Considering Computational Thinking, Culture & Assessment

Leveraging Evidence-Centered Design to Develop Authentic Assessments of Computational Thinking Practices

ISDDE 2018 | Galway, Ireland

Eric Snow, Daisy Rutstein, Satabdi Basu | SRI International

Overview

- Considering Computational Thinking (CT)
- Assessment Design & Culture
- Formalizing the Consideration of Culture and CT in Assessment Design
- Applications of Evidence-Centered Design (ECD)

Considering Computational Thinking (CT)

Problem-solving skills or behaviors that students demonstrate to fully engage in today's data-rich and interconnected world.

Foundational problem-solving skills applied in a computational context.

Includes a basic understanding of concepts such as computers, computation, data, information, devices, and the like.

Increasingly recognized as an essential form of literacy for all informed citizens in the modern world, not just computer scientists.

Computational Thinking Practices

Application of fundamental CS conceptual knowledge and programming skills in authentic computational problem-solving contexts, e.g.,

Designing and applying abstraction and models

Designing and developing computational artifacts

Analyzing the effects of developments in computing

Automating solutions through algorithmic thinking

Analyzing one's own computational work and the work of others

Considering Culture

Systems of meaning & how people use those systems to act in the world

One way people make sense of their worlds is through the representations they see and think with

Assessment Design & Culture

Assessments elicit evidence of what students know and can do through using different representations and contexts

Representations and contexts need to be *authentic*, or consistent with those that students use in their worlds to make meaning

Assessment, Authenticity & Design Under Constraints

Authenticity in assessment often manifests as student choice, multiple representations and open-ended formats

Assessment is also an art of *design under constraints*, many of which challenge principles of authenticity

Questions

How do we formalize the consideration of culture and CT in assessment design?

How do we address constraints when designing for authenticity?

What representations and contexts do students use to meaningfully engage in CT?

Evidence-Centered Design (ECD)

ECD is a framework for assessment design and development.

- Views assessment as a process of gathering evidence to support an argument about what a student knows and can do.
- Provides a structure for an approach that incorporates validity evidence into the assessment design process.
- Particularly useful when the knowledge/skills to be measured involve complex, multistep performances, such as those required in computational thinking.

ECD Layers	Activity
Domain Analysis	Gather substantive information about the domain of interest (e.g., computational thinking practices) that has implications for assessment; how knowledge is constructed, acquired, used, and communicated; identify representations and contexts that are meaningful given the domain
Domain Modeling	Express information from domain analysis in narrative form as an assessment argument: • knowledge and skills to be assessed • kinds of tasks/situations that elicit performances • features of performances that convey evidence
Conceptual Assessment Framework	Further specify assessment argument in structures and other details for tasks and tests, evaluation procedures, and measurement models
Assessment Implementation	Implement assessment, including presentation-ready tasks and calibrated measurement models
Assessment Delivery	Coordinate interactions of students and tasks: task- and test-level scoring; reporting

More information about ECD can be reviewed at:

https://ecd.sri.com/

Applications of ECD

Principled Assessment of Computational Thinking (PACT)

CoolThink@JC

Principled Assessment of Computational Thinking

ECS | Exploring Computer Science

Principled Assessment of Computational Thinking (PACT)

Create design templates for supporting CT assessment development for secondary CS.

>>> *CT Design Patterns*

Design, develop, validate assessments of computational thinking practices aligned with the Exploring Computer Science (ECS) curriculum.

Assessment Design Patterns

Specifies and organizes assessment information in narrative form based on information from Domain Analysis

High-level representation of assessment information

Links specialized knowledge and skills in a domain with the specialized knowledge about the more technical machinery of assessment

PACT CTP Design Patterns

Analyze the effects of developments in computing

Design and implement creative computational solutions and artifacts

Design and apply abstractions and models

Analyze computational work (both own and others)

Communicating computational thought processes, procedures and results to others

Collaborate with peers on computing activities

These design patterns are not aligned with a specific curriculum, but with general practices in the CT domain.

Design and Apply Abstractions and Models

Overview

Thinking strategically about abstraction is a hallmark of computational thinking. This design pattern supports the development of tasks in which students use ideas and representations that capture general to specific aspects, or patterns, of an entity or a process and the relationships/structures among entities or processes, including level of detail. This may include designing general solutions to problems or generalizing a specific solution to encompass a broader class of problems (functional abstraction). These ideas and representations may be used in different contexts (problem or disciplines). Students demonstrate knowledge of the representational properties of discrete mathematics, models, diagrams, computer programs (data abstraction), items found in the natural and man-made world, and others. They also demonstrate an understanding of the limitations of models to represent phenomena and attention to the purpose of the model or abstraction.

Focal Knowledge, Skills, and Attributes

Foundational Knowledge Involving Abstraction in Computing

1. Ability to explain what abstraction is, both functional and data.
2. Ability to reason about a problem at multiple levels of detail.
3. Ability to explain the benefits of using abstraction in problem solving, e.g., to manage complexity and generalize patterns.
4. Ability to explain that an algorithm is a form of abstraction that contains a sequence of instructions whose end state or output can be determined once given a particular starting state.
5. Ability to explain the characteristics of problems for which abstraction would be useful.
6. Ability to describe how a computer model makes a representation of the real world.
7. Ability to explain how computers represent mathematical objects and logical operations for purposes of computation and modeling.
8. Ability to explain how computers represent objects as data and data as objects (e.g., media files, QR codes).
9. Ability to explain the connections between elements of mathematics and computer science including binary numbers, logic, sets, and functions.

Learn more about the Computational Thinking Practice Design Patterns:

http://bit.ly/2u6t0Nw

ECS Design Patterns

Unit 1:
Human-computer interaction

Unit 2:
Problem solving

Unit 3:
Web design

Unit 4:
Introduction to programming

These design patterns are aligned with learning objectives in the ECS curriculum.

4. **For a class assignment, Gabriela and Lucia both created an algorithm that has a dog run laps on the screen.**

Gabriela's algorithm	Lucia's algorithm
• **Step 1:** Ask how many laps the dog should run. Go to Step 2. • **Step 2:** Check the number entered. ▪ **Step 2a:** If the number entered is less than 2, then the dog says "Not enough laps." Then skip to Step 5. ▪ **Step 2b:** If the number entered is greater than 200, then the dog says "Too many laps." Then skip to Step 5. ▪ **Step 2c:** If the number entered is greater than or equal to 2 AND less than 100, then skip to Step 3. ▪ **Step 2d:** If the number entered is between 100 and 200 (including 100 and 200), then skip to Step 4. • **Step 3:** The dog runs the number of laps entered. Skip to Step 5. • **Step 4:** The dog runs **half** of the number of laps entered. Go to Step 5. • **Step 5:** The program ends.	• **Step 1:** Ask how many laps the dog should run. Go to Step 2 • **Step 2:** Check the number entered. ▪ **Step 2a:** If the number entered is less than 2, then the dog says "Not enough laps." Then skip to Step 4. ▪ **Step 2b:** If the number entered is NOT less than 2, then move to Step 3. • **Step 3:** The dog runs the number of laps entered. Go to Step 4. • **Step 4:** The program ends.

a) In Gabriela's algorithm, what is shown on the screen if the number entered is 120?

- ☐ Dog says "Not enough laps."
- ☐ Dog says "Too many laps."
- ☐ Dog runs 120 laps.
- ☐ Dog runs 60 laps.

b) In Lucia's algorithm, what is shown on the screen if the number entered is 120?

- ☐ Dog says "Not enough laps."
- ☐ Dog says "Too many laps."
- ☐ Dog runs 120 laps.
- ☐ Dog runs 60 laps.

You are the programmer for **Lucia's algorithm**.

c) Which step(s) in Lucia's algorithm would you program using the *If* structure block? Select all that apply.

If structure block:

- ☐ Step 1
- ☐ Step 2a
- ☐ Step 2b
- ☐ Step 3
- ☐ None of the steps

d) Which step(s) in Lucia's algorithm would you program using the *Repeat Until* structure block? Select all that apply.

Repeat Until structure block:

- ☐ Step 1
- ☐ Step 2a
- ☐ Step 2b
- ☐ Step 3
- ☐ None of the steps

More information about PACT can be reviewed at:

https://pact.sri.com/

香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
同心同步同進 RIDING HIGH TOGETHER

CoolThink@JC
賽馬會運算思維教育

SRI Education™
A DIVISION OF SRI INTERNATIONAL

CoolThink@JC

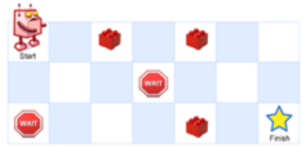
Bringing computational thinking to Hong Kong students in Primary 4-6

Target outcomes based on Brennan and Resnick (2012) CT Framework:

>>> *CT Concepts, CT Practices, CT Perspectives*

SRI used ECD and the PACT CTP design patterns to develop student assessments aligned with these outcomes and the pilot lessons

A robot has to travel from the 'Start' square to the 'Finish' square. During each step, the robot can move to the square directly up, down, left or right, if such a square exists. Every time the robot encounters a red block on a square, there is a fine of $5. Each step takes the robot 1 minute to cover. However, if the robot moves into a square that has a Wait sign, the next step takes 4 minutes.



Here are 3 possible methods for the robot:

Method 1	Method 2	Method 3
1. Move Right	1. Move Right	1. Move Down
2. Move Right	2. Move Right	2. Move Right
3. Move Right	3. Move Right	3. Move Right
4. Move Right	4. Move Down	4. Move Right
5. Move Right	5. Move Left	5. Move Down
6. Move Right	6. Move Down	6. Move Right
7. Move Down	7. Move Right	7. Move Right
8. Move Down	8. Move Right	8. Move Right

a. Which of the methods will get the robot to the Finish square? ☐ Methods 1 and 2 ☐ Methods 1 and 3 ☐ Methods 2 and 3 ☐ All 3 methods get the robot to the Finish square.	b. Sumi wants the robot to take the fastest route that will reach the 'Finish' square. Which method should Sumi choose? ☐ Method 1 ☐ Method 2 ☐ Method 3 ☐ Any of the 3 methods, they all take the same time
c. Choi wants his robot to take the route that costs the least amount of money that gets to the 'Finish' square. He does not care about the time taken. Which method should Choi choose? ☐ Method 1 ☐ Method 2 ☐ Method 3 ☐ Any of the 3 methods, they all cost the same	d. A competition is organized where the goal is to have the robot move from the 'Start' square to the 'Finish' square in 10 minutes or less by paying $5 or less. Which method can be used to satisfy the goal of the competition? ☐ Method 1 ☐ Method 2 ☐ Method 3 ☐ None of the 3 methods can satisfy the goal of the competition

More information about *CoolThink@JC* can be reviewed at:

https://www.coolthink.hk/en/

and

https://bit.ly/2sepChE

Closing Comments

Representations, contexts, authenticity, & design under constraints…oh my!

ECD is particularly relevant when the knowledge/skills to be measured involve complex, multistep performances, such as those required in computational thinking.

Closing Comments

ECD Design Patterns help formalize the consideration of culture and authenticity in assessment development.

Help designers specify choices and weigh tradeoffs between authenticity and other design constraints.

More research is needed on the types of representations and contexts that are meaningful to students when they engage in CT.

Slides will be available on the PACT Publications page:

https://pact.sri.com/publications.html

THANK YOU!

This work was conducted with support from the National Science Foundation under contract numbers, CNS-1132232, CNS-1240625, and DRL-1418149, and with support from the Hong Kong Jockey Club Charities Trust.

SRI Education

NSF

Principled Assessment of Computational Thinking