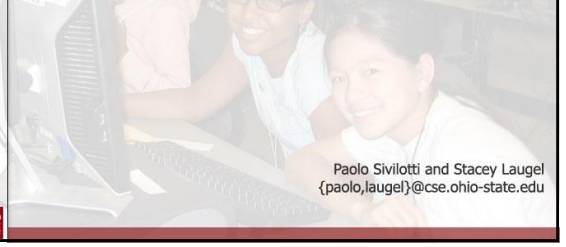
The Ohio State University Department of Computer Science & Engineering

Scratching the Surface
of Advanced Topics in Software Engineering:

A Workshop Module for Middle School Students



Paolo Sivilotti and Stacey Laugel
{paolo,laugel}@cse.ohio-state.edu

The Challenge: Context

- "Future Engineers Summer Camp"
  - Summer day camp at OSU (1 week, free)
  - 30 8th grade girls, mostly from Columbus area
  - *Many* science and engineering disciplines
- Our opportunity
  - Design a "computer activity"
  - Time budget: 3 hours

Our Goals

- Intellectual honesty
  - Relate to intellectual core of computational science
- Exposure to deep ideas
  - Introduce advanced, college-level, topics in CS
- Short-term engagement
  - Keep students' attention for the duration of module
- Long-term engagement
  - Afford continued independent involvement afterwards

Option 1: Kinesthetic Learning

- Kinesthetic learning activities
  - Students are physically engaged
- People as processors
  - network routing, flow-chart hopscotch
  - Self-stabilizing distributed algorithm for mutual exclusion (SIGCSE '03)
- People as data structures
  - human binary tree, sort the students
  - Parallel garbage collection (SIGCSE '07)
- Benefit: No hardware infrastructure needed
  - What we've previously used

Option 2: Programming

- Introduction to programming
  - Students learn a (simple) programming language and environment
- Many Options
  - Alice, Scratch, Karel, Logo, Phrogram, …
- Benefit: Foundations of computational thinking

Option 3: Technology Use

- Computers are exciting, enabling tools
  - Students use some computer-based technology to learn and have fun
- Examples
  - Web pages, Google Earth, Dance Dance Revolution
- Benefit: Theme of "computer literacy"
  - computers have a huge impact on our life

Comparing Goals

	Intellectual Honesty	Deep Ideas	Short-Term Engagement	Long-Term Independent Engagement
Kinesthetic Learning Activities	Yes if done right	Yes if done right	Yes	No
Intro to Programming	Yes	No	Yes	Yes
Technology / End-user	No	No	Yes	Yes
Our Module	Yes	Yes	Yes	Yes

Our Ambition

	Intellectual Honesty	Deep Ideas	Short-Term Engagement	Long-Term Independent Engagement
Kinesthetic Learning Activities	Yes if done right	Yes if done right	Yes	No
Intro to Programming	Yes	No	Yes	Yes
Technology / End-user	No	No		
Our Module	Yes	Yes	Yes	Yes

Scratch

- Visual programming environment
  - Drag-and-drop control blocks
- Emphasis on multimedia
  - Create sprites
- More information:
  - http://scratch.mit.edu
  - SIGCSE special session and paper later this afternoon

Scratch: Strengths

- Easy to learn
  - Very little "lecture time" needed
- Fun
  - Multimedia, (re)mixing
- Computational thinking 101
  - Variables, conditionals, etc.
- Active online community
  - Forums, galleries of projects (with code)

Deep Idea 1: Refinement

$$R \Rightarrow R' \mapsto (R, E) \leq (R', E)$$

- Specifications are *antimonotonic* in their preconditions
  - Weaker precondition ➔ stronger spec

$$E \Leftarrow E' \mapsto (R, E) \leq (R, E')$$

- Specifications are *monotonic* in their postconditions
  - Stronger postcondition ➔ stronger spec

Deep Idea 2: Composition

$$A \leq A' \wedge B \leq B' \mapsto A \| B \leq A' \| B'$$

- Large systems are assembled from collections of (correct) components

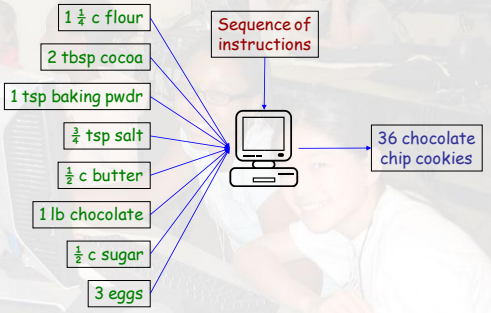
Research Questions

1. Can a short module really achieve all these goals?
2. Which goals are compromised, and to what extent?
   - Does addressing "deep ideas" decrease the degree of short-term engagement?
   - Assessment with control group

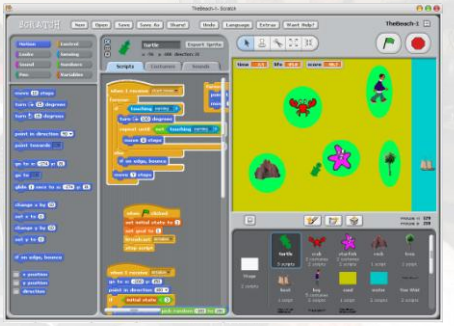
Module: Overall Structure

- Intro lecture:
  - Role of CS in society
  - Software engineering as "recipe engineering"
  - Scratch intro (5 min!)
- Lab activity: "Save the Turtle"
  - Specification Refinement
- Lab activity: "Butterflies and Dragons"
  - Composition

Recipe Engineering



I: "Save the Turtle"



Lab Activity

- Solve the same problem with progressively harder restrictions
  - A) Fixed Start/Reach the Ocean
  - B) Random-facing Start/Reach the Ocean
  - C) Fixed Start/Reach the Ship
  - D) All-random Start/Reach the Ocean

What we observed

- A is simple
  - Strong precondition, weak postcondition
- B is still pretty easy
  - Moderate precondition, weak postcondition
- C is doable
  - Strong precondition, strong postcondition
- Most teams got these three to work

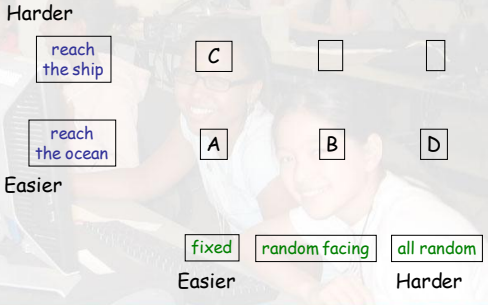
Results

- D is a struggle
  - Weak precondition
  - Most teams could accomplish with some hints
    - A few even went on to get All-random Start/Reach the Ship!

Student Lessons Learned

- Students agreed A is easiest, D is hardest
  - Why is C harder than A?
    - More constraint on programmer to reach the ship
  - Why is B harder than A?
    - Less information about where the turtle starts

Comparing Specifications



Bringing it all together

- Tie specification refinement back to "recipe engineering"
  - How difficult is it to make chocolate chip cookies if you have ½ cup unsalted butter? ½ cup fat? Some fat?
  - How easy is it to make 12 chocolate chip cookies? Any 12 cookies? Something sweet?
- Easiest when we know a lot about the input (and have few requirements on the output)

II: "Dragons and Butterflies"

Lab Set Up

- Two parts
  - First write individual sprites
  - Then mix sprites together and modify
- Use the "break" to merge/distribute individual sprites

Lab Activity

- Part 1
  - Each team gets assigned a sprite (fluttering butterfly, fire breathing dragon, score keeper, etc.)
  - Given specific requirements
    - A fluttering butterfly that flies around in a random pattern
  - Free to incorporate sound effects, colors, and other elements
    - as long as requirements are still satisfied

Lab Activity

- Break
  - Collect the sprites into one directory
  - Reexamine the "recipe engineering" analogy
    - Final dish (output) can be consumed as an ingredient (input) for another dish

Lab Activity

- Part 2
  - Students mix and match sprites to compose a game
  - Assuming sprite requirements were followed, "ingredients" fit together
  - In remaining time, students customized games

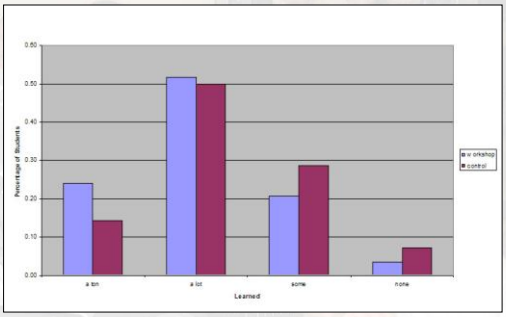
Student Lessons Learned

- Programs can be made from smaller programs
  - Side note: Specifications are important

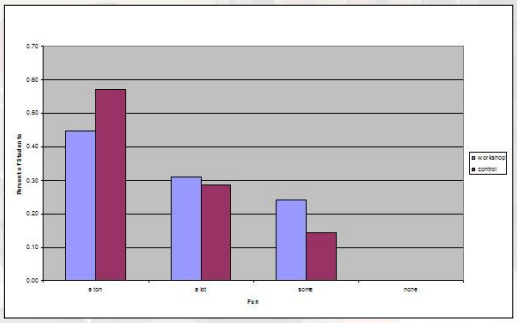
Module Assessment

- Survey after activity
  - Background, experience
  - How much they learned about CS
  - Short-term engagement
  - (Expected) future engagement
- Compared with a control group
  - Classic learn Scratch and program
  - 14 students vs. 30 in workshop group

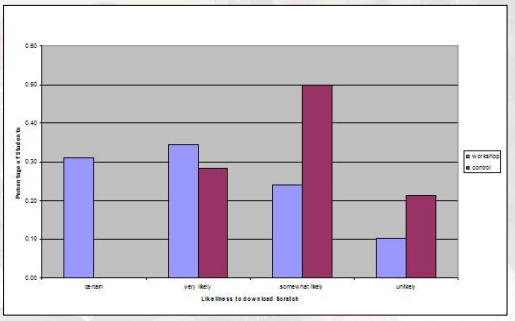
How much did you learn?



Short-term Engagement



(Exp.) Long-term Engagement



Results Summary

- How much did you learn?
  - Almost everyone felt they learned something about CS
  - A little higher for workshop group
- Short-term Engagement
  - Fun is about the same in both groups
- (Exp.) Long-term Engagement
  - Workshop group more likely to download Scratch
- Most students "changed their mind about computing" (ALL of them for the positive)

Conclusions

- A 3 hour workshop can be intellectually honest, introduce deep ideas, and be engaging (short-term and long-term)
- Scratch is not designed to teach deep ideas
  - Very good for engagement
  - Don't change it!
- 3 hours (with breaks) is a little tight

Materials

- Lectures, handouts, and code
  - http://www.cse.ohio-state.edu/~paolo/outreach/

Scratching the Surface

of Advanced Topics in Software Engineering:

A Workshop Module for Middle School Students

Questions?

Paolo Sivilotti and Stacey Laugel
{paolo,laugel}@cse.ohio-state.edu

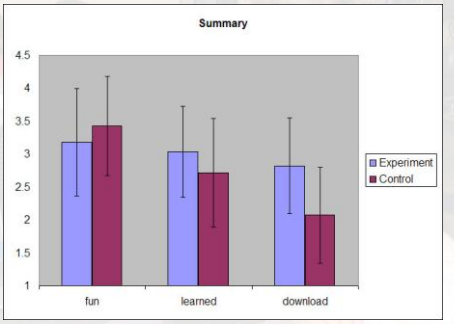
Survey Data

	Response	Workshop n=29	Control n=14
1. Before this lab, had you ever written a computer program?	Yes	9	1
	No	19	13
2. Do you have a computer and Internet connection at home?	Yes	27	13
	No	1	1
3. How likely are you to download Scratch at home in the next week?	Certain	9	0
	Very likely	10	4
	Somewhat likely	7	7
	Unlikely	3	3
4. How much fun did you have in this activity?	A ton	13	8
	A lot	9	4
	Some	7	2
	None	0	0
5. How much do you feel you learned in this activity?	A ton	7	2
	A lot	15	7
	Some	6	4
	None	1	1
6. Did this lab change your opinion of computer science (as a discipline or a career option) in any way?	Yes, a lot	3	4
	Yes, a little	21	9
	No, not at all	5	1
7. If you answered "yes" above, how did your opinion change? Do you now view computer science more or less favorably?	More favorably	24	12
	Less favorably	0	1

Scratching the Surface of Advanced Topics in Software Engineering
A Workshop Module for Middle School Students

The Ohio State University Department of Computer Science & Engineering

Summary of Survey Data



Stacey Laugel SIGCSE 2008