# Instructional Strategies for Abstraction in K-12 Computer Science

Christine Liebe, PhD Candidate, Walden University
Education: Curriculum, Instruction, Assessment
Adjunct Professor, Colorado Mountain College (cliebe@coloradomtn.edu)
Christineliebe.com
christine@christineliebe.com

Lecture Notes
7.17

1. Welcome! This presentation was intended to present research on my dissertation research, a qualitative examination of the instruction of abstraction in K-12 computer science (CS). However, because I am still revising my proposal, I will share instructional implications from my review of current literature. I am a licensed K-12 teacher in Colorado with endorsements in elementary education and secondary Science and Language Arts. I have trained as an AP Computer Science instructor, designed college level intro CS courses in AppInventor and Unity. Currently, I am learning SQL and R focusing on improving my data analysis skills.

2. The College Board has proposed abstraction as one of the 7 "big ideas" in Advanced Placement (AP) courses necessary for CS mastery. Wing (2008), author of computational thinking, and Brennan & Resnick (2011), MIT educational CS researchers, also incorporate abstraction in their conceptual frameworks for computer science.

Big Idea 1: Creativity. Computing is a creative activity. ...
Big Idea 2: Abstraction. ...
Big Idea 3: Data and Information. ...
Big Idea 4: Algorithms. ...
Big Idea 5: Programming. ...
Big Idea 6: The Internet. ...
Big Idea 7: Global Impact.

3. Education is a longitudinal process, a conglomeration of many experiences and interactions. The teacher provides input and the student creates output. I found 2 studies in which researchers studied abstraction in computer science (high school and college student surveys). I did not find one study out of several hundred relevant articles, including computational thinking, problem-solving, computer science teaching, and more key words that focused on the instruction of abstraction. Because so little research exists on computer science instruction and less on the instruction of abstraction, I decided to conduct a qualitative study to gather basic information to help inform professional development efforts, CS instructional practices, and develop ideas for future quantitative research. My specific research questions are:

a. What types of instruction do K-12 teachers find most effective for teaching abstraction in CS?
b. How do teachers determine objectives and outcomes for abstraction?
c. How do teachers assess CS abstraction skills and knowledge?

4. My bias: I think that computer science should be taught to all kids, so that people can learn to express their creativity through technology. Standards, also called competencies, help teachers organize course objectives, then designate learning outcomes, and offer individualized instruction.

5. You have examples of coding artifacts on your tables. If you are accessing this presentation online, you can view the examples at the end of the slides. Please share with a friend. What is your definition of abstraction in CS? Which lines of code from your samples provide evidence of abstraction?

6. Some coding procedures that provide evidence of abstraction include: Recursion, Constructing variables, Using event handlers, Remixing code, creating unique renditions of programs, creating new programs
This list of procedures is a start, and there are surely more. The procedures are somewhat subjective and hard to assess objectively.
(Brennan & Resnick, 2012; Wing, 2006; Colburn, 2000; Lee, 2010)

7. Abstract is a noun, a verb, and an adjective which doesn't help teachers. Abstraction is a process. It is easy to get confused and think that if students can think about non-concrete solutions, they can do abstraction. Imagination is part of the process of abstraction.

8. Abstraction occurs at many levels of hardware, software….

9. During the human computer interaction, or relationship, as an inforg, the expression of human creativity facilitates epistemological development. Just as writing helps us to see our thoughts, computers also help us to express our knowledge.

10. Abstraction is a soft and hard concept, requiring conceptual understanding and procedural skill. According to (Hazzan, Lapidot, Ragonis, 2015), soft concepts are not easy to explain, are not connected to an exclusive topic, but are expressed throughout the curriculum. Soft concepts must be "sensed" and require context for understanding. They metaphorically are represented by SOFT-ware. Hard concepts are factually based and have rigid rules, metaphorically represented by HARD-ware.

11. An example of variables as both a soft and hard concept.

12. The Perrenet, Kassenbrood, and Groot (2005), or PKG Hierarchy provides another framework for teaching and evaluating the instruction of abstraction. The execution level with the algorithm and the machine. The program level, i.e. taking code

from one program and molding it into another. Then at the object level, people perceive a program or an algorithm as a thing rather than the complex processes they are. Finally, abstraction takes place on the problem level when people deductively pose a solution, such as an iPhone, or a Mcirobit.

13. Abstraction requires both deduction and induction. Marzano & Kendall (2007) call abstraction retroduction. Many teachers know how to teach critical thinking, induction and deduction.

14. For instance, questions are instructional tools that stimulate metacognition. Festo (2016) noted that question taxonomies for Chemistry include recall, algorithmic, and conceptual questions. Recall questions help students access declarative knowledge; algorithmic questions– procedural knowledge; and conceptual questions– applied knowledge.

15. 16. 17. Fuller, et. al., 2007, proposed taxonomy for learning computer science. The researchers described a variety of pathways for learning computer science. Students may only attain theoretical knowledge. Students may only attain practical knowledge. Students can take a variety of pathways to learn how to create operational computer science applications.

18. Some DON'T's for teaching abstraction include: Get confused with concrete and abstract as adjectives.
Don't reinvent the wheel. Teachers know a lot about how to teach. Access their background knowledge.
Don't rely on software to teach. Humans need to teach humans how to creatively express themselves with technology.
Don't get overwhelmed. Many small steps lead to a long journey.

19. DO instructional suggestions from CS, Math, & Science research include:
Teach to declarative and procedural knowledge of abstraction (Marzano & Kendall, 2007).
Offer students critical thinking and problem-solving activities to practice both induction and deduction.
Recognize levels of abstraction, such as PKG Hierarchy.
Explore human the computer relationship with your students.
Teach in microworlds – Weintraub & Wellensky, 2014
Use question taxonomies to develop critical thinking and metacognition (Festo).
Use open-ended questions to facilitate problem-solving skills (Promraksa, Sangaroon, & Inprasitha, 2014).
Use collaboration & context (Vygotsky / Papert) – propagation of memes, peer programming, and peer-tutoring.
Remember there are a variety of pathways to achieving mastery (Fuller et.al., 2007).
Distinguish between students who need pedagogy and andragogy –guided scaffolding or constructionism (inquiry based learning).

Allow students to be active and reflective learners, doers and watchers (especially ELLs and reluctant students), and processing time outside the classroom.
Cross-reference abstraction in Art (impressionism), Literature (poetry, outlines), Science (Chemistry), Math (Algebra).

Please see my dissertation proposal for more instructional implications at christineliebe.com.

## Presentation Bibliography

Abelson, H., Ledeen, K., & Lewis, H. R. (2008). *Blown to bits: Your life, liberty, and happiness after the digital explosion*. Upper Saddle River, NJ: Addison-Wesley.

Armoni, M. (2013). On teaching abstraction in Computer Science to novices. *Journal of Computers in Mathematics and Science Teaching*. *(32)* 265-284.

Brenan, K., Resnick, M. (AERA, 2012). *New frameworks for evaluating and discussing the development of computational thinking*. White paper. MIT Medial Lab.

Colburn, T., & Shute, G. (2007). Abstraction in computer science. *Minds and Machines*, *17*(2), 169-184.

CSTA (2015). Computer science education research. Retrieved from http://csta.acm.org/Research/sub/KeyResearch.html

Festo, K. (2016). Question Classification Taxonomies as Guides to Formulating Questions for Use in Chemistry Classrooms. *European Journal of Science and Mathematics Education*, *4*(3), 353-364.

Fuller, U., Johnson, C., Ahoniemi, T. et al (2007). Developing a computer science specific learning taxonomy. ITiCSE working group report on innovation and technology in computer science education.  doi: 10.1145/1345443.1345438

Floridi, L. (2011). *The philosophy of information*. Oxford University Press.

Hazzan, O., Lapidot, T., & Ragonis, N. (2015). *Guide to teaching computer science: An activity-based approach*. London, UK: Springer.

Lee, Y. J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia*, *19*(3), 307-326.

Marzano, R. J., & Kendall, J. S. (Eds.). (2006). *The new taxonomy of educational objectives*. Corwin Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Harper Collins.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725.