, and not introducing barriers that may stand in the way of their computational thinking. A word problem or a math problem with a lot of symbols and formalizations may stump a neurodiverse learner, even though they really understand the content underneath. We want to be able to eliminate the barriers that may keep them from demonstrating that understanding.

Teon: Don't forget the teachers.

Jodi: Right. In our **research practice partnership with Braintree Public Schools**, a number of the teachers started realizing that computational thinking overlapped tremendously with exactly the type of problem solving and executive function skills they try to teach all the time. Teachers said, oh, I've been teaching this for 20 years, but computational thinking finally gives me words for the type of problem solving skills and the executive function skills that we know our kids need.

Computational thinking gives the teachers a set of practices they can distribute across their curriculum, and they actually have words and touchstones for these practices. Just having that hook to hang their hat on is grounding for the teacher, and allows them to return to this and thread it through their curriculum.

Mike: You mentioned executive function a couple of times.

Jodi: We're really excited about the overlap between computational thinking and executive function. Executive function can be described as the processes one needs to be able to solve goal-oriented tasks or problem solving. These boil down to processes like breaking down problems into smaller problems and keeping track of where you are in that problem solving, seeing patterns across sets of problems that can be generalized and used in life skills as well as academic settings, and building routines, or you might call them algorithms, to accomplish problem solving and to pursue the tasks that we need to do in our daily lives.

These closely overlap with the practices we're looking at in computational thinking. So we see a lot of potential for CT to address and assess neurodiverse learners.

Is Data Science Part of Computational Thinking?

An Interview with Andee Rubin

Delving further into the ways people at TERC have been defining … or at least thinking about … CT in education, we turn to part of our interview with **Andee Rubin**. Andee's a mathematician and computer scientist, who's been at TERC for over 27 years. She's been working in statistics and data science education, often using computers, since the late 1980s, with students as young as kindergarten and as old as seniors in high school, and with teachers across the grades.

When we asked to interview Andee, she questioned if she was an appropriate person to include. So with this post and Andee's input, we're exploring—but not trying to answer—a top-level question: What's the relationship between data science and computational thinking?

Mike: Thank you for agreeing to sit down with us, Andee. I know you weren't certain this was a good fit. Still, if we forced you to define CT …

Andee (cutting Mike off): I would refuse! (laughter all around)

Mike (smiling): Alright, then can you start us off with some of your thoughts on computational thinking?

Andee: Let me see. I think if using computational tools is computational thinking, then we're doing that all the time in our work with data. It doesn't make sense to me to talk about working with data without thinking about using computational tools. I'm not sure it ever made sense, but we didn't have good tools until the last couple of decades.

What's important is the facility with which a computational tool allows you to engage with data and make meaning—I like using that term, making meaning with data, because that's not always the way it's presented, but that's really what it's about for me. Anyway, making meaning with data is almost impossible to do without technology.

So I think there is computational thinking, by my definition, just in the use of computer tools to analyze and make meaning with data. But I think the deeper piece of computational thinking happens when you develop a piece of re-runnable code that can do the same process multiple times.

Teon: That's a lot to take in, and that last piece, in particular, gets us closer to some areas where people will see more obvious CT connections. However, I want to first step back for a moment to why we're talking at all with someone focused on data science when our primary focus is computational thinking.

Andee: Putting data science in the context of computational thinking comes out of the **Weintraub paper** that many people use as their definition of computational thinking. There are four parts of that framework, and one of them includes multiple ways of working with data that I would call components of data science. So I'm happy to consider data science part of computational thinking, or to consider it a separate field that has its own right to exist, whether or not we're working on computational thinking.

Mike: Fair enough. So backtracking to what you said earlier about the use of computational tools and their role in making meaning with data and possibly in computational thinking. Can you talk a little bit more about this?

Andee: The tools we're using [in our projects with students] involve direct manipulation of data—students can drag a variable to an axis, can filter to look at only a subset of the data, and such. Students don't need to code to accomplish those things, but they're doing things that could be accomplished by coding, especially if you want to do the same set of steps on multiple data sets.

I think, perhaps, the computational thinking really comes in when you want to do the same process over and over again, so you need to define.

Mike: At the risk of putting terms in your mouth, it sounds like creating algorithms is quite important for something to be defined as computational thinking.

Andee: For me, yes. And it's not just creating algorithms. I think sometimes we call anything that is a series of steps that have to happen in order computational thinking. There's the classic making a peanut butter and jelly sandwich as computational thinking, and I'm not so sure that gets us very far. I think including variables in the steps about how you make a sandwich—take the first thing and take the second thing—gets closer, to me, to computational thinking. It has the idea of variables and abstraction in it. But I think we oversimplify computational thinking by just saying, oh, it's any sequence of steps that has to go in that order.

Teon: You've now touched upon algorithm design as another possible component of computational thinking. How about abstraction?

Andee: Yes, I believe that abstraction is key. But again, we have to be careful about what we mean by abstraction. Abstraction is a term people have been using for a long time, and I bet we all mean something slightly different by it. So I think it bears more analysis.

We also need to be careful in talking about "finding patterns," which is another one of the often-mentioned aspects of computational thinking. We ask students to find patterns in alternating colors in kindergarten and first grade in math. Do we want to say that's computational thinking? What does it get us to say that, or what does it get us to say, no, that actually isn't CT?

So I think that's the question to ask: What leverage does it [talking about something as CT] get us?

Teon: A question for us all to keep in mind. Thank you, Andee.

Key Take Aways

In her interview, Andee discussed the contribution of Weintrop's paper from 2015 and how it relates to CT and in her work. She also points out that computational tools are important when discussing CT. However, she will not offer a concrete or operational definition of CT. Andee cautions the field that many terms associated with computational thinking are being oversimplified and there might be different meanings for each term. She encourages everyone to be specific as possible when discussing computational thinking to make sure others know exactly how they are defining these terms in their work.

Computational Thinking and Executive Function: Where Neurodiversity Shines

By Jodi Asbell-Clarke

Educators understand more and more these days that each student's brain works a little bit differently. Every learner has unique cognitive strengths (or assets) and some weaknesses (or deficits). Parents know that each child learns and plays differently too. Some children express themselves readily through art or music, some are fascinated by the natural world outdoors, while others are delighted by an entire afternoon with a difficult jigsaw puzzle.

As schools serve increasingly diverse student populations, the need for educators to differentiate learning activities to meet the needs of their students is growing tremendously (Immordino-Yang & Darling-Hammond, 2018). Adapting a lesson to engage all students—including those with learning issues related to neurology (e.g., ADHD, autism, or dyslexia)—and to keep them persistent and productive in their tasks is not easy. It requires considering the cognitive assets and deficits of each child to leverage learners' strengths to support them while they power through tougher assignments. Educators need support to deliver classroom approaches that are inclusive and draw on the unique strengths of neurodiverse learners (Tomlinson, C. A., & Strickland, 2005). In particular, technology such as video games may play a key role in supporting learners with diverse needs (Goodwin, 2008; Parsons, Leonard, & Mitchell, 2006).

Neurodiverse learners' tendency toward systematic behavior and compulsion for detail, labeled in school as a "learning disability" related to cognitive inflexibility, can be seen as exactly the skillset needed to thrive in a computational world (Abraham, Windmann, Siefen, Daum, & Güntürkün, 2006; Dawson et al., 2007; Schmidt & Beck, 2016; White & Shah, 2011). Many IT companies, such as Microsoft, have specific hiring programs for neurodiverse people, because the companies understand the unique capabilities these employees bring to the table for tasks such as quality assurance and debugging software. Divergent thinking and impulsive reactions that might be seen as disruptive to classrooms could be just what a design team needs to break through a rut in problem-solving.

This overlap between neurodiversity and technology-related problem solving has led our team to study the intersection between Computational Thinking (CT) and Executive Function (EF). These are two "hot areas" in education and may have more in common than first meets the eye. Our current project INFACT (Including Neurodiversity in Foundational and Applied Computational Thinking) developed out of our research observing how students build CT skills from their use of video games, and educators' reflections on how different types of learners engage with CT. We are now building tools that prepare students for a computational world and also support executive function, so each learners' unique strengths can shine.

Computational Thinking

Computational Thinking has been discussed in education since the mid 1990s and is now being adopted in many state standards (CSTA, 2017; Shute, Chen & Asbell-Clarke, 2017; Wing, 2006).

There are numerous programs aiming to teach CT, from pre-school through adult classes. CT can be thought of as the set of practices used when humans solve problems similarly to how computers solve problems. It involves devising and classifying problems that could have similar solutions, then building sets of instructions (algorithms) for solving groups or classes of problems, rather than solving each new problem from scratch. CT practices include:

- **Problem Decomposition:** breaking up a complex problem into smaller, more manageable problems;
- **Pattern Recognition:** seeing patterns among problems that may have similar types of solutions;
- **Abstraction:** generalizing problems into groups by removing the specific information and finding the core design of each problem; and
- **Algorithmic Thinking:** thinking of problem-solutions as a set of general instructions that can be re-used in different settings.

While a natural application of CT is coding (computer programming), there are many learning activities and uses for CT without a computer. When considering CT as a mode of problem-solving, one can see many applications of CT even in daily life. For example, writing a recipe or designing an instruction manual for a piece of equipment is sometimes described as a CT activity. Recipes and manuals could be seen as algorithms—sets of instructions to be implemented by another user.

We would argue, however, that a handwritten recipe card from your grandparent with instructions for their famous sweet and sour chicken (for instance) is not an algorithm, because it does not demonstrate the concept of abstraction that is core to CT. Abstraction is about generalizing instructions (here, a recipe) to provide the basic structure that a user can apply to a variety of contexts. An abstracted recipe (or algorithm) could describe how a chef makes a sweet and sour sauce. In this case, we see the structure:

- **one third something savory**
- **one third something tangy**
- **one third something sweet**

This general pattern is an algorithm that is re-usable with different ingredients. In one case the chef may use soy, lemon, and honey; and in another case they may use herbs, vinegar, and sugar. But even for folks who are not aspiring chefs or computer programmers, CT may be a useful way to think about how our brains work.

Executive Function

Executive Function (EF) is rapidly being recognized as a key area of focus for education for all learners, not just those in special education (Immordino-Yang & Darling-Hammond, 2018; Meltzer, 2018). A neurological description of EF usually includes:

- **Working Memory:** how we store information in the short term as we are solving a problem;
- **Cognitive Flexibility:** how well we can express and modify our thinking when provided new information; and
- **Inhibitory Control:** how well we can squelch tendencies to do things we shouldn't do, and focus on the things we should do.

Psychologists and educators consider the social and emotional aspects of executive function including emotional regulation, motivation, and metacognitive processes like planning a task,

organizing steps and information, and monitoring progress. An educational perspective of EF refers to how these processes play out in the classroom with regard to students' ability to:

- **retain information while reading a passage or solving a word problem;**
- **express their thinking and refine their ideas with experience;**
- **focus and navigate their way through a task; and**
- **manage frustration and regulate emotions.**

CT and Executive Function

Over the past several years, our team has been studying how learners in grades 3-8 build CT practices through games such as TERC's popular logic puzzle game, *Zoombinis* (available at zoombinis.com). We also partnered with a Massachusetts school district in a Research-Practice Partnership to infuse CT into their classroom curriculum for grades 3-8. Throughout our research, we found teachers observing that some learners who struggled in other subjects became more engaged and more productive when doing CT activities—sometimes even becoming leaders in their class.

The struggles of many learners in school boil down to issues with EF. The practices of CT—breaking down problems into smaller pieces and finding patterns in problems so that they can generalize solutions—are also practices that support EF. They help learners to focus and navigate their way through tasks and to refine their ideas with experience.

We also found that special education teachers were excited by teaching CT, because, as one put it, "These are the problem-solving skills I always try to teach our kids, and now I have words for it. And I have a way to embed it right in the curriculum." Teachers with English Language learners noted the same thing: CT helped them support their learners by making learning explicit. Teachers saw this type of success spill over into other areas by the building of student confidence and social capital as well as academic skills. These observations led us to study the intersection between EF and CT.

Examples of How to Support EF in CT

CT can be seen as useful strategies for solving problems of all kinds, particularly when encountering similar problems or tasks over and over again. Calling out and emphasizing CT practices may help support EF in other areas.

Decomposition

A 1,000-piece jigsaw puzzle may seem like a daunting task at first, but when you break it up into sub-tasks it becomes more manageable. There are many ways puzzle solvers decompose the problem, such as working on the edge first before tackling the interior, or choosing one region of the puzzle to work on at a time.

Pattern Recognition

Many people sort pieces by color, while others look at the shapes of pieces and the number of "innies" and "outies". These patterns help the problem become more manageable and provide information that makes the puzzle solution more apparent.

Abstraction

Puzzle solvers may begin to generalize about types of pieces, for example by collecting all

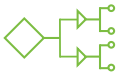
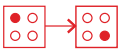
those that have "outies" side by side before finding pieces that exactly fit together. In this systematic method, the solver doesn't have to try each piece every time; they only have to start by finding pieces that fit the general category of "outie".

Algorithmic Thinking

An expert jigsaw puzzler might always put the edge together first, then group by color, then sort those piles by shape before they assemble. Their problem-solution can be thought of as a set of general instructions that can be re-used in different settings and that helps them develop fluency in puzzle solving.

These same practices can be thought of as ways to support EF in problem solving. Table 1 shows the relationship between CT practices, jigsaw puzzle-solving activities, and EF.

Table 1

CT Practice		Jigsaw Puzzle Activities	EF Support
	**Problem Decomposition**	Breaking down puzzle into regions or types of pieces	Smaller problems are easier on the working memory.
	**Pattern Recognition**	Grouping pieces by shape or color	Seeing patterns can help with cognitive flexibility by relating one context to another.
	**Abstraction**	Searching for general types of pieces to fit into places	Generalizing solutions can simplify problems which can help with retaining information, and focused navigation through a task.
	**Algorithmic Thinking**	Developing re-usable routines to solve the puzzle	Developing an algorithm helps with explicit thinking and task navigation.

Designing Supports for Neurodiverse Learners

INFACT is using these ideas to design, implement, and study a comprehensive and inclusive CT program to support teachers and students for grades 3-8. It focuses on the cognitive assets of neurodiverse learners and builds in supports for learners with a wide variety of differences in attention, metacognition, and self-regulation. The materials engage learners' EF within CT activities to help make learners' thinking visible and their problem-solving productive. Learn more at **https://www.terc.edu/projects/infact/**

For example, we are building a flashlight tool that highlights relevant information that a learner might not be attending to, so that they can focus on the salient information in an activity. We are designing graphical organizers that help learners keep track of necessary information, removing the load on their working memory, and helping them organize the information in ways that make meaning. We are also providing an expression tool that helps learners make their implicit thinking visible, so they can see exactly what they've done in one task and re-use similar strategies for future problem solving.

Currently we are designing these supports within digital learning activities, such as games like Zoombinis, so that we will be able to use data mining algorithms to make the supports adaptive. We are building models to detect when students are getting overly frustrated or bored and where in the activities they are no longer productive. When students persist unproductively, it is called wheel-spinning and can lead to disengagement. By detecting in real-time the "trigger" points just before wheel-spinning starts, we are planning to intervene with a "just-in-time" support—like suggesting a strategy they've used previously, highlighting useful information they might be missing, or suggesting they take a break and come back after re-energizing. Finding ways to react to each learner's levels of engagement and potential wheel-spinning through automated data mining detectors will allow us to support individual learners' unique needs.

CT activities offer unique opportunities to support EF, and in turn by supporting EF we strive to improve learners' CT. This symbiosis of these two areas may show us a way to help create a world where learners with many different cognitive assets and challenges will thrive. And where our society will benefit from the creativity and intelligence that all learners have to offer.

Acknowledgements

The author is grateful for generous funding from the National Science Foundation's Division on Research and Learning and the US Department of Energy's Education Innovation and Research programs for ongoing funding in this area. This work would not be possible without the talents and contributions of the EdGE at TERC team, and the many students and teachers with whom we've worked.

References

Abraham, A., Windmann, S., Siefen, R., Daum, I., & Güntürkün, O. (2006). Creative thinking in adolescents with attention deficit hyperactivity disorder (ADHD). *Child Neuropsychology, 12(2)*, 111–123.

CSTA K-12 CS Standards (2017). Computer Science Teachers' Association. Retrieved from https://www.csteachers.org/page/standards.

Dawson, M., Soulières, I., Ann Gernsbacher, M., & Mottron, L. (2007). The level and nature of autistic intelligence. *Psychological Science, 18(8)*, 657–662.

Goodwin, M. S. (2008). Enhancing and accelerating the pace of autism research and treatment: The promise of developing innovative technology. *Focus on Autism and Other Developmental Disabilities, 23(2)*, 125–128.

Immordino-Yang, M. H., Darling-Hammond, L., & Krone, C. (2018). *The brain basis for integrated social, emotional, and academic development*. Aspen Institute, National Commission on Social, Emotional, and Academic Development. https://www.aspeninstitute.org/publications/the-brain-basis-for-integrated-social-emotional-and-academic-development/

Meltzer, L. (2018). Creating strategic classrooms and schools: Embedding executive function strategies in the curriculum. In *Executive function in education: From theory to practice.* Meltzer, L. (Ed). Guilford Publications.

Parsons, S., Leonard, A., & Mitchell, P. (2006). Virtual environments for social skills training: Comments from two adolescents with autistic spectrum disorder. *Computers & Education, 47(2)*, 186–206.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Tomlinson, C. A., & Strickland, C. A. (2005). *Differentiation in practice: A resource guide for differentiating curriculum, grades 9-12*. ASCD.

White, H. A., & Shah, P. (2011). Creative style and achievement in adults with attention-deficit/hyperactivity disorder. *Personality and Individual Differences, 50(5)*, 673–677.

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

Computational Thinking Resources and Examples

TERC's Work in Computational Thinking Education

TERC integrates Computational Thinking (CT) into STEM education across grade levels and in both formal and informal settings. Educators at TERC explore what CT can look like in STEM learning environments, both with and without coding. Check out the projects showcasing TERC's work in CT integration, research methods and assessments, and much more.

CodePlay

Through a Researcher Practitioner Partnership (RPP) between Braintree Public Schools and EdGE, the team is building CodePlay—a strong foundation of teachers and suite of materials for the teaching and learning of CT—in upper elementary and middle schools across Braintree MA, considering a broad audience of diverse learners with cognitive differences.

Learn more

Building Systems from Scratch

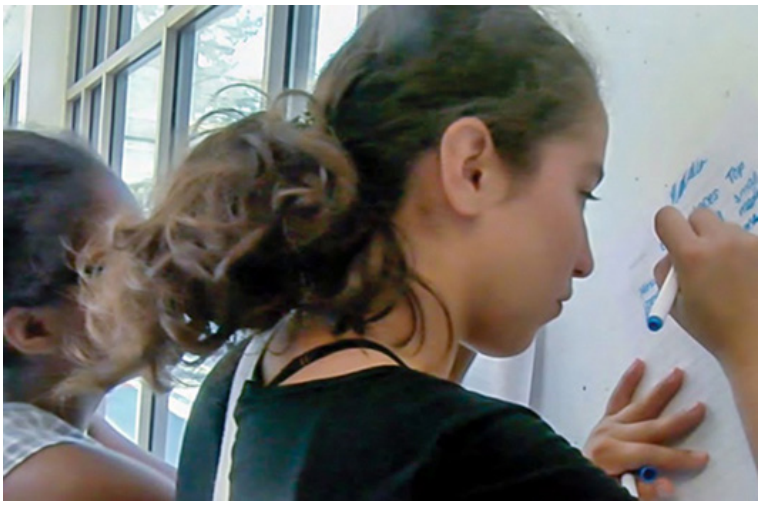
We know that students learn from playing educational games. Building Systems from Scratch believes there is more joy and empowerment, and deeper learning when students design the games themselves.

Learn more

Data Clubs

Developing data science materials and activities for after-school and summer experiences. Data Club's goal is to introduce middle school youth to the power of data by giving them simple tools for visualizing and analyzing data on topics they care about.

Learn more



IDATA

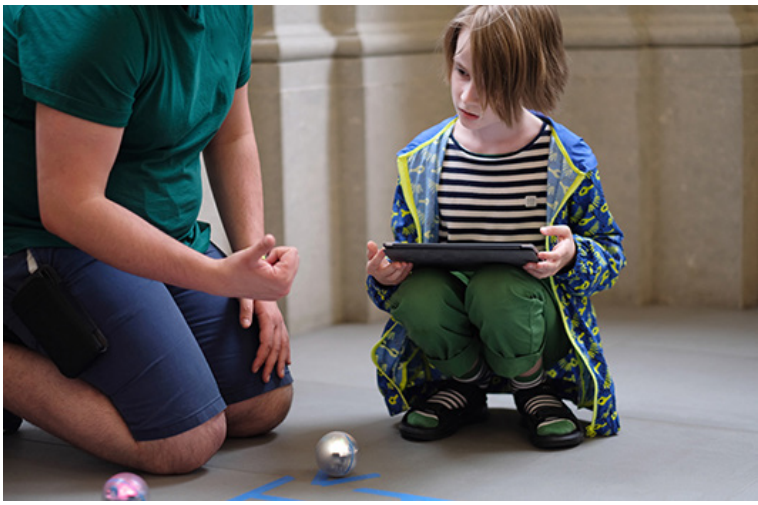
IDATA engages middle and high school students in designing software to make astronomy accessible to people with blindness or visual impairments.

Learn more

INFACT

INFACT is designing, implementing, and studying a comprehensive program for inclusive CT for grades 3-8, focusing on the cognitive assets of neurodiverse learners and building in supports for learners with a wide variety of differences in attention, metacognition, and self-regulation.

Learn more



Zoombinis

This project leverages the existing Zoombinis game by studying the development of players' CT skills, especially problem decomposition, pattern recognition, abstraction, and algorithm design.

Learn more

Designing Biomimetic Robots

Designing Biomimetic Robots project will develop and study an education program that integrates science, engineering, and computing by engaging students in biomimicry design challenges. In these challenges, students will first study the natural world to learn how animals and plants accomplish different tasks. Then, they will engineer a robot that is inspired by what they learned.

Learn more