Learn to code? No, it's about code to learn!

Ongoing research on the benefits of learning to program at school

Gregorio Robles
grex@gsyc.urjc.es

Universidad Rey Juan Carlos (Madrid, Spain)

Gothenburg, December 15th 2016

About me

Figure : My (main) research

Overview

1. What is CT?
2. Learn to code
3. Code to learn

Section 1

What is CT?

Definition of Computational Thinking

- Computational Thinking is the process by which a problem is formulated and a solution is expressed in such a way that a computer can carry it out effectively. (Wing, 2006)
- It is based on an iterative process with three stages:
  1. The formulation of a problem (abstraction),
  2. the expression of a solution (implementation),
  3. and the execution and evaluation of the solution (analysis).

Background picture: Simon Cunningham

History of learning *with* computers

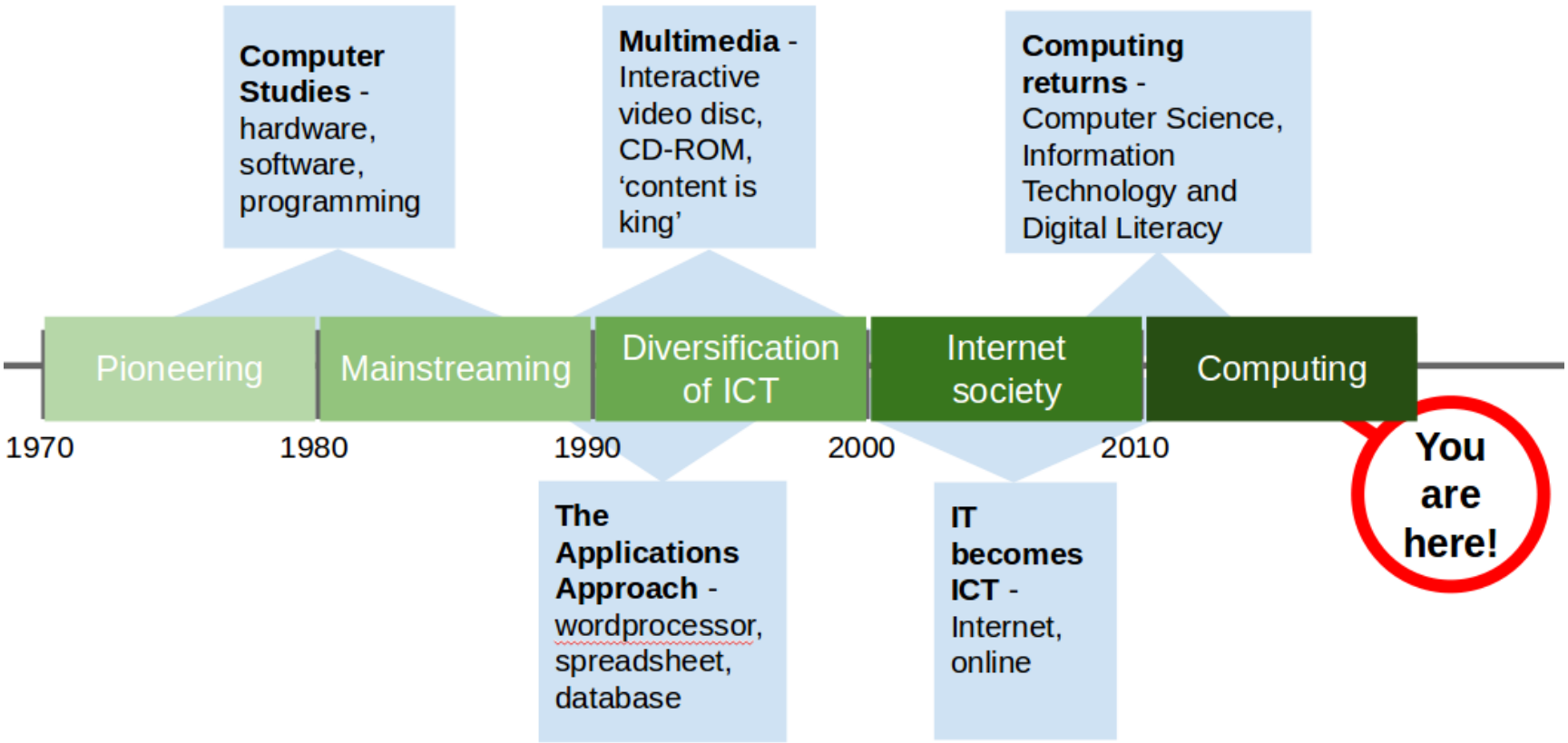


Figure : You are here!

Source: *REaCT EU project proposal*

Is CT a new skill?

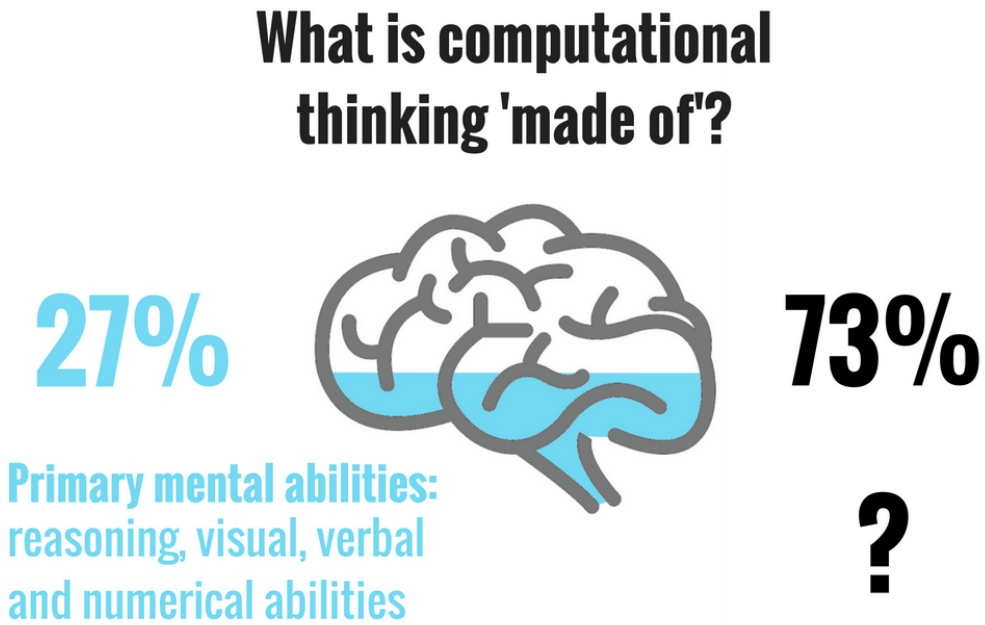


Figure : Computational thinking is an independent skill, as only 27% of its variance can be explained by the four primary mental abilities.

Impact of Research on Cognitive Effects of Programming

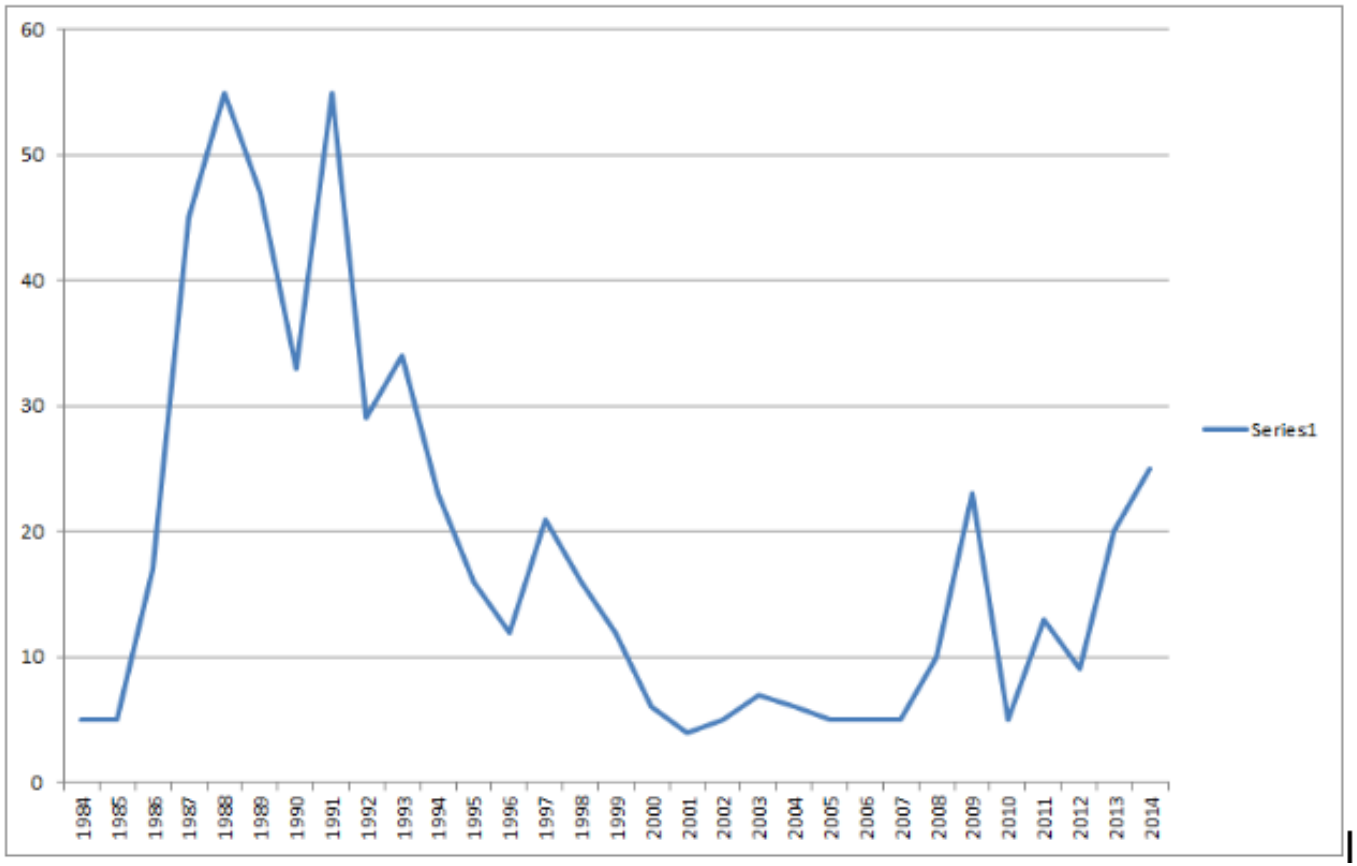
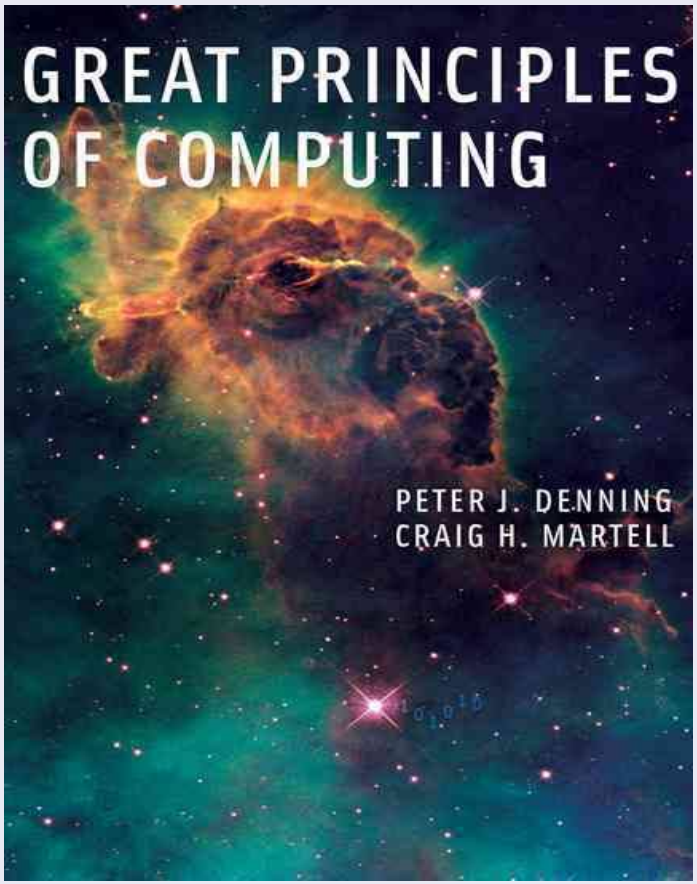


Figure : Sum of citations of 11 key papers on cognitive effects of programming 1984-2014. Calculated using Web of Science

Source: *Nina Breshnihan, Trinity College Dublin*

Great Principles of Computing

The Book



Seven Principles

1. Information
2. Machines
3. Programming
4. Computation
5. Memory
6. Parallelism
7. Queueing
8. and Design

Section 2

Learn to code

What is Scratch?

An example:

`https://scratch.mit.edu/projects/49905542/`

Some fun

https://scratch.mit.edu/projects/114556022/

Why automatic analysis? (Learner perspective)

Analyzing a Python program with Pylint


```
Global evaluation
-----------------
Your code has been rated at 9.41/10

Raw metrics
-----------

+----------+-------+------+---------+-----------+
|type      |number |%     |previous |difference |
+==========+=======+======+=========+===========+
|code      |115    |64.61 |NC       |NC         |
+----------+-------+------+---------+-----------+
|docstring |40     |22.47 |NC       |NC         |
+----------+-------+------+---------+-----------+
|comment   |4      |2.25  |NC       |NC         |
+----------+-------+------+---------+-----------+
|empty     |19     |10.67 |NC       |NC         |
+----------+-------+------+---------+-----------+
```

Why automatic analysis? (Educator perspective)

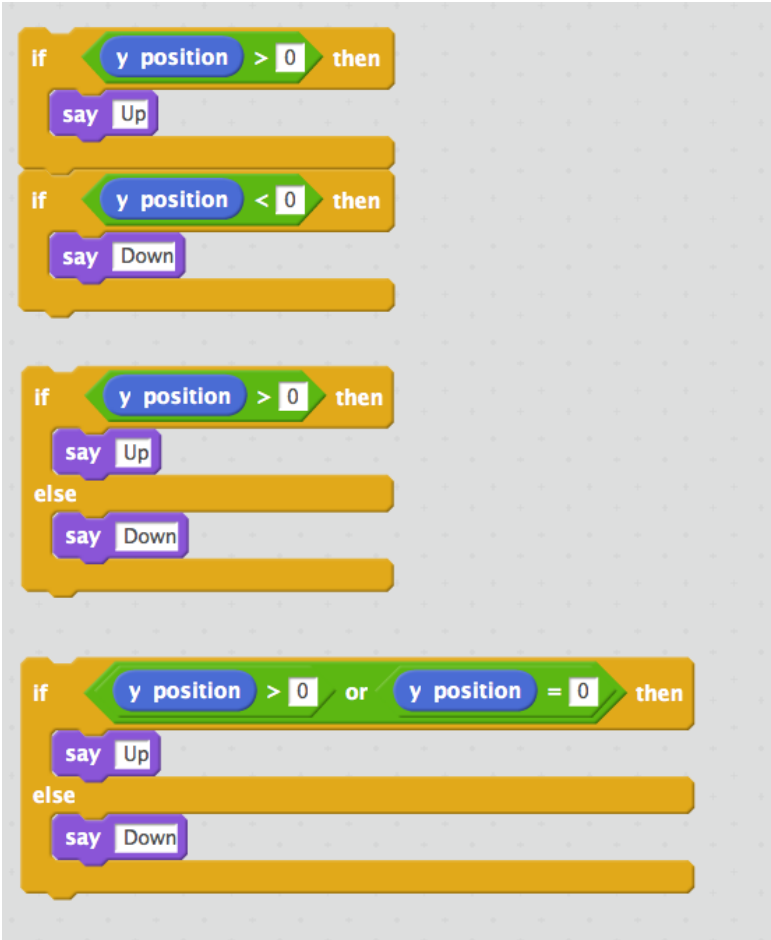
You know what I am talking about

Assessment of CT development

CT dimension	Basic	Developing	Proficient
Data representation	modifiers of sprites properties	operations on vars	operations on lists
Logical Thinking	if	if else	logic operations
User interactivity	green flag	key pressed, sprite clicked, ask and wait, mouse blocks	when %s is >%s, video, audio
Algorithmic notions of flow control	sequence of blocks	repeat, forever	repeat until
Abstraction and problem decomposition	more than one script and more than one sprite	def block	clones (instances)
Parallelism	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, two scripts when %s is >%s, two scripts on when backdrop change to
Synchronization	wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	wait until, when backdrop change to, broadcast and wait

Table : Level of development for each CT dimension

Assessment of CT development: Logical Thinking



Different levels of development of logical thinking: basic (top), developing (center) and proficient (bottom).

Code smells (I)

branch: master ▾ hairball / hairball / plugins / duplicate.py

bboe on Apr 15 Merge caching support.

2 contributors

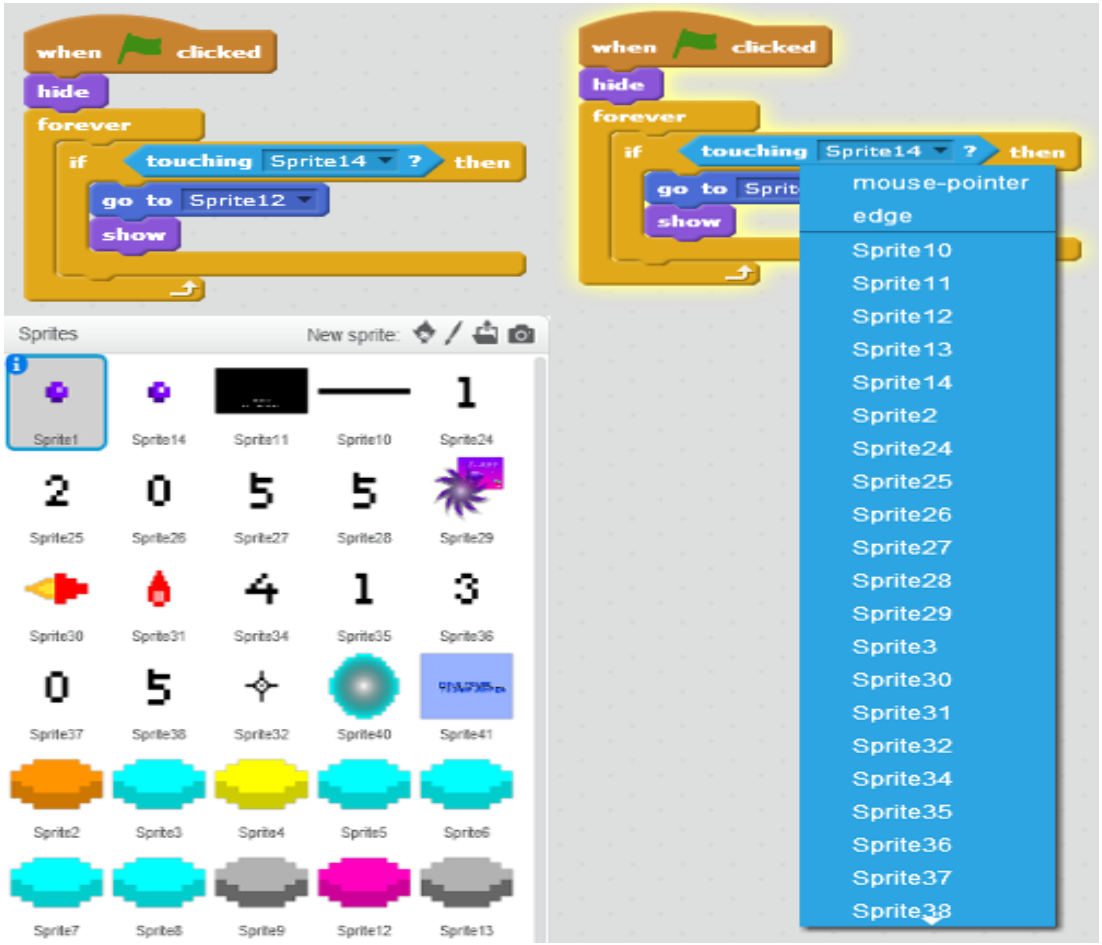
44 lines (34 sloc) 1.56

Errors or bad programming habits

- Dead code
- Attribute initialization
- Default names
- Repeated scripts

```
1  """This mo
2
3  from __futu
4  from hairba
5
6
7  class Dupli
8
9      """Plug
10
11     def __init__(self):
12         """Initialize an instance of the DuplicateScripts plugin."""
13         super(DuplicateScripts, self).__init__()
14         self.total_duplicate = 0
15         self.list_duplicate = []
16
17     def finalize(self):
18         """Output the duplicate scripts detected."""
19         if self.total_duplicate > 0:
```

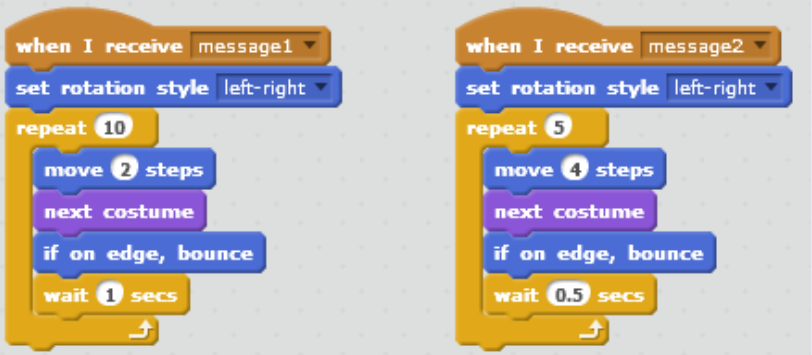
Code smells (II)



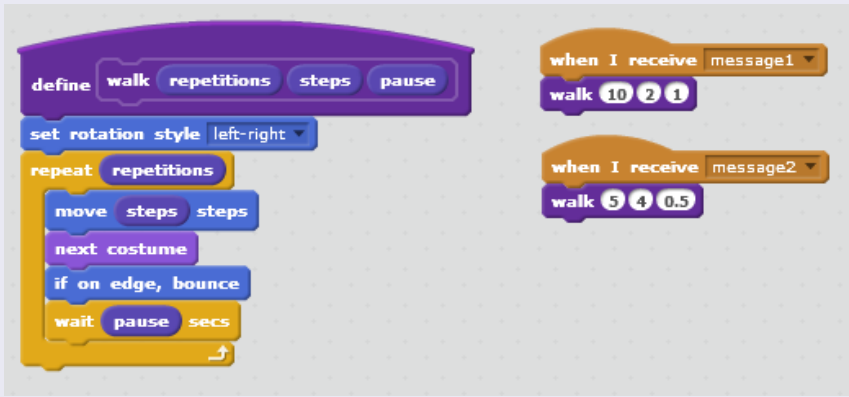
Bad/default naming of sprites

Code smells (and III)

Example of repeated code

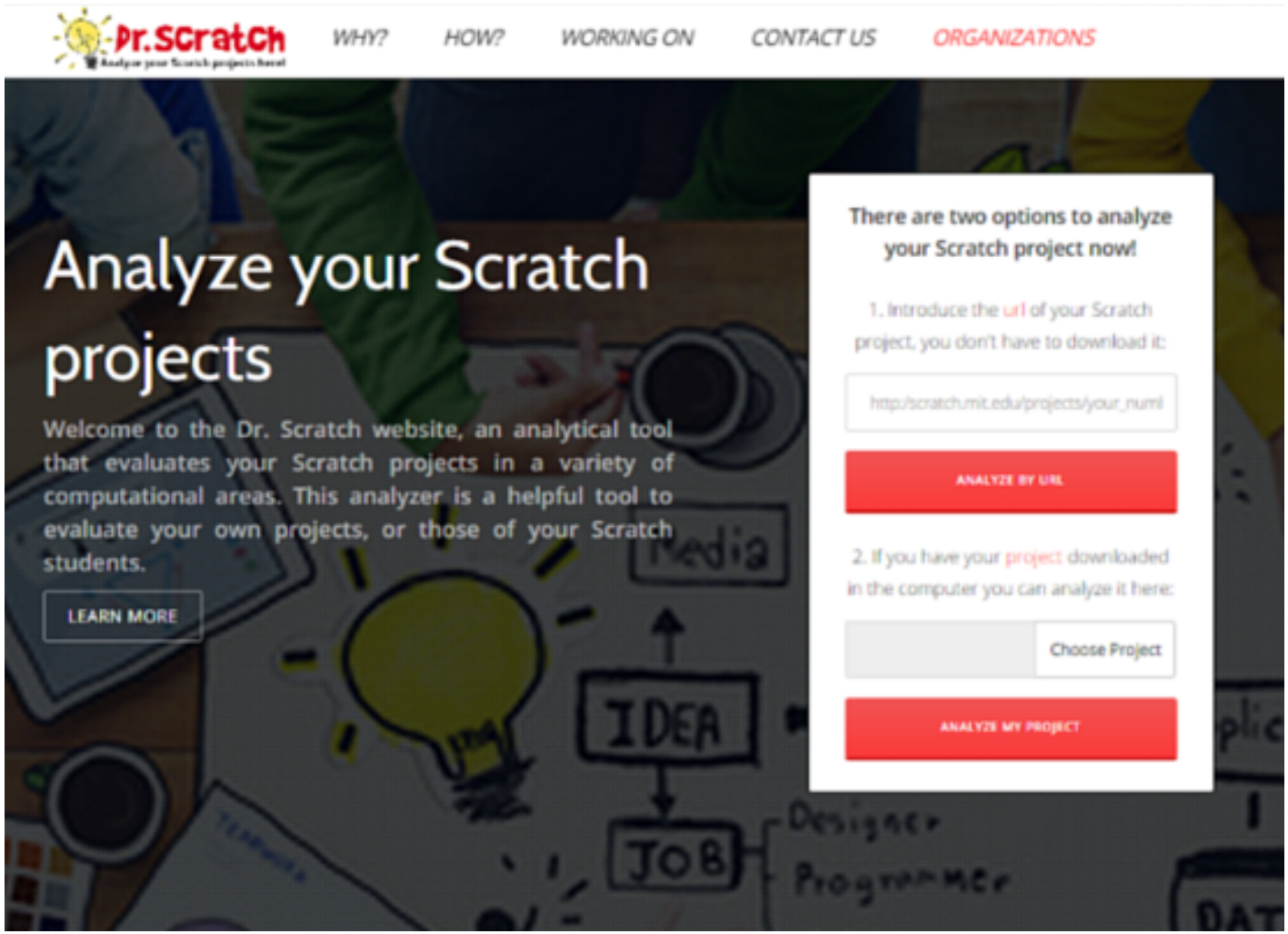


Solution to avoid repeated code



Blocks should be created to avoid repetition of code

Dr. Scratch

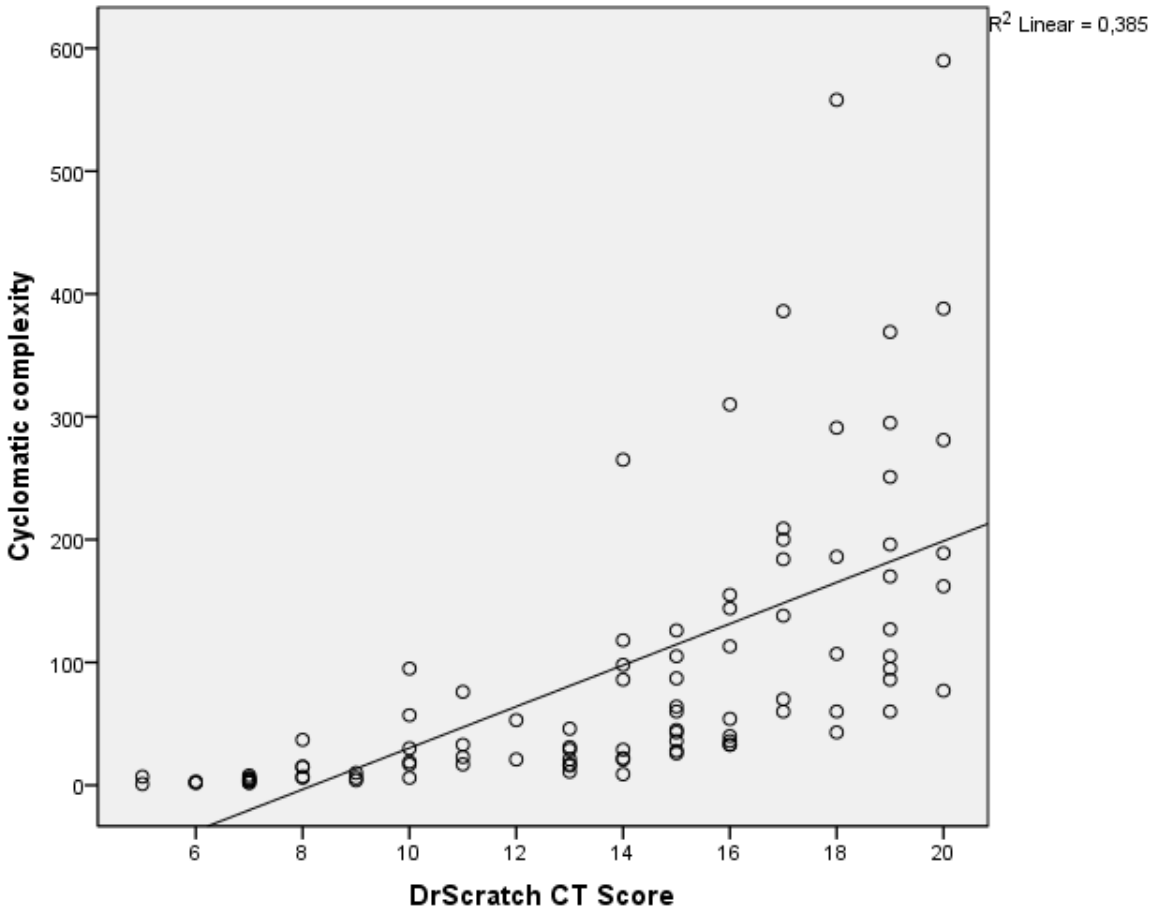

Dr.SCratch
WHY?
HOW?
WORKING ON
CONTACT US
ORGANIZATIONS
Analyze your Scratch projects
Welcome to the Dr. Scratch website, an analytical tool that evaluates your Scratch projects in a variety of computational areas. This analyzer is a helpful tool to evaluate your own projects, or those of your Scratch students.
LEARN MORE
There are two options to analyze your Scratch project now!
1. Introduce the url of your Scratch project, you don't have to download it:
http://scratch.mit.edu/projects/your_num
ANALYZE BY URL
2. If you have your project downloaded in the computer you can analyze it here:
Choose Project
ANALYZE MY PROJECT
IDEA
JOB
Designer
Programmer

Feedback from/tested with learners

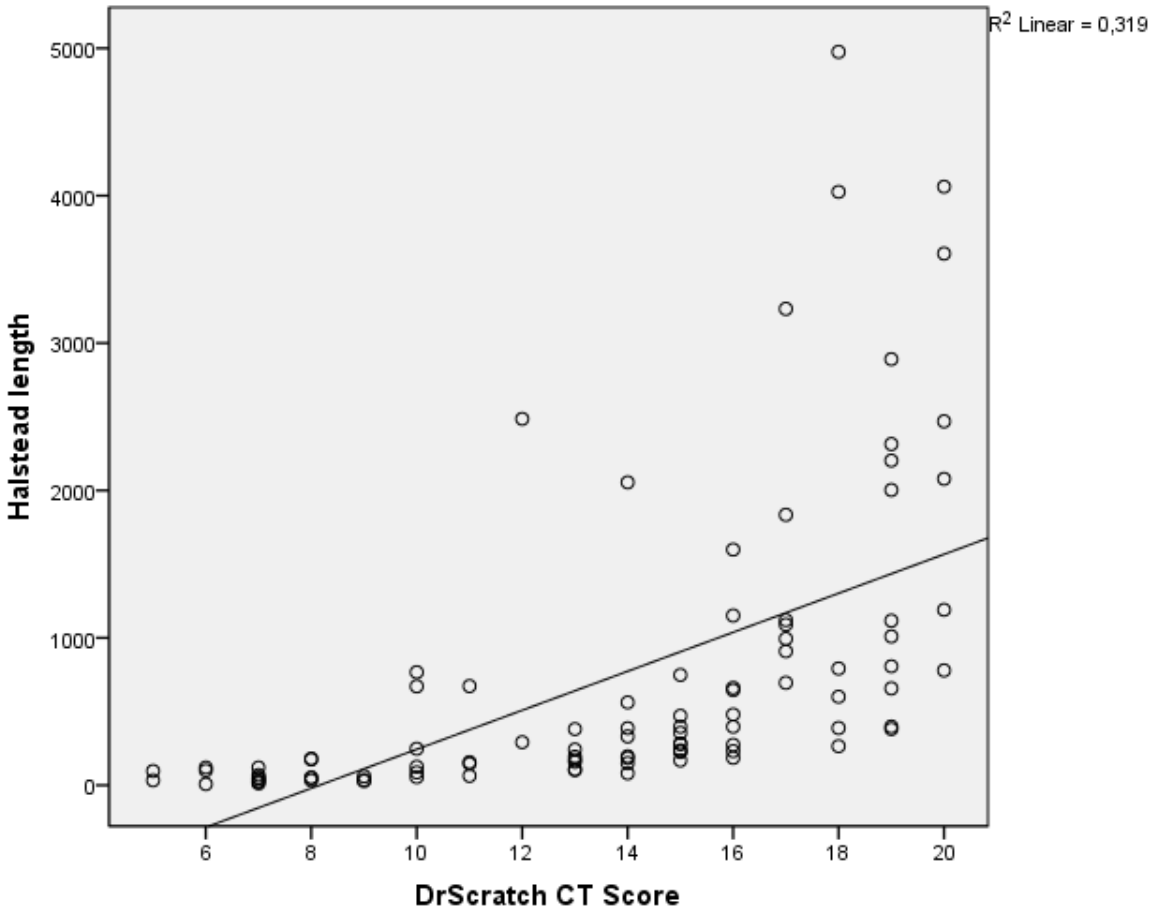
Tested with learners and teachers

Workshop at CEIP Lope de Vega, Madrid (Spain)

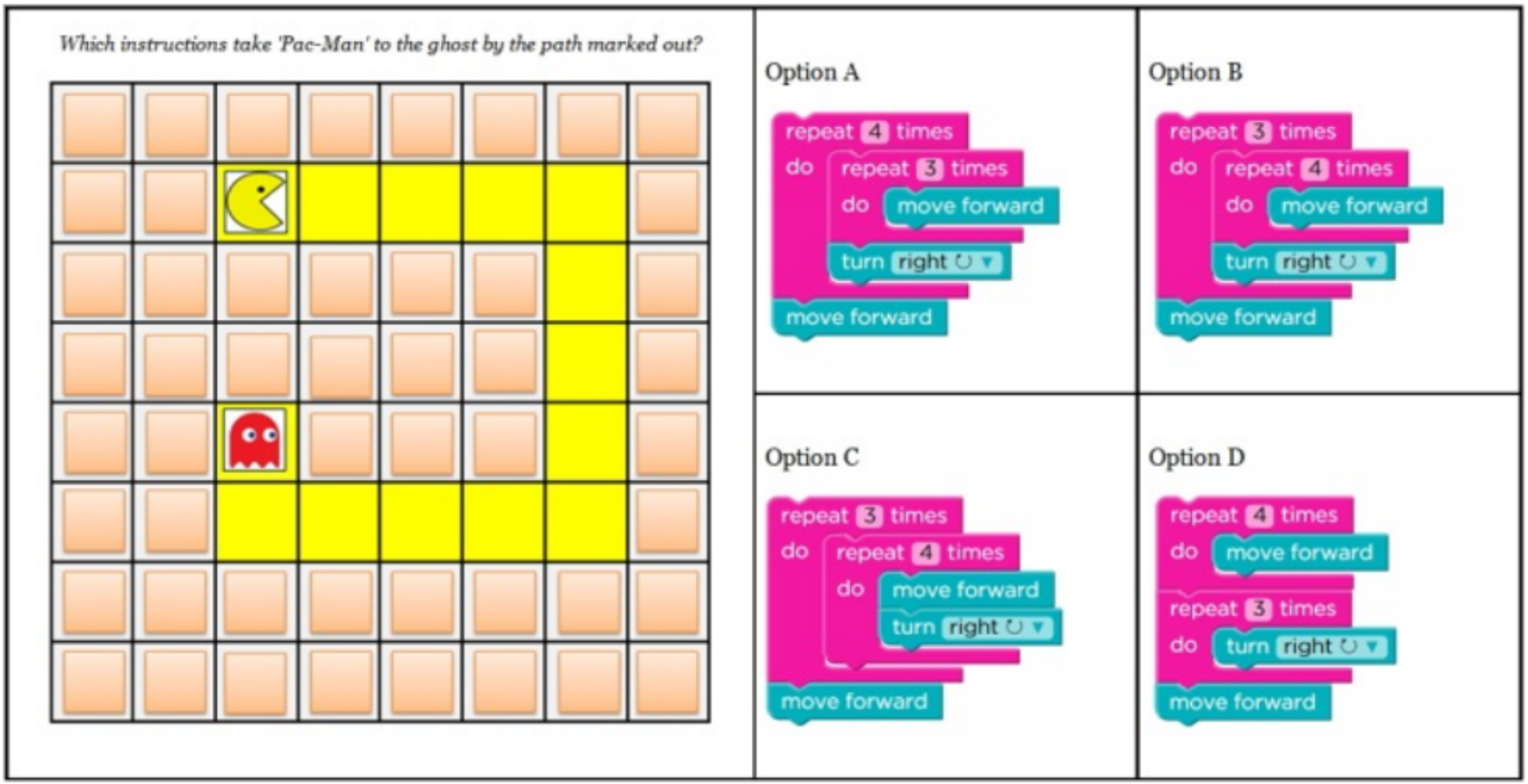
Dr. Scratch vs classic Software Engineering Complexity (I)



Dr. Scratch vs classic Software Engineering Complexity (II)

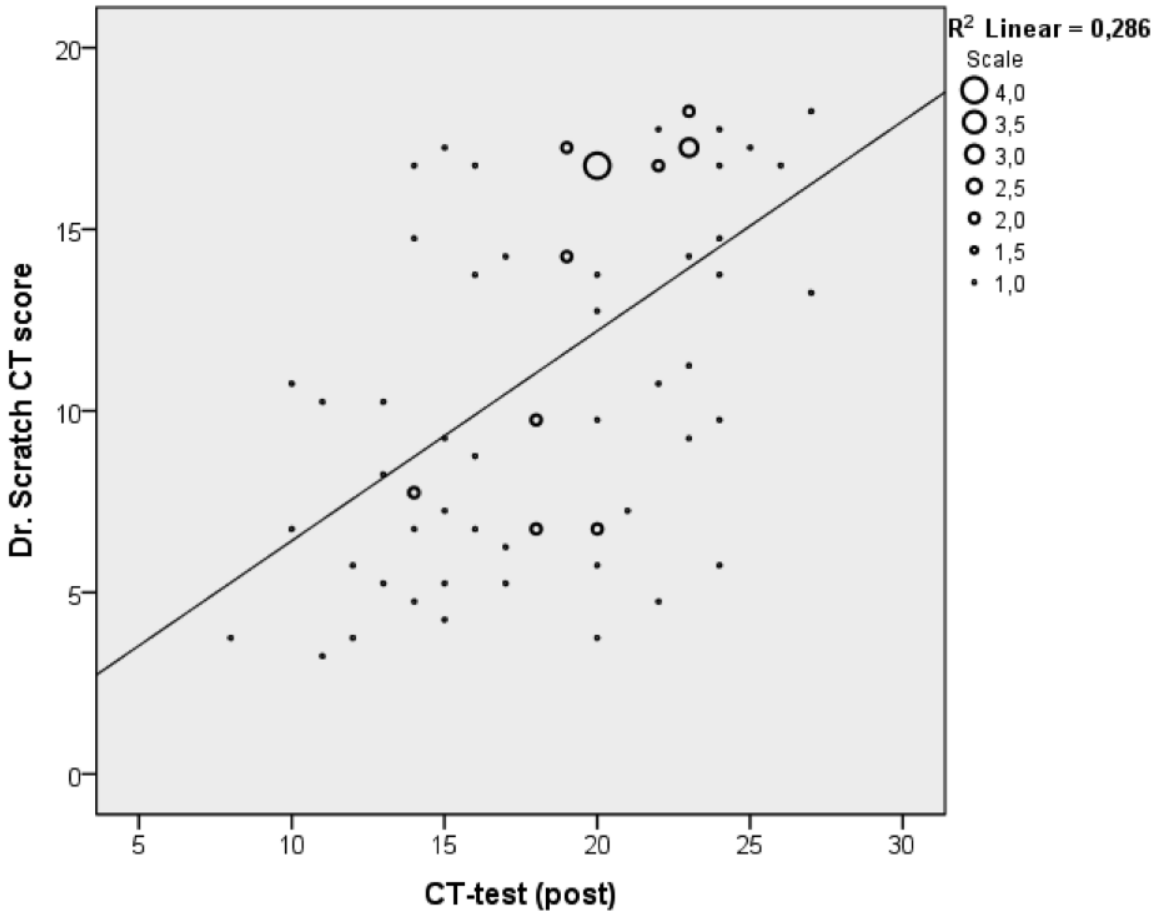


Dr. Scratch vs CT-test (I)

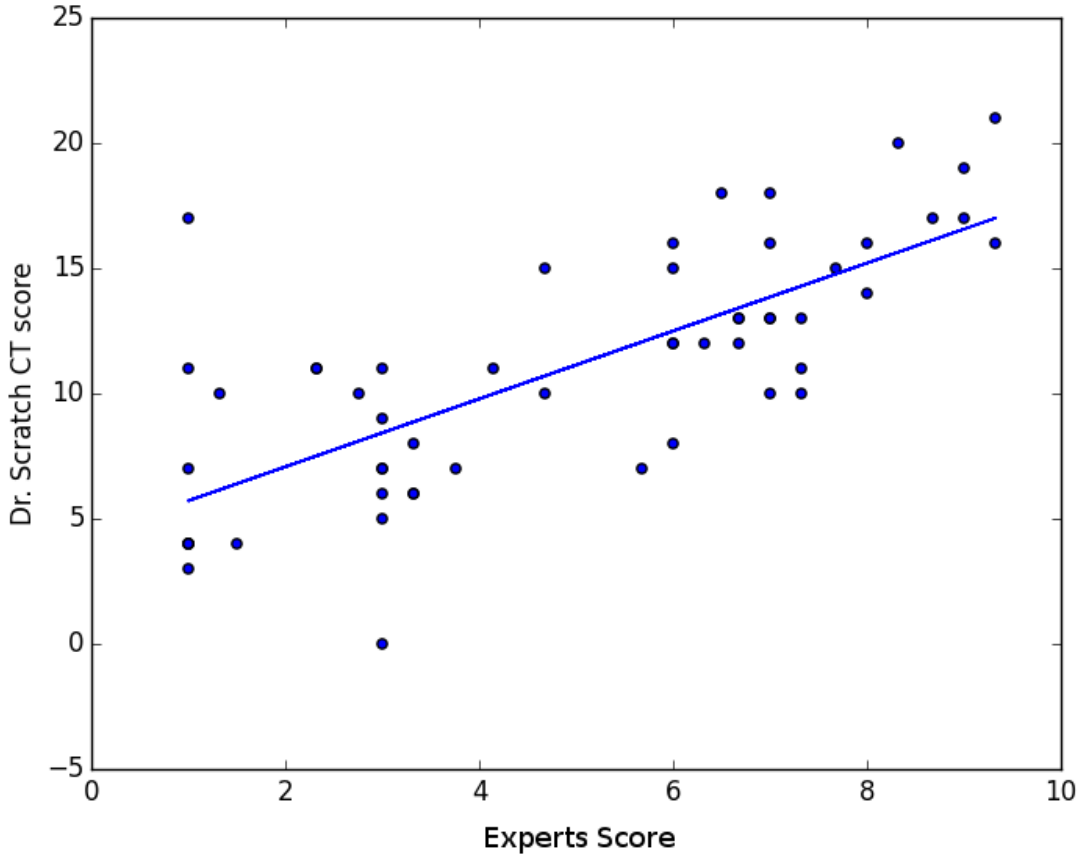


One of the CT-test items

Dr. Scratch vs CT-test (and II)



Dr. Scratch vs Expert judgment ($r > 0.7$)



Dr. Scratch and Cheating?

- Still work in progress
- Tell learners to try to obtain the highest possible score in Dr. Scratch
- Hypothesis: You can cheat up to your level of CT skills

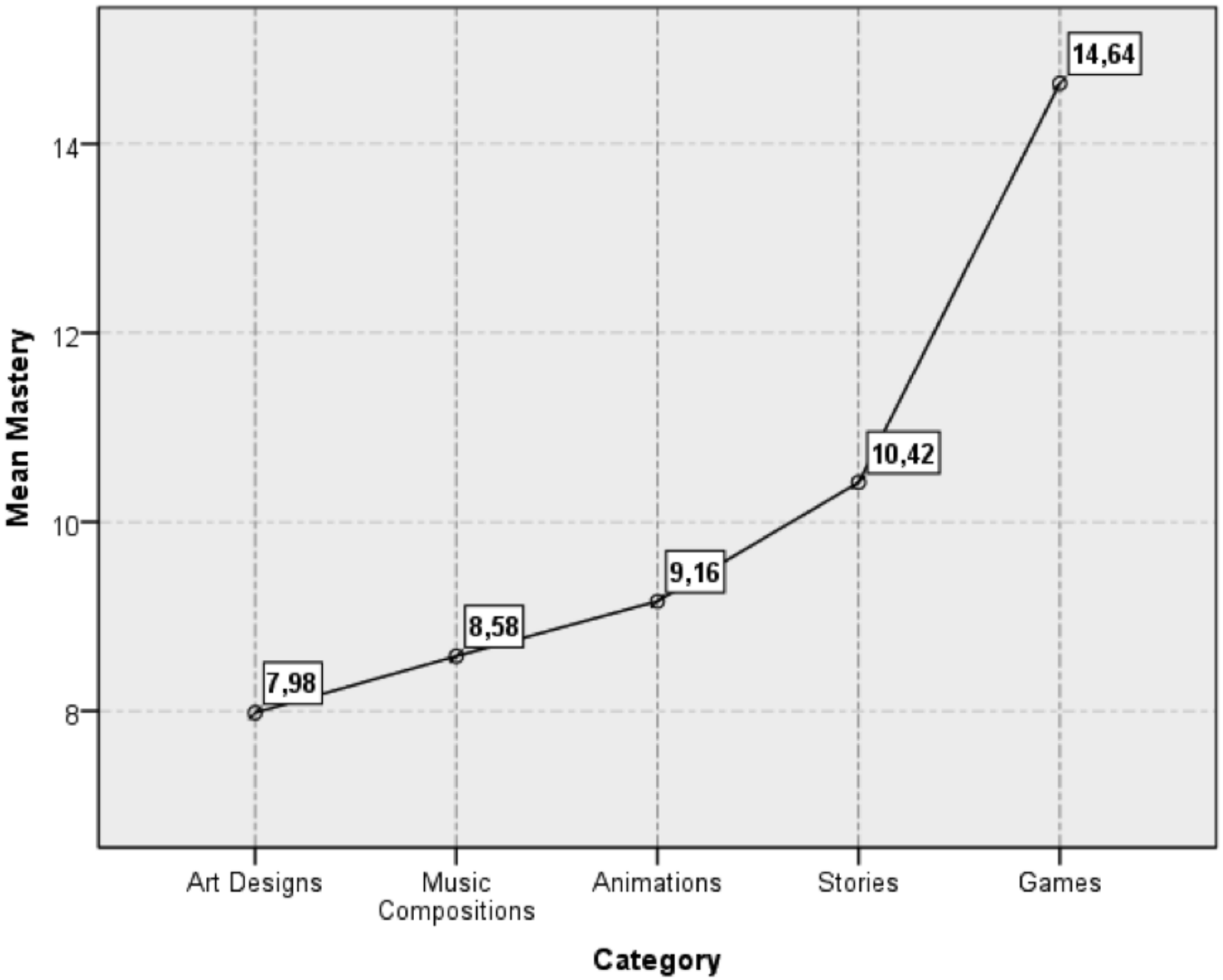
Limitations

Teachers should not rely exclusively on Dr. Scratch

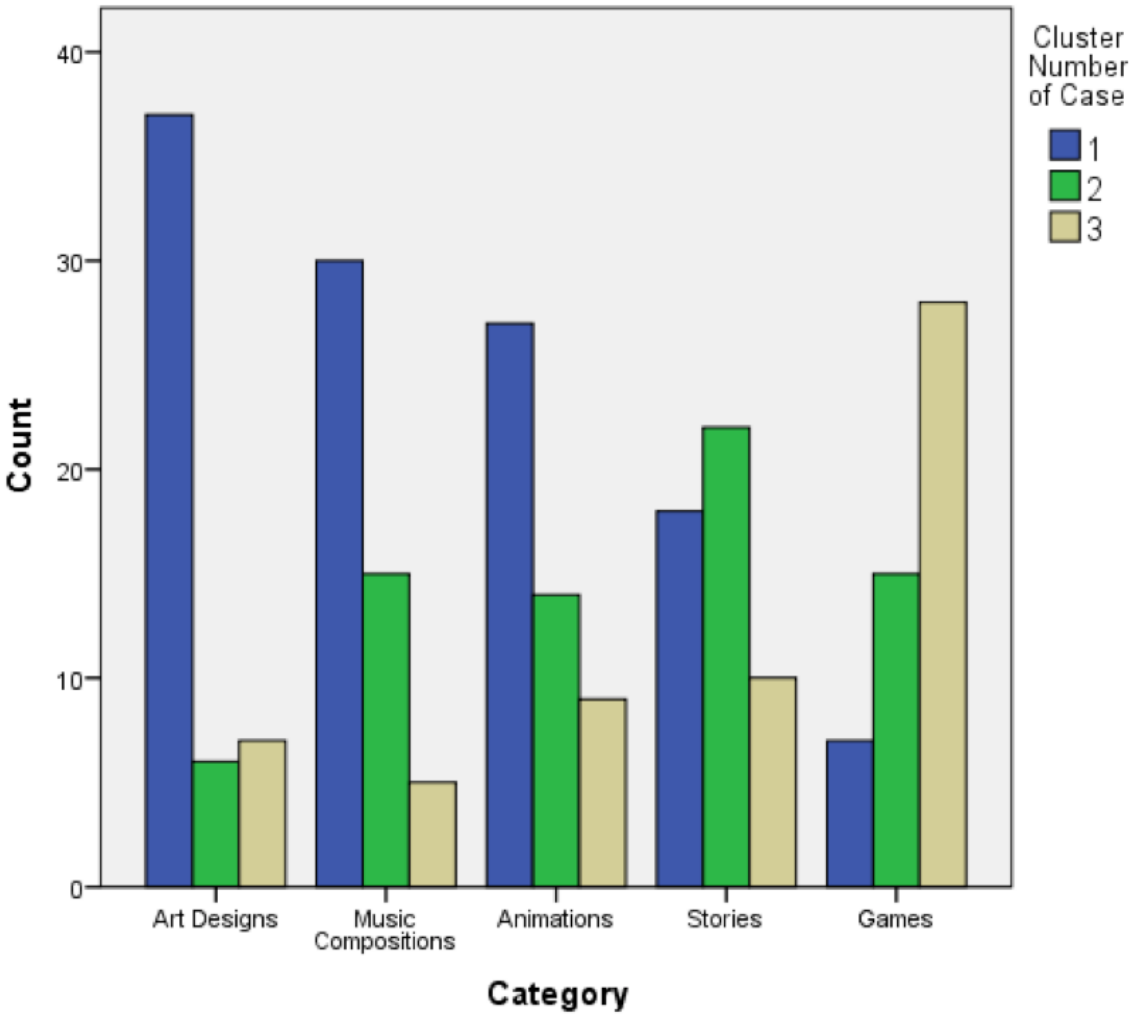
- Fundamental CT skills are not assessed
  1. Debugging
  2. Remixing (aka. Forking)
  3. Design/Modeling
- Functionality or creativity is not evaluated.
- Democratizing digital expression (MIT Medialab's view) may become secondary.

Background picture: Robert Couse-Baker

Data-driven itineraries



Data-driven itineraries



Does CT correlate with personality?

- Empirical evidences of the correlations between CT and the five factors of personality from the **Big Five** model:
  - Expected positive correlations with *Openness* (r = 0.41) and *Conscientiousness* (r = 0.27)
  - Unexpected positive correlation with *Extraversion* (r = 0.30)
  - No relationship with *Agreeableness* (r = 0.02) and *Neuroticism* (r = 0.01)

Social learning

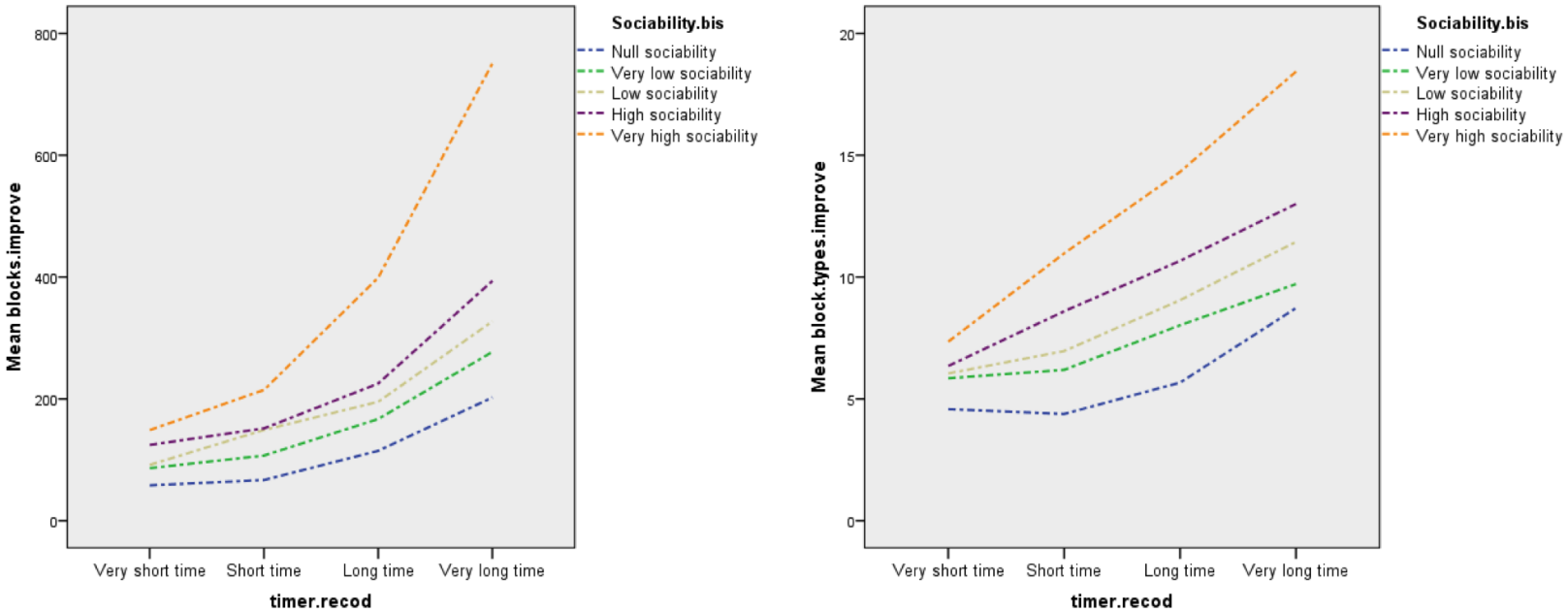


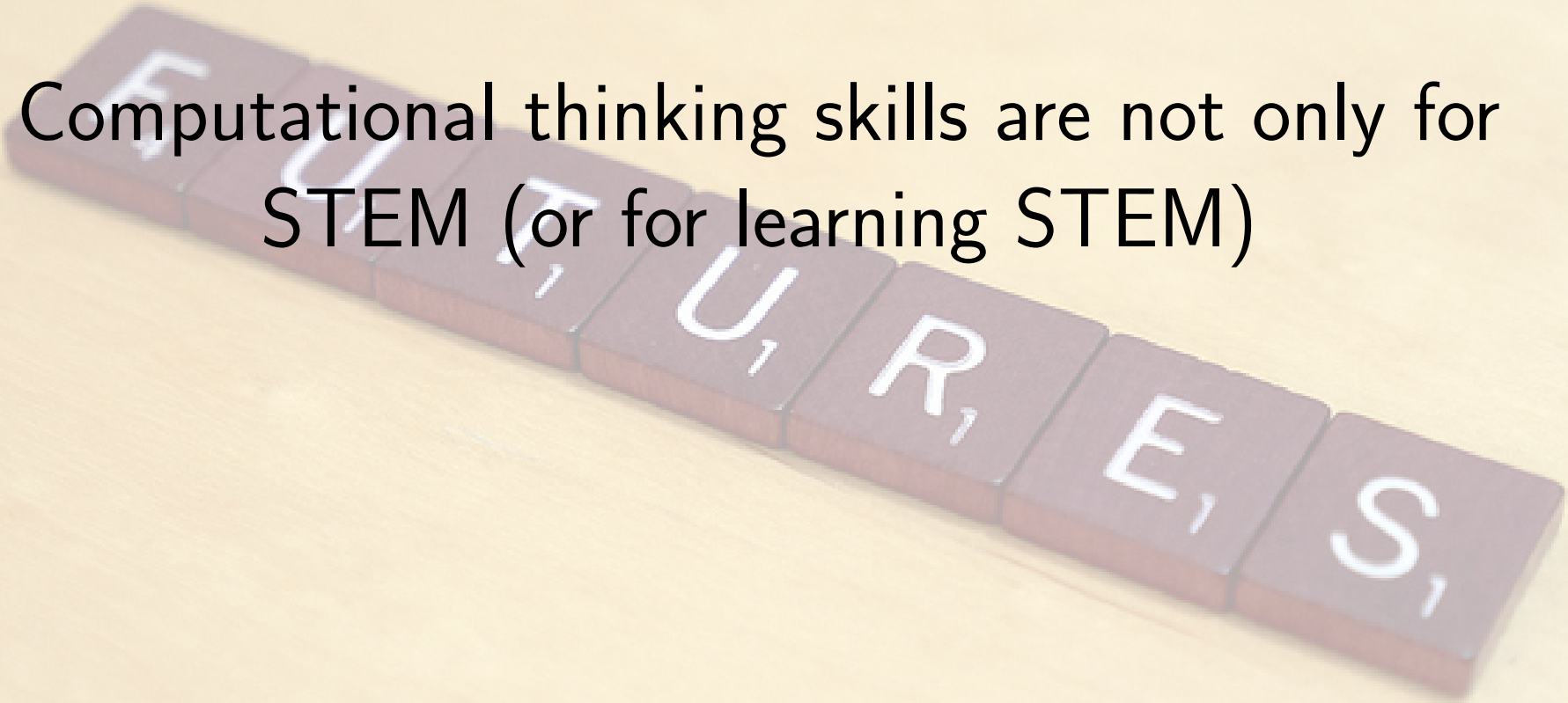
Figure : (l) Relationship of level of sociability with improvement in depth for each time group. (r) Relationship of level of sociability on improvement in breadth for each time group.

Section 3

Code to learn

Code to learn

Computational thinking skills are not only for STEM (or for learning STEM)



Background picture: Simon Cunningham

Coding beyond STEM

- Programming is a way of expression (that is complementary to other ways of expressing yourself)
- Learners become *prosumers*, not just consumers of technology
- STEM + Art = STEAM. Haven't heard about it before?
- There is (preliminary) research on the (good) effects of coding in other subjects

Background picture: ngu.edu

Maths

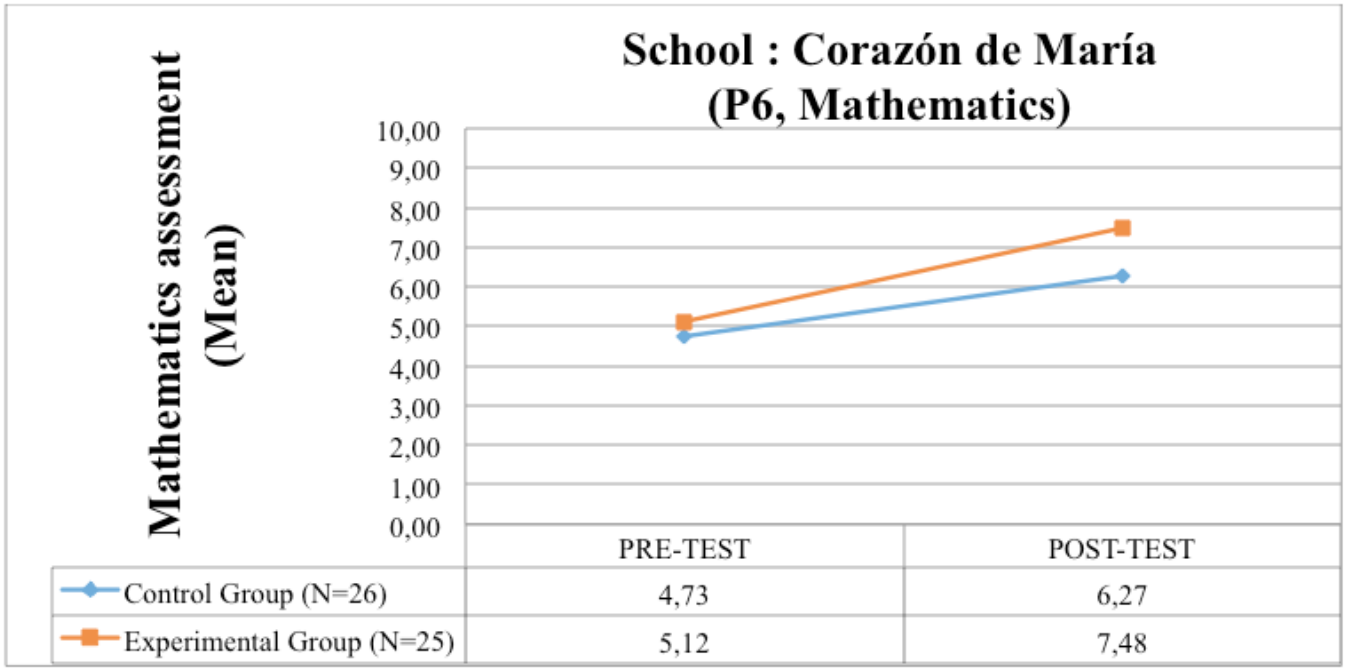


Figure : Corazón de María school. Comparing the improvement between pre- and post-tests of control and experimental groups.

Social sciences

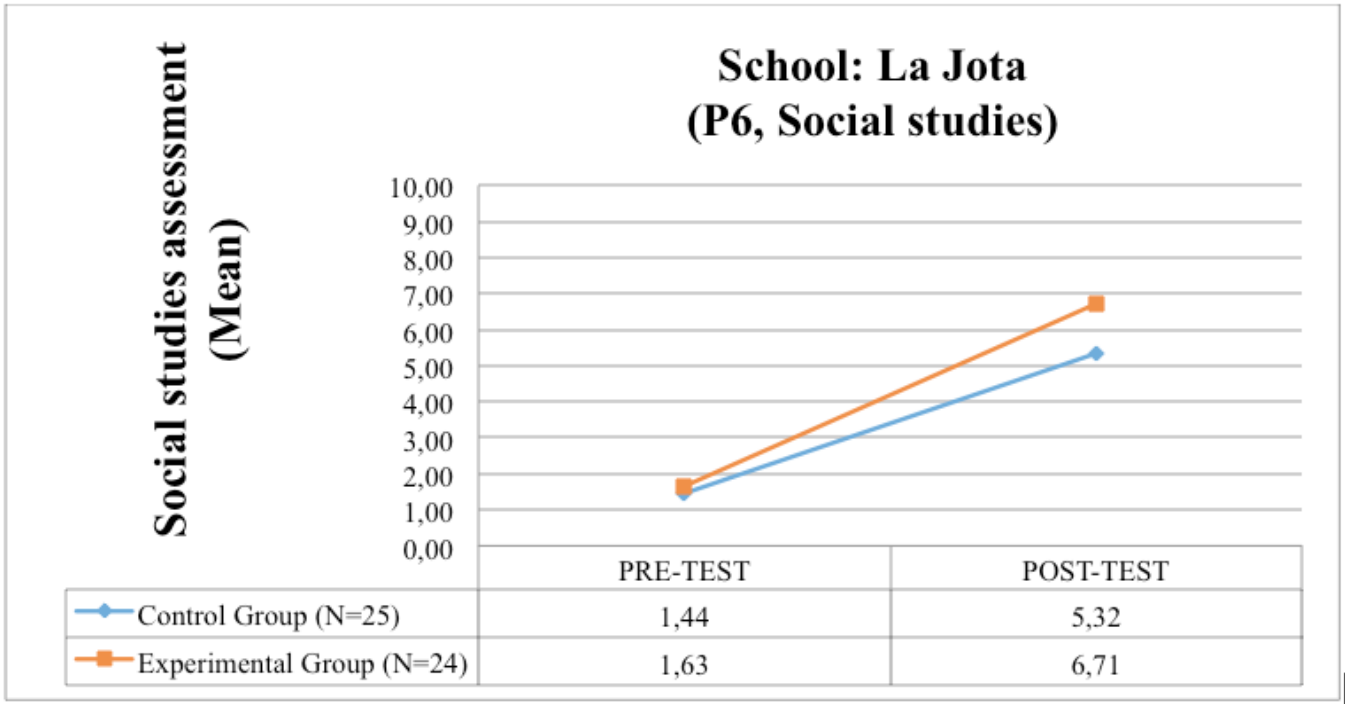


Figure : La Jota school. Comparing the improvement between pre- and post-tests of control and experimental groups.

Arts

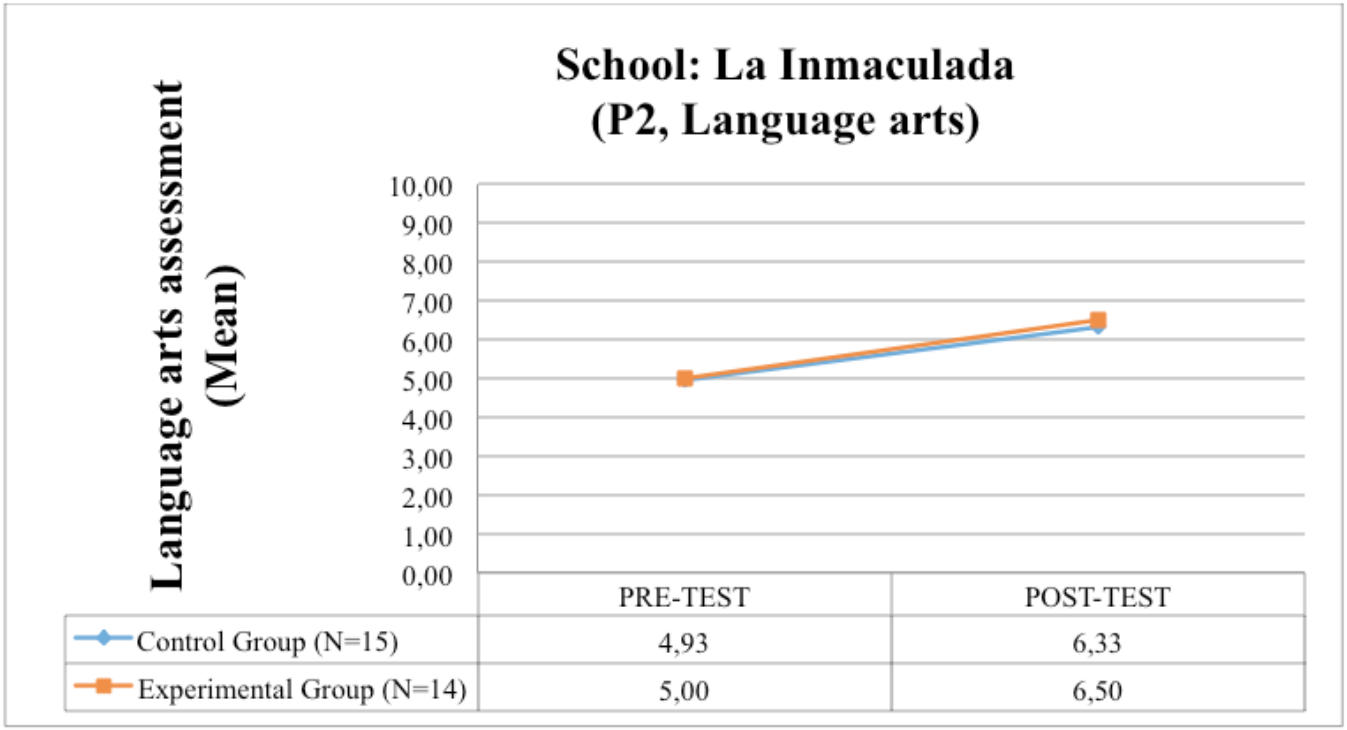


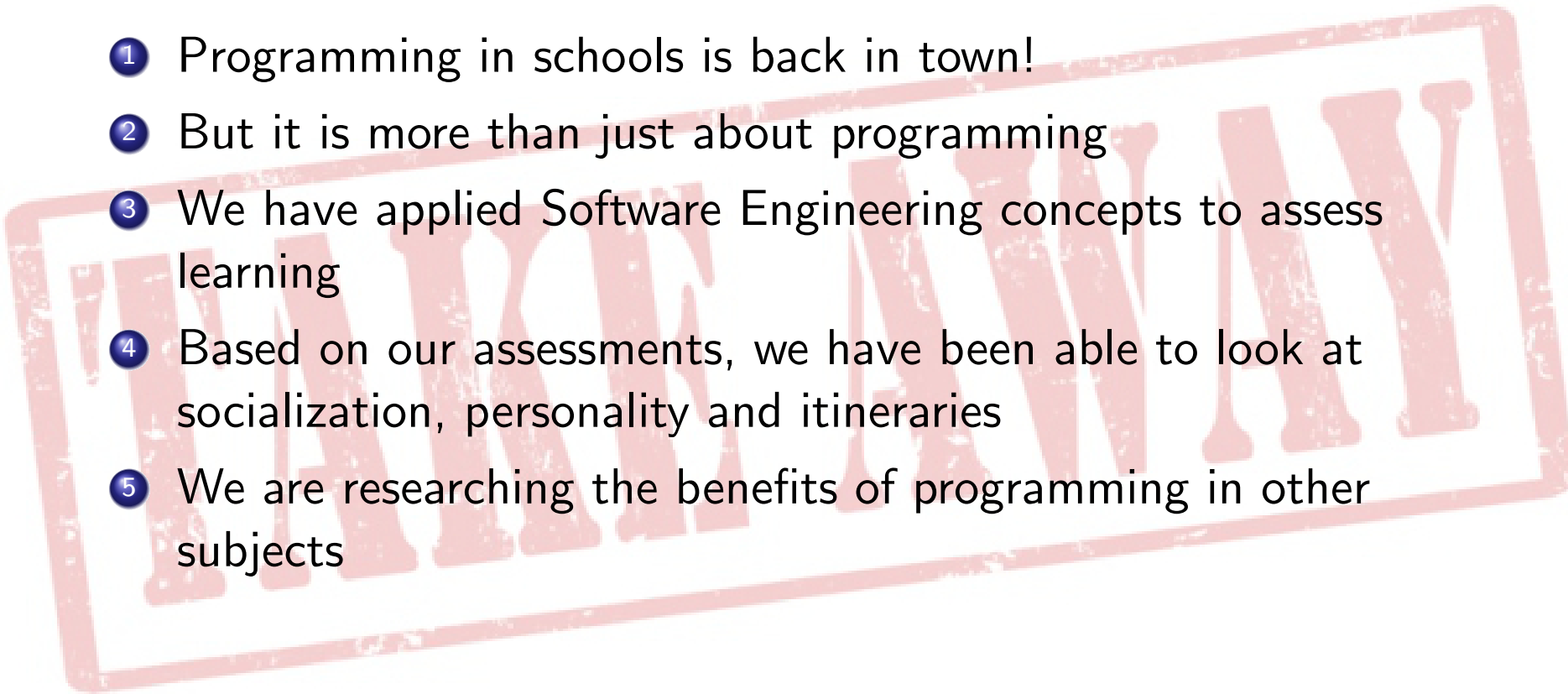
Figure : La Inmaculada school. Comparing the improvement between pre- and post-tests of control and experimental groups.

Research in Progress: Art with 3-year old kids

Figure : A five year old student coding a programmable robot to develop arts skills and learn about the painting *Bathers at Asnières*.

Takeaways

1. Programming in schools is back in town!
2. But it is more than just about programming
3. We have applied Software Engineering concepts to assess learning
4. Based on our assessments, we have been able to look at socialization, personality and itineraries
5. We are researching the benefits of programming in other subjects



Background picture: flamingcow.co.uk

Learn more

Some references

- Moreno-León, J., Robles, G, & Roman-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED. Revista de Educación a Distancia, 15*(46).
- Moreno-León, J., Robles, G, & Roman-González, M. (2016). Comparing computational thinking development assessment scores with software complexity metrics. In *Global Engineering Education Conference (EDUCON), 2016 IEEE*. IEEE.
- Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2016, November). Does computational thinking correlate with personality?: the non-cognitive side of computational thinking. *In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 51-58). ACM.

Learn to code? No, it's about code to learn!

Ongoing research on the benefits of learning to program at school

Gregorio Robles
grex@gsyc.urjc.es

Universidad Rey Juan Carlos (Madrid, Spain)

Gothenburg, December 15th 2016