Promising Practice for Integrating Computational Thinking Across the K-8 Curriculum

Irene Lee, Santa Fe Institute
Katie Apone, Code.org
CSTA – July 15, 2014
Outline of today’s presentation

Part I
• The 21st Century Workplace / New Skills needed
• What is Computational Thinking (CT)?

Part II
• Why CT in K-8?
• Supporting CT in K-8 with example activities
• Promising Practices
• Cross curricular integration of stories and modeling and simulation.

Part III
• Discussion: potential barriers and possible solutions
• Discussion: sharing ideas for cross-curricular integrations
Figure 3: Index of Changing Work Tasks in the U.S. Economy 1960-2009

Dancing with Robots - Human Skills for Computerized Work, Levy and Murnane, 2013
Computational Thinking
(Wing 2006, 2008)

- Skills, habits and approaches **integral to solving problems using a computer**
- **Thinking patterns** that involve systematically and efficiently processing information and tasks.
- Reasoning at multiple levels of abstraction; Understanding and applying automation; Understanding dimensions of scale
• The underlying idea in computational thinking is developing models and simulations of problems that one is trying to study and solve.
Three Pillars of CT
(Cuny, Snyder, Wing 2010)

• **Abstraction** stripping down a problem to its bare essentials and/or capturing common characteristics or actions into one set that can be used to represent all other instances.

• **Automation** using a computer as a labor saving device that executes repetitive tasks quickly and efficiently.

• **Analysis** validating if the abstractions made were correct.
### Computational Thinking (Wing 2010)

<table>
<thead>
<tr>
<th>IS</th>
<th>IS NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptualizing</td>
<td>(Only) Computer programming</td>
</tr>
<tr>
<td>Fundamental</td>
<td>Rote skills</td>
</tr>
<tr>
<td>A way humans think</td>
<td>A way that computers think</td>
</tr>
<tr>
<td>Complements and combines</td>
<td>Only mathematical and engineering thinking</td>
</tr>
<tr>
<td>mathematical and engineering</td>
<td></td>
</tr>
<tr>
<td>thinking</td>
<td></td>
</tr>
<tr>
<td>Ideas</td>
<td>Artifacts</td>
</tr>
<tr>
<td>For everyone, everywhere</td>
<td>Only programming, computer science jobs</td>
</tr>
</tbody>
</table>
CT is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution
Computational Thinking is a human thought processes

Computational tools and resources
- Models and simulations
- Computing resources (CPUs)
- Computer languages and environments

Computational Thinking:
- Abstraction
- Automation
- Analysis
### Table 1: Examples of CT in Three Domains

<table>
<thead>
<tr>
<th>Domain</th>
<th>Abstraction</th>
<th>Automation</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modeling &amp; Simulation</td>
<td>Selecting features of real-world to incorporate in a model</td>
<td>Time stepping using a model as an experimental testbed</td>
<td>Were the correct abstractions made? Does the model reflect reality?</td>
</tr>
<tr>
<td>Robotics</td>
<td>Design robot to react to a set of conditions</td>
<td>Program checks sensors to monitor conditions</td>
<td>Are there situations that were not taken into account?</td>
</tr>
<tr>
<td>Game Design &amp; Development</td>
<td>Games are abstracted into a set of scenes containing characters</td>
<td>Game responds to user actions</td>
<td>Do the elements incorporated make the game fun to play?</td>
</tr>
</tbody>
</table>
The potential

- While teaching computational thinking may result in more computer science degrees, the more important contribution it will make is giving more people across all fields the ability to solve problems like a computer scientist and to speak the language of computer programming.

Supporting growth in CT

• The first is the use of rich computational environments. Rich computational environments are ones in which the underlying abstractions and mechanisms can be inspected, manipulated and customized.
Rich Computational Tools

**Scratch** *(scratch.mit.edu)*
Create stories, games, and animations.
Share with others around the world.

Remix animations and learn to create your own.
Rich Computational Tools

Blockly (https://code.google.com/p/blockly/ and learn.code.org)
Solve puzzles, create stories, games, and animations.
Rich Computational Tools

**StarLogo Nova (slnova.org)**
Allows exploration of emergent and complex systems

Users create simulations by writing simple rules for individual “agents”

No sophisticated mathematics or advanced programming skills are required
App Inventor (appinventor.mit.edu)
Allows creation of interactive apps.

Users create apps by constructing programs from command blocks.
Why blocks?

Familiar metaphor for young students (blocks that snap together.)

Reduces syntax errors:
  - due to typing errors of var names
  - due to syntactical errors “e.g. [ ]”.

Automatic type checking

Color cues

Exposure to core concepts

Common command blocks
  (anecdotal evidence of transference)
Use-Modify-Create trajectory

Figure 5: Use-Modify-Create Learning Progression

"Not Mine"

"Mine"

Test

Refine

Analyze
Use-Modify-Create progression

USE

MODIFY

CREATE

“Not Mine”

“Mine”

Refine → Test → Analyze

→ Refine → Test → Analyze
DEMOS
Puzzles to Stories

**Code.org K-5 curriculum**

Learn.code.org/s/course1
Learn.code.org/s/course2
Learn.code.org/s/course3
Scientific Investigations

Modeling and Simulation

Project GUTS afterschool curriculum
  Ecosystems, Epidemics, Emergency Egress
  Social systems: traffic, networks, shared resources, cooperation.

Project GUTS Code.org CS in Science curriculum
  Earth, Life and Physical Science
Cross curricular integration

- Science
- Social science
- Math
- Language arts
- Arts and music

- Integrating cross disciplinary projects into CS classes
Discussion in small groups...

1. What are you already doing to integrate CT? OR, Where could integrate CT?

2. What barriers have you encountered or feared?

3. What potential solutions can you share?

4. How would you explain CT to someone else? What’s your elevator speech to sell CT to others?

5. What does a classroom that fully integrates CT look and feel like?
Recap of today’s presentation

Part I
• The 21st Century Workplace / New Skills needed
• What is Computational Thinking (CT)?

Part II
• Why CT in K-8?
• Supporting CT in K-8 with example activities
• Promising Practices
• Cross curricular integration of stories and modeling and simulation.

Part III
• Discussion: potential barriers and possible solutions
• Discussion: sharing ideas for cross-curricular integrations
Resources

• CSTA Computational Thinking resources  
  http://csta.acm.org/Curriculum/sub/CompThinking.html
• Scratch  http://scratch.mit.edu
• Blockly  https://code.google.com/p/blockly
• Code.org  http://learn.code.org
• StarLogo Nova  http://slnova.org
• App Inventor  http://appinventor.mit.edu
• Project GUTS  http://projectguts.org
  also see http://code.org/curriculum/mss
For more information:

Irene Lee, Santa Fe Institute
lee@santafe.edu

Katie Apone, code.org
katie@code.org

With support from the National Science Foundation