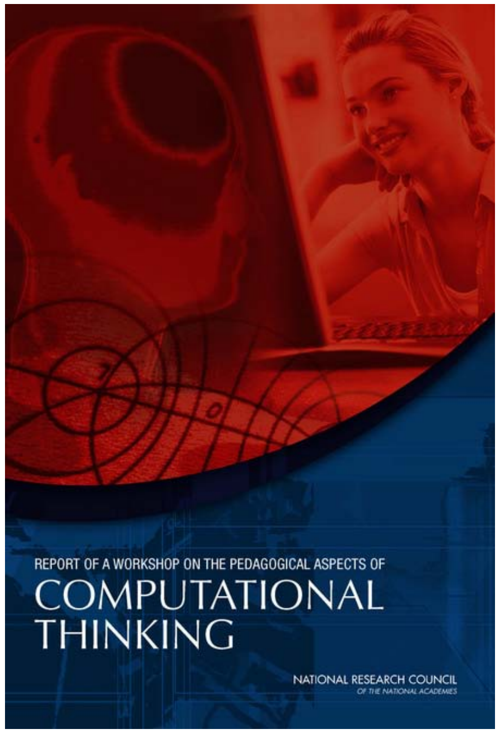
Computational Thinking

NRC Report on Pedagogy

web site: www.cs.vt.edu/~kafura/CS6604

NRC Report on Pedagogy for CT

- Second of two workshops
- Focused on K12 Education
- Identified different approaches to the teaching of computational thinking
- What do these approaches and ideas mean for the university level?


REPORT OF A WORKSHOP ON THE PEDAGOGICAL ASPECTS OF
COMPUTATIONAL THINKING
NATIONAL RESEARCH COUNCIL
OF THE NATIONAL ACADEMIES

Unless otherwise noted all quotations are from this report.
Page numbers are those in the report not those in the PDF document.

Questions

- What are the relevant lessons learned and **best practices** for improving computational thinking in K-12 education?
- What are some **examples** of computational thinking and how, if at all, does computational thinking vary by discipline at the K-12 level?
- What **exposures and experience**s contribute to developing computational thinking in the disciplines?
- What are some innovative **environments** for teaching computational thinking?
- Is there a **progression** of computational thinking concepts in K-12
- education? What are some criteria by which to order such a progression?
- How should **professional development** efforts and classroom support be adapted to the varying experience levels of teachers such as pre-service, inducted, and in-service levels?
- What **tools** are available to support teachers as they teach computational thinking?
- How does computational thinking education **connect with other subjects**? Should computational thinking be integrated in other subjects taught in the classroom?
- How can learning of computational thinking be **assessed**? How should we measure the success of efforts to teach computational thinking?

Need for Definition

- "…adopting a consistent definition of computational thinking is necessary because people see computational thinking through only their own lenses—and efforts to advocate for computational thinking in the curriculum will not be credible in the absence of consensus about its structure and content." [p3]

K12 Context

- Observations attributed to Jeannette Wing [p4-5]
  - Math education has a long history of defining learning progressions based on human development
  - Computational education in K12
    - Lacks a clear plan of progressions
    - Belief that abstract concepts of computing cannot be learned until late in K12 (8th grade) because of the highly symbolic/abstract nature of computation
    - Belief has not been subjected to rigorous study

Perspectives

- "…computational thinking [is] generalized problem solving with constraints." [p6]
- "…core of computational thinking is to break big problems into smaller problems that lend themselves to efficient automated solutions." [p8]
- "…the ability to construct rules to specify the behavior of an agent is important to computational thinking." [p8]
- Using "computational media to create, build, and invent solutions to problems is central to computational thinking." [p8]
- "systems thinking is an essential activity in computational thinking." [p 9]
- "understanding complex systems requires computational thinking." [p9]
- Common points
  - relationship to problem solving
  - constraints come from need for solution to be automatable
  - activities are constructive and involve computational elements as first-order entities, not merely using compute-based tools
  - view world as collection of interacting parts, manage complexity, break problems down

Role of teams/groups

- “…students need a way to design solutions that are rich enough to cope with complexity and interactivity in a manner often associated with computational expression. And the design environment needs to support social cooperation in constructing meaningful expressions.” [p 8]
- “…most students find programming in pairs highly motivating.” [p 9]

Role of context

- "…the power of computational thinking is best realized in conjunction with some domain-specific content." [p 9]
  - Opportunities in the science and social science domains
- "Developing expertise in computational thinking involves learning to recognize its application and use across domains." [p 10]
- How to resolve this tension?
- What does this mean in terms of progressions?
- What does this mean for a university curriculum?

Advantages/disadvantages of Context

- Linn [p 49]
  - "We could call for emphasizing computational thinking everywhere and end up finding that it is nowhere because no one felt responsible for it. In addition, even if we did incorporate computational thinking into every course we might fail to build competence because the experiences were not cumulative. We need to think about ways to build coherent understanding of computational thinking as students encounter it across disciplines."
- Wilensky argued that computational thinking is important enough that it should not have to be squeezed in on the margins or sneaked in on the side. [p 43]
- Resnick [p 68] : argued, most people work better on things they care about and that are meaningful to them, and so embedding the study of abstraction in concrete activity helps to make it meaningful and understandable.

Declarative vs. Procedural Knowledge

- "… often typical instruction is oriented toward declarative knowledge, whereas computational thinking is oriented toward procedural knowledge. In this view, declarative knowledge provides content (and is essential to particular fields or careers), whereas computational thinking is most useful for integrating and building connections in the midst of such knowledge." [p 30]
- Offers a rationale for embedding study of computational thinking into domain-specific classes
- Puts at risk the explicit recognition, understanding and adoption of computational thinking itself outside of the context in which it was seen

Context and Transfer Learning

- Kolodner 's argument[p 57]
- "...it is important not to fall prey to the mistaken notion that if one learns computational thinking skills in one context, one will automatically be able to use them in another context.
- "Rather, it will be important to remember that one can learn to use computational thinking skills across contexts only if
  - (1) the skills are practiced across contexts,
  - (2) their use is identified and articulated in each context,
  - (3) their use is compared and contrasted across situations, and
  - (4) learners are pushed to anticipate other situations in which they might use the same skills (and how they would)."

Examples of Context

- Everyday life [p 10]
  - Just because you can describe/articulate a situation in computational terms does not necessarily mean that the people involved are using computational thinking
  - Analogies must be in service of deeper learning
- Games and Gaming[p 10-11]
  - Playing a game is not computational thinking; defining/modifying the rules of the game is
  - Provides an opportunity for team work and context-based grounding (a science game)
- Science [p 11]
  - Collecting/graphing data is not computational thinking
  - Using genomic databases or environmental simulation to learn science is not computational thinking
  - Defining the rules of behavior, observing the result and modifying the rules to achieve a goal is computational thinking
  - Defining a representation of a geo-image and realizing the advantages and limitations of that representation is

Context

- Science (cont)
  - Geographic data can be used to illustrate [p13]:
    - Continuous vs. discrete data
      - Implications of each representation
    - Implication of color coded data
      - The underlying numeric representation (model) is separable from its display (view)
    - Boolean operations (threshold based selection)
    - Spatial relationships
    - Multiple constraint satisfaction
- Engineering
  - Connections between science and engineering is not computational thinking
- Journalism
  - Similarities/difference of natural vs. computational languages
  - Relationship between the steps in journalistic editing and computational problems solving is only a surface analogy

Pedagogical Environments

- "…interactive visualizations or simulations are at the heart of computational thinking." [p17]
  - But only if the representation/abstraction and its advantages/limitations are specifically investigated
  - Just using visualization/simulation is not enough
- Modeling/troubleshooting of data sets
  - Student collect data to form and refine a model
    - Comes to grips with abstraction and representation
- Finding patterns in data
  - What are the computational thinking ideas?

Progressions

- Lack of "developmentally appropriate definition of computational thinking." [p 46]
- Kafai: "…we really do need a more profound understanding of what kids' engagement with computational thinking at different ages is, and then how we can kind of build pedagogies, examples, on it." [p 46]
- Is there a need to think about a learning progression at the university level?
  - How broadly across the curriculum is CT embedded?

Paradigm

- Use-modify-create paradigm [p25]
  - Steps
    - Use a model
    - Adjust model controllers (sliders) to see effect
    - Add new controls
    - Develop a model and its controllers
  - Learning moves from passive use of computational tools to active use of computational thinking skills
- Wilensky's alternative : "...creating" in small bites as well, and sometimes creation is a lot easier than modifying as a different kind of entry point, and all of the outcomes are ones that we want." [p 45]

Example progression

- Tinker's example [p 67]
  1. There are numeric values associated with every object and their interactions.
  2. These values change over time.
  3. These changes can be modeled.
  4. Models involve lots of simple steps defined by simple rules (e.g., the molecular dance).
  5. Models can be tested to find their range of applicability.
  6. You can make models.
  7. Many other applications of computers share the same features.

Assessment

- "…narrow goals for evaluation are counterproductive…need to appreciate the links among topics and goals for courses." [p 26]
- Kolodner: [p 60]
  - …one has to apply an entire toolbox of assessment and evaluation tools"
  - Indicators
    - Student's reflection on a computational activity
    - Being able to teach/help someone else learn the concept
    - Being able to effectively articulate the relevant computational process at issue
- Aho: determining what students are learning in computational thinking activities may be difficult. He noted that assessing how a student has internalized the abstractions of computational thinking may be challenging, and even assessing programming skills can be difficult.
- How is assessment to be done at the university level?

Purposes for assessment

- " In addition to knowing what one wants to assess, one must consider the purpose of the assessment, because the reason for any assessment plays a critical role in determining the data and process necessary to perform it." [p 61]
- Possible goals:
  - to judge the curriculum and related materials and pedagogy,
  - to judge the progress of individuals, e.g., for giving grades, and
  - to manage instructor training and support.

Assessment vs. Evaluation

- Assessment
  - What have students learned
  - How they feel about something
  - Capabilities
  - Kinds
    - Formative
    - Summative
- Evaluation
  - How well a curriculum or software component is working
    - efficacy
    - cost
    - usability

References

- [NRC 2010] *Report of a Workshop on the Scope and Nature of Computational Thinking*. 2010, National Research Council.
- [NRC 2011] *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. 2011, National Research Council.