‘Think as a computer scientist’

Short presentation for Bratislava-Cepis

Mini-conference Coding and Computational Thinking (CT) at Schools

Program:

1. Personal introduction
2. Coding/programming: the difference
3. Coding in the EU
4. Implementing Coding and Computational Thinking (CT)
5. CT pedagogy and how to practice e.g. language
6. Questions and discussion

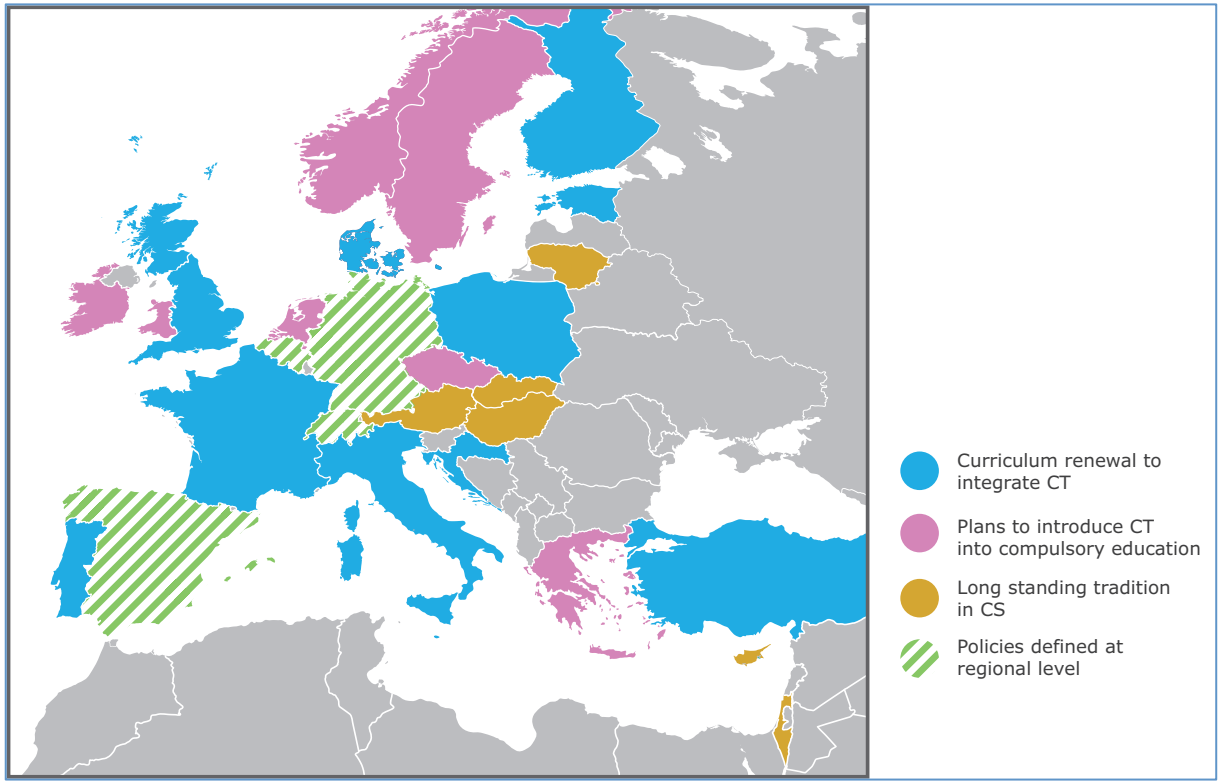
Personal Introduction

- Cepis-SIN/CIS member
- Editor and founder of www.komenskypost.nl
- NHL-University for Applied Science (Leeuwarden, Friesland) : focus on changing pedagogy and ICT
- Staff member for informatics at Nat.Institute for Curriculum Development
- Studied Dutch language & literature and Theoretical linguistics (Computational linguistics) at UvA

An analysis of educational approaches to develop Computational thinking. Implications for policiy and.practice. JRC- Science report. Brussels 2016, In press.

- How can we define CT as a key 21st century competence for all citizens/schoolchildren?
- What is the relationship of CT to coding / programming?
- How do you educate the teachers?
- Must CT integrate in STEM or or as a cross-curriculum topic
- How to test CT? Traditional test methods do not seem so appropriate.
- How CT continue the education agendas.

Coding in the EU



- *An analysis of educational approaches to develop Computational thinking. Implications for policiy and.practice.* JRC- Science report. Brussels 2016, In press.

CT approaches?

Definitions:

- Computational thinking : computer principles, language and problem solving skills (Sharples 2015)
- Leads to learning a ‘language’ and grammar compare Niels Bohr:“Science is not to tell us about the universe, but to tell us how to talk about the universe.” compare: CT is not about how a computer works but how can we ‘communicate’ with and about IT
- Computational thinking: is about problem solving, algorithms, decomposition, conceptualizing not just programming, fundamental for everyone, combi and complement of math.thinking and engineering (Wing 2006)

Computational thinking skills

- Wing (2006) Is more than just coding. What can humans do better than computers? And What can computers do better than humans?
- Most fundamentally it addresses the question: What is computable? Today, we know only parts of the answers to such questions. (Wing, 2006)
- CT is about communication with computers -→ language
- Grasp the notion of computability (incompleteness th. Gödel)
- Sharples (2015) key-elements: decomposition, pattern recognition, abstraction, algorithm design, debugging present a solution – Refining those steps.

- Language is a great teaching vehicle.
- For CT is not simply a way to gain problem-solving skills, but also a way to express yourself with digital media.
- CT competencies makes it able to design and collaborate. Mitchel Resnick emphasizes the relationship between language and CT as a form of literacy (bildung)
- Resnick (at Scratch 2016): “It’s not about learning to program but about programming to learn’”

Problem based learning

- 1. Examine a case and clarify terms
- 2. Identify the problem
- 3. Analyse the problem
- 4. Draft an explanatory model
- 5. Establish learning goals
- 6. Work individually to collect additional information
- 7. Apply and discuss additional information (Sharples 2015)

The pedagogy of Scratch and the maker movement, Resnick

- Listening is forgetting
- Reading is remembering
- Making is understanding

Why focus on language within CT?

1. **History argument:** ‘50: development of higher programming language: Math/automat. theory, electronic engineering , theoretical linguistics, cognitive psychology

2. **Pedagogical argument:** discover and explain theoretical concepts for not only beta-students e.g. (formal) grammar, natural vs. artificial language, AI, recursion, algorithms, finite state and (CF)-pushdown automata, Girls like language orientated applications

3. **Sustainability** Is a sustainable element of CT (till we have quantumcomputer?)

4. **Language technology**: automatic translation, speech and speaker recognition etc.

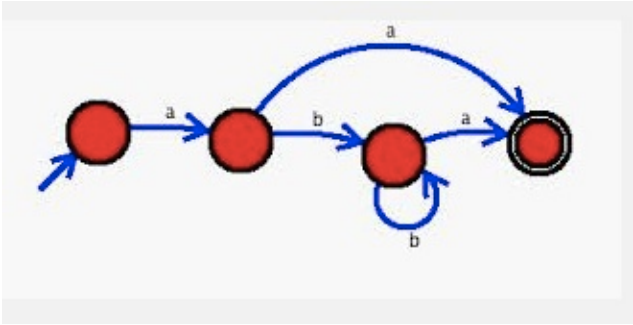
5. **Related to CT key-issues** (Wing)

The computational capacity: a basic concept within CT/Coding

- consists of a mechanism with finite amount of terminal elements (words, number, sounds) and a set of combination rules
- that can generate an infinite amount of structured utterances (sentences, melody's) with a specific system of rules: a grammar

How?
e.g. by using a ‘toolbox’

- Using CF-sentence generators: for language (push down automata)
- Generating Haiku’s
- Rap’s
- Sentences
- Stories



- The key are the assigments, a combi of constructionism interaction, some instruction and make it meaningfull. There is not general recipe.

Conclusions

- CT, coding/programming asks for a non-instructionalist based approach
- Some instruction is always OK but not dominating
- Focus on active learning
- Let students/pupils collaborate
- Constructionist: maker education. Use 3D-printing, laser cutters.
- This demands for ‘new’ pedagogical skills of teachers
- Challenge is: develop more examples of teaching and assessment

Literature:

- Papert, Seymour. *Mindstorms. Children, Computers, and Powerful Ideas.* New York, 1980.
- Sharples, Mike*. Cognition, computers and creative writing*. Chichester, 1985
- Resnick, Mitchel. *Turtles, Termites, and Traffic Jams. Explorations in Massively Parallel Microworlds.* Cambridge (Mass.) 1997.
- Bruer, John T, *Schools for Thought*. *A science of learning in the classroom.* MIT-Press, 1997.
- Martinez S. and G. Stager. *Invent to Learn. Making, Tinkering, and Engineering in the Classroom.* Torance (CA.), 2013.
- Wing See Jeannette M. Wing, 2006, "Computational Thinking," in *Communications of the ACM* 49(3): 33-35,
- Sharples, Mike. Innovating Pedagogy. In: *Exploring new forms of teaching, learning and assessment, to guide educators and policy makers*. P24. About Scratch. Open University 2015.
- Stefania Bocconi, Augusto Chioccariello, Giuliana Dettori, Anusca Ferrari, Katja Engelhardt *An analysis of educational approaches to developing Computational Thinking*. *Implications for policy and practice Brussels 2016 (in press)*