



# Designing Low-floor, Meaningful Computational Modeling Experiences for Elementary Students and Teachers

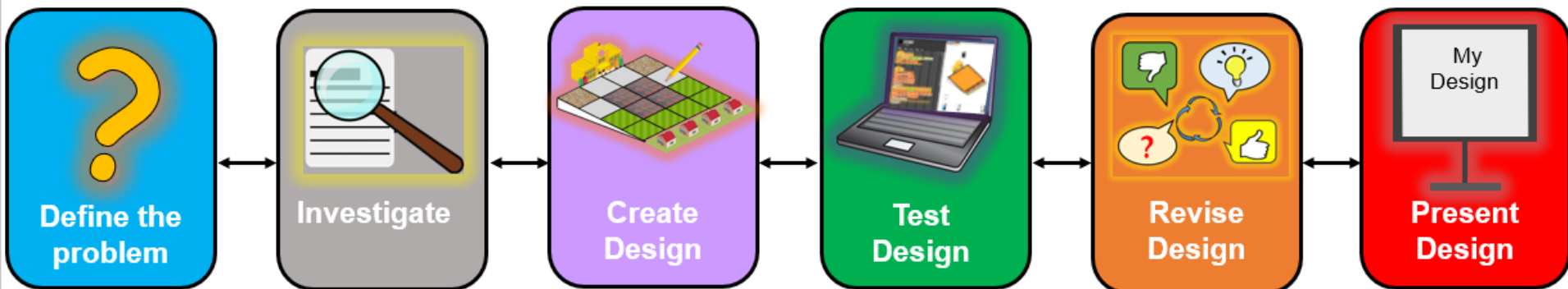
Satabdi Basu, SRI International

Project PIs: Kevin W. McElhaney, Gautam Biswas, Jennifer L. Chiu

STEM+C PI Summit Evidence Sharing Session, Pre-K-5  
September 18, 2019

# Curricular context: Minimizing urban water runoff

- Multi-week, NGSS-aligned, 5th grade curriculum unit integrating earth science, engineering, and computational thinking (NGSS PEs 5-ESS3-1, 3-5-ETS1-3).
- Students develop an engineering solution to minimize the problems caused by urban water runoff at a given school.
- Students develop a computational model that predicts water runoff given total rainfall and surface



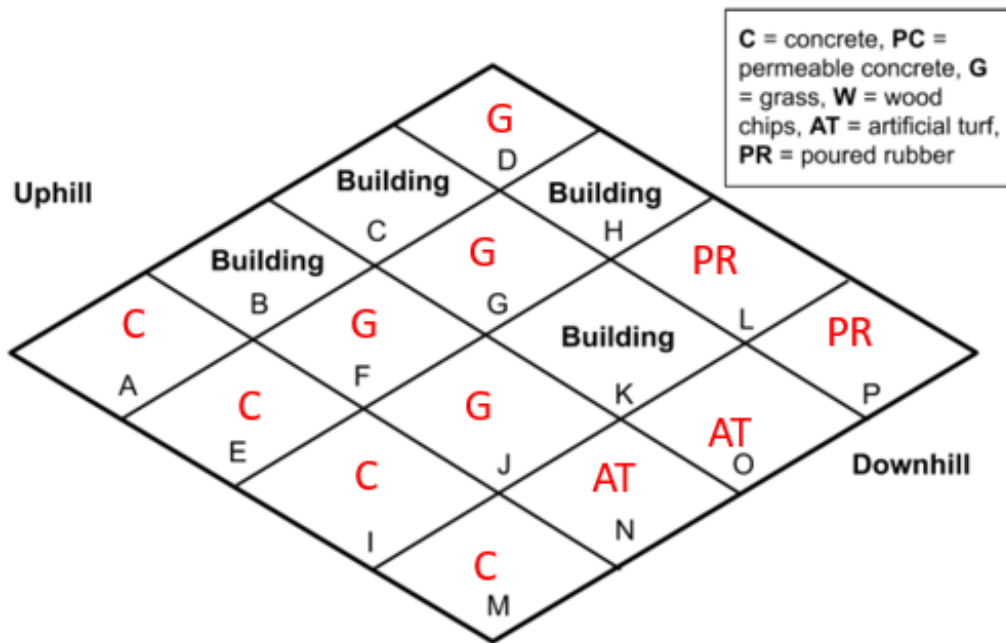
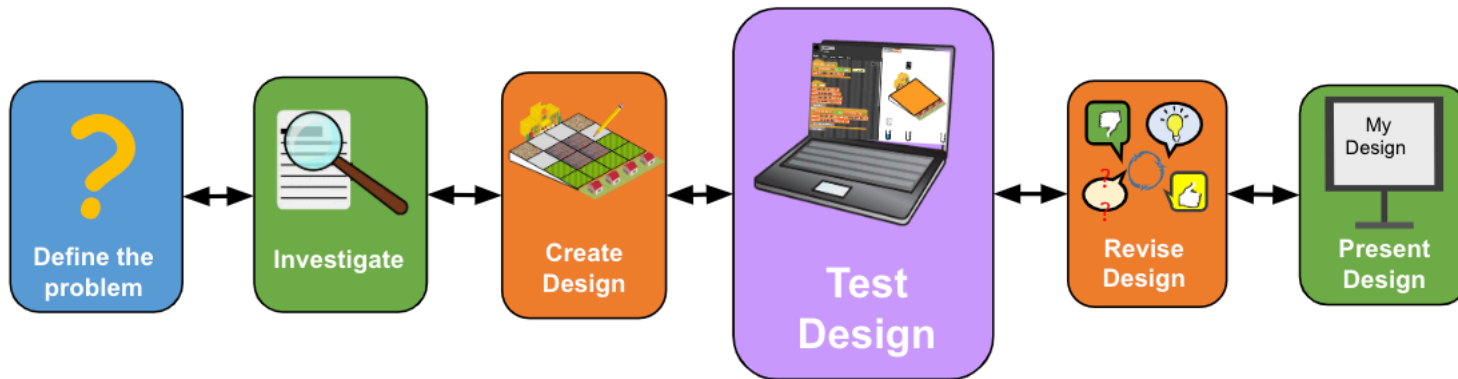
# Designing computational modeling curricular units

- Numerous design considerations, especially for elementary students
  - What incentive do students have for developing a computational model?
  - What aspects of the physical phenomenon will be modeled and what can be abstracted?
  - What modeling representations and computational modeling language should be used?
  - What CS concepts and practices are required and how will students be introduced to them?
  - How will students be supported in testing and debugging their computational models?

# Design guidelines informing computational modeling in SPICE

- Insights and evidence from three pilot studies helped arrive at the 4 design guidelines:
  1. Create an explicit need for computational modeling in the curricular sequence.
  2. Maintain coherence among system representations across science, computing and engineering.
  3. Introduce students to CS concepts required for computational modeling through intentionally designed unplugged activities.
  4. Support systematic testing and debugging of computational model through test cases and debugging scenarios.

# 1. Creating a need for computational modeling



Students create designs on paper



Computational modeling helps test the designs students create and compare different designs by calculating cost and water runoff, without creating physical designs

when clicked ▶ Let's start from here

```

set total rainfall (inch) to 1.2
Set absorption limit of the selected material
if total rainfall (inch) is equal to absorption limit
set total absorption (inch) to absorption limit
set total runoff (inch) to 0
if total rainfall (inch) is greater than absorption limit
set total absorption (inch) to absorption limit
set total runoff (inch) to
total rainfall (inch) - total absorption (inch)
if total rainfall (inch) is less than absorption limit
set total absorption (inch) to total rainfall (inch)
set total runoff (inch) to 0
  
```

current material wood chips

rainfall 1.5 inches  
absorption 1 inches  
runoff 0.5 inches



total cost 1425000

rainfall 2 inches  
absorption 0.88 inches  
runoff 1.13 inches

Students build a computational model for calculating runoff for one grid, and their models are used to calculate runoff for a 16-grid model to help students test their engineering designs

# Benefits of this approach

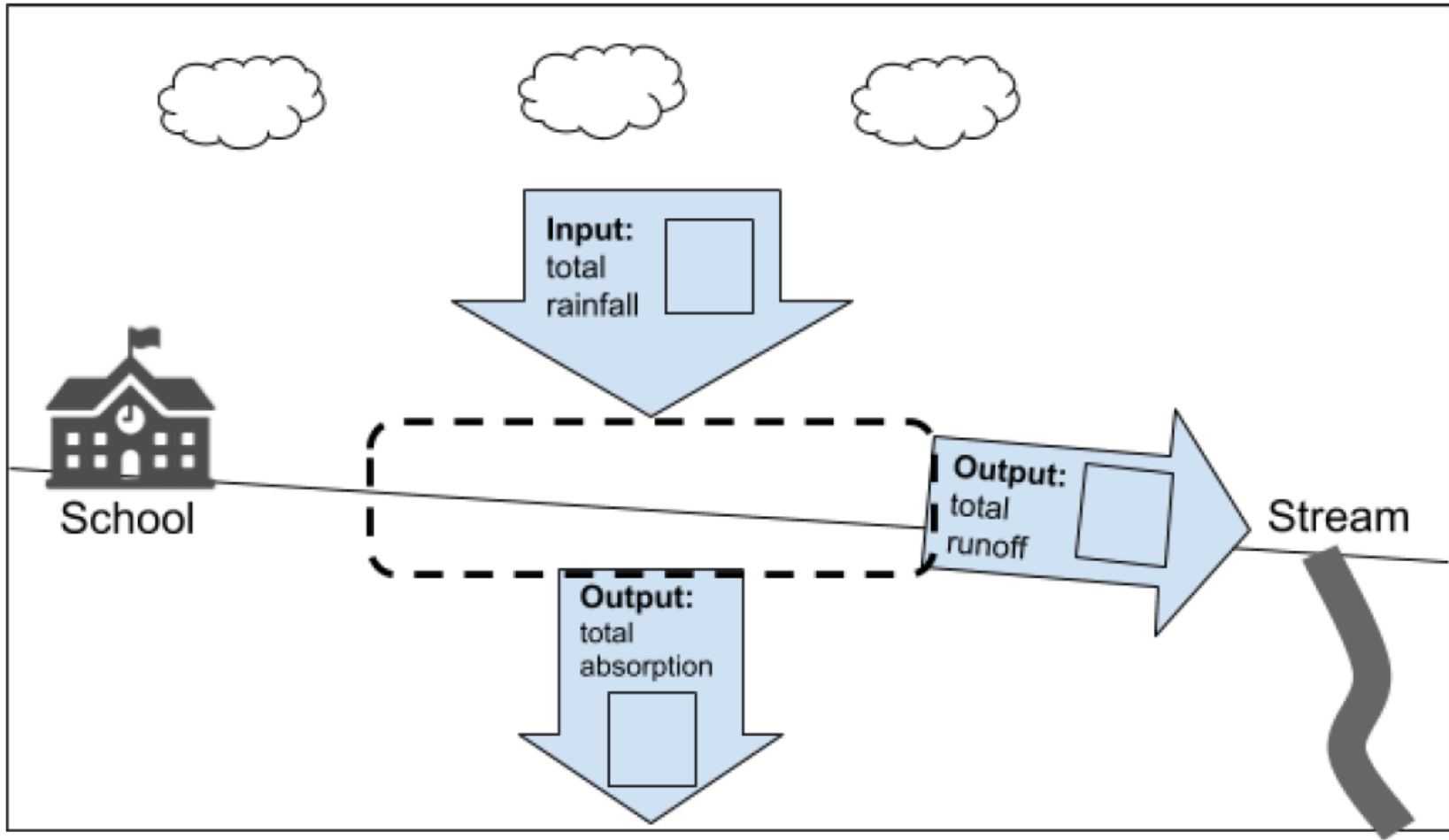
- Giving students an incentive for creating accurate computational models increases their engagement with the activity
- Introduces students to authentic professional practices of scientists and engineers
- Promotes the CS practice of ‘Recognizing and defining computational problems’

## 2. Systems and system models to connect science, computing and engineering

- Prepare students for computational modeling by engaging them in modeling using other representations like text-based descriptive models, pictorial models, mathematical expressions, and pseudocode.
  - Same underlying science content conveyed through all models
  - Consistent terminology across models
- Encourage systems thinking by specifying inputs and outputs across science content, computational representations and comparison of engineering design solutions



# Pictorial runoff model



Total runoff = total rainfall – total absorption (matter conservation)

# Computational runoff model

## Pseudocode:

Input variables: \_\_\_\_\_

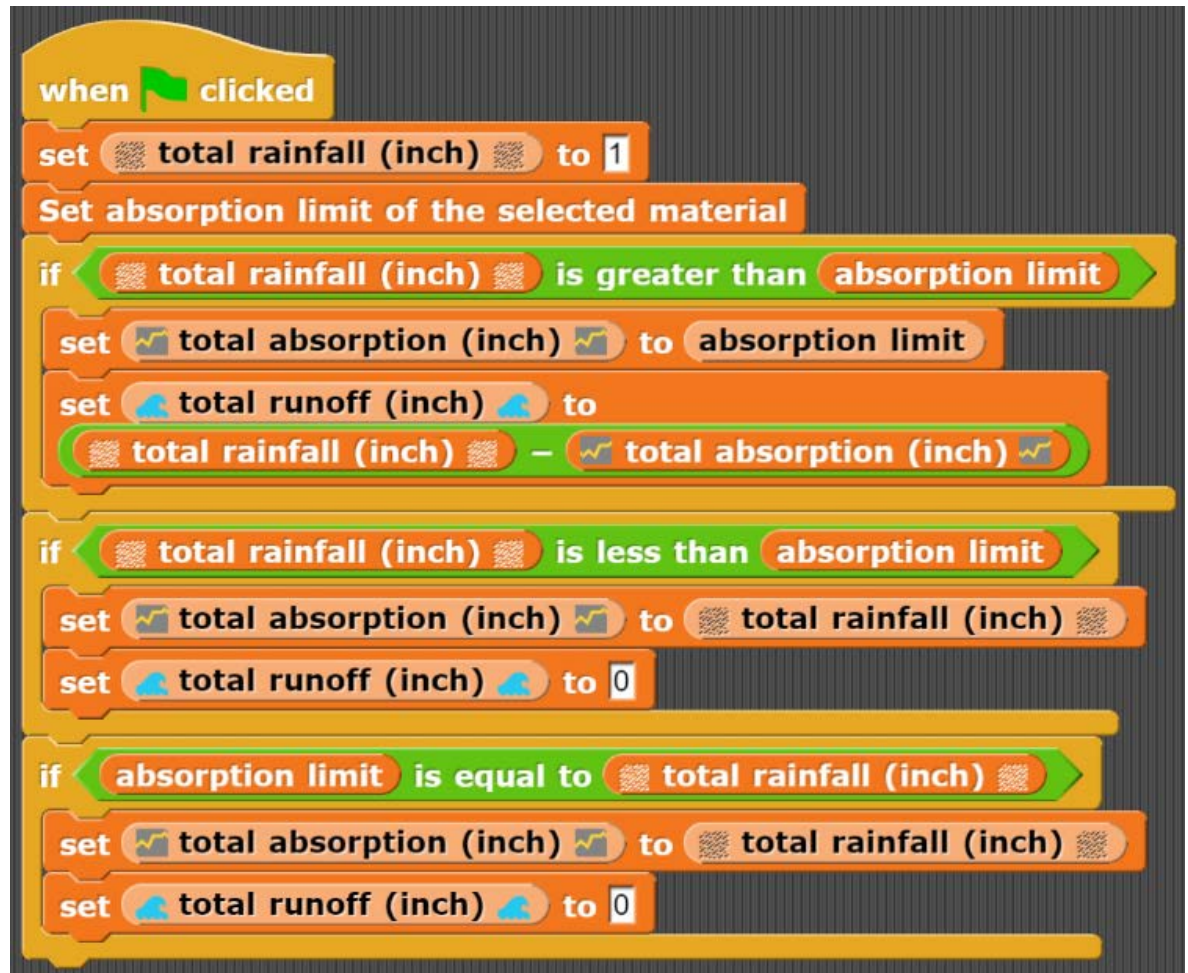
Output variables: \_\_\_\_\_

Rules for calculating the outputs given the inputs:

Rule #1:

Rule #2:

Rule #3:



To manage model complexity as well as to maintain similar content across model representations, we moved away from a time-based computational model to a time-agnostic version

# Testing engineering design

Test your design with your computer model to see how well it meets project criteria.  
Fill the tables below.

Inputs					
Total rainfall (inches)	# Building squares	# "Grassy" squares	# Play squares	# Parking squares	# Accessible squares

Outputs	
Total runoff (inches)	Cost (\$)

### 3. Identifying appropriate CS concepts and developing a domain-specific modeling language

- Introduce just as much CS as needed to model the phenomenon
  - Activities tractable for elementary students and helps lower the barrier to integrating CS for elementary teachers
  - CS concepts limited to:
    - Sequence
    - Variables
    - Arithmetic and logical expressions
    - Conditionals
- Block-based modeling representation chosen with a domain specific modeling language to ensure a low-floor computational modeling experience



# Introducing CS concepts through unplugged activities

- Unplugged activities engage students (and teachers) with required CS concepts in a familiar context
- Activities should mirror how CS concepts will subsequently be used in the model



```
when clicked
  Set values of Dice1 and Dice2 by rolling your dice
  if Dice1 is equal to Dice2
    set Player1_score to Dice1
    set Player2_score to Dice2 + 6
  if Dice1 is less than Dice2
    set Player1_score to 0
    set Player2_score to Dice2 - Dice1
  if Dice1 is greater than Dice2
    set Player1_score to Dice1 + Dice2
    set Player2_score to Dice1 - Dice2
```

Follow along with your class and note the values that are being filled out on the whiteboard in the table below.

Round	Input variables		Output variables	
	Dice1	Dice2	Player1_score	Player2_score
1				
2				
3				
4				
5				


## 4. Support systematic testing and debugging

- Decompose the computational modeling activity into 3 lessons with students modeling 1 rule per lesson
- Support students in designing meaningful test cases for each rule across surface materials
- Include debugging scenarios into modeling activities and embedded assessment activities

# Systematic testing using test cases

Choose wood chips that have an absorption limit of 1 inch.

**Rule #1:** For 1 inch of rainfall:

- Run your program with the following input:  .
- Move your mouse over the rain gauges to see the absorption and runoff. Record the results below.

Total absorption: \_\_\_\_\_ inches

Total runoff: \_\_\_\_\_ inches

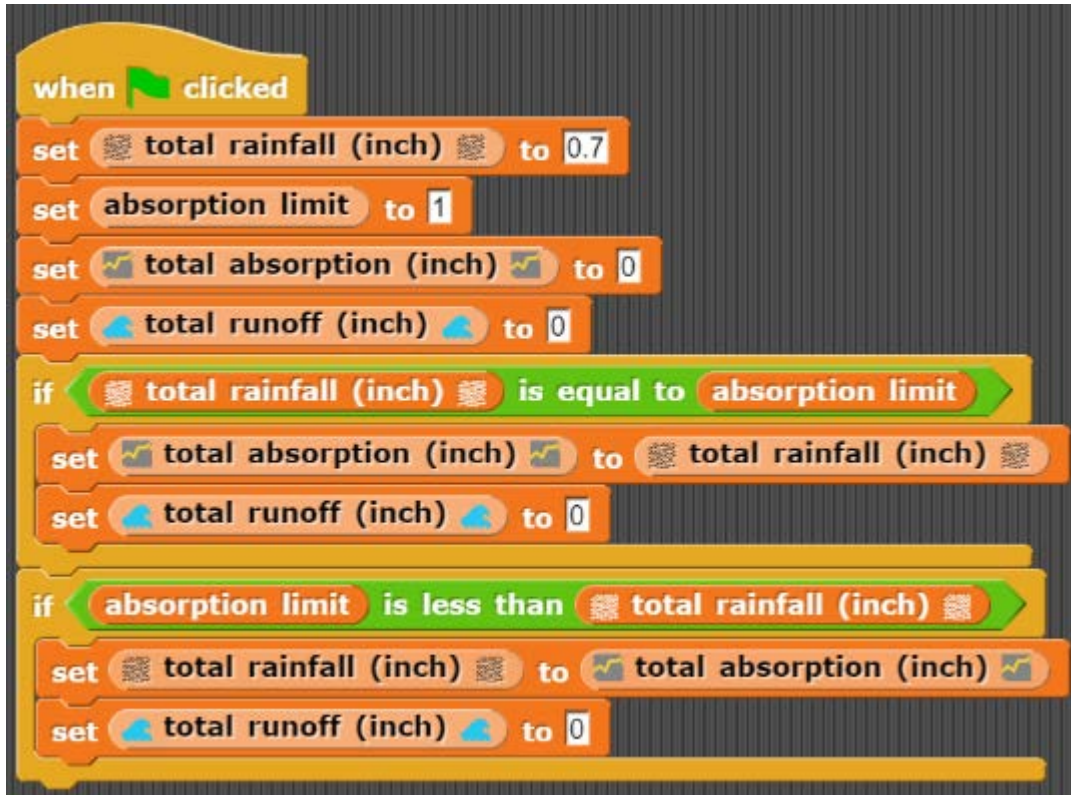
**Rule #2:** For 0.7 inches of rainfall:

**Rule #3:** For 1.4 inches of rainfall:



# Debugging practice

Sumi is testing her model by selecting a surface material whose absorption limit is 1 inch.



Can you identify the error(s) in Sumi's model? Hint: Look closely, there might be more than one error.

How can you fix Sumi's model?

What the values of output variables SHOULD be	Output variable values actually produced by the computer model
Total absorption in inches: _____	Total absorption in inches: _____
Total runoff in inches: _____	Total runoff in inches: _____



# Synthesis

- The NGSS crosscutting concepts of ‘Systems and system models’ and ‘Matter and Energy’ (conservation of matter) help achieve coherence across science, engineering and computing
- Science-anchored computational engineering problems make computational modeling practices consequential for students
- Computational modeling helps shift the focus from engineering as building to engineering as a problem-solving process

# Project team

PIs: Kevin W. McElhaney (SRI), Jennifer L. Chiu (UVA), Gautam Biswas (Vanderbilt)

## SRI

- Nonye Alozie
- Satabdi Basu
- Ron Fried
- Reina Fujii
- HeeJoon Kim
- Jennifer Knudsen
- Beth McBride

## UVA

- Chris Dittrick
- Sarah Fick
- James Hong
- Sarah Lilly
- Anne McAlister

## Vanderbilt

- Ningyu Zhang

**Acknowledgements:** This material is based upon work supported by the National Science Foundation under Grant No. DRL-1742195. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

