Computers in Human Behavior xxx (2017) 1–5

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

Exploring the computational thinking effects in pre-university education

Keywords:
Computational thinking
21st century competences
Coding
Pre-university education

A B S T R A C T

Several countries have usually adopted several priorities for developing ICT competences from kindergarten to secondary education. Most of them are focused on the development of key competences and/or coding skills. Although coding may be very attractive for young students and a very good practice or experience, it could be more interesting to develop students' logical thinking skills and problem-solving skills throughout programming approaches or computational thinking. This is a very exciting challenge with lots of possibilities regarding coding, robots, mobiles devices, Arduino-based application, game-based learning and so on. Thus, it is very important to explore the effect that these experiences have been taking into the pre-university students, both at primary and secondary education, with a special focus on the computational thinking as one of the components inside the toolbox to develop a reflexive and critical education in order to help children to solve problems using the technology with which they will live daily.

© 2017 Published by Elsevier Ltd.

1. Introduction

We live in a software-driven world (Manovich, 2013) and current Society demands skilled professionals for ICT (Information and Communication Technologies) business sector. A very common situation in countries with a high rate of unemployment is they have unfilled positions for engineers and technicians for the industry and digital services. This has caused an increasing approach for introduce digital or information technology (IT) literacy from the early beginning of the individual development (Bers, Flannery, Kazakoff, & Sullivan, 2014; Cejka, Rogers, & Portsmore, 2006; Kazakoff & Bers, 2012) till the high school courses (Allan, Barr, Brylow, & Hambrusch, 2010), even in post-secondary institutions (Astrachan, Hambrusch, Peckham, & Settle, 2009), combining it with other key competences such as reading, writing and math skills.

New devices (Alonso de Castro, 2014; Ramírez-Montoya & García-Peñalvo, 2017; Sánchez-Prieto, Olmos-Migueláñez, & García-Peñalvo, 2014), from smartphones and tablets to electronic learning toys and robots, find new audiences with increasingly young children (Fonseca Escudero, Conde González, & García-Peñalvo, 2017). This causes new challenges for teachers (Sánchez-Prieto, Olmos-Migueláñez, & García-Peñalvo, 2017), for example how to define developmentally appropriate activities and content for children of different ages (Bers et al., 2014).

The most frequent approach to teaching digital literacy has been to gradually encourage the learning of programming, and the term code-literacy (diSessa, 2000; Hockly, 2012; Vee, 2013) has been coined to referrer the process of teaching children programming tasks, from the simplest and most entertaining to the most complex, this way the student's progress is centered on the difficulty of the tasks and in their motivating characteristic.

However, this approach has epistemological antecedents in Papert (1980) works with Logo programming language, which promotes a constructionism rooted in Piaget (1954) constructivism that conveys the idea that the child actively builds knowledge through experience and the related "learn-by-doing" approach to education.

Consequently, at the same time that children learn human languages, both for speaking and writing, natural languages, encompassing all matters related with the experimental sciences (physics, chemistry, biology, etc.), and humanity languages, involving social sciences and humanities, it is also necessary they learn digital languages, in which ones of the competences to be success in the digital world are included, using coding as the way to solve problems and computational thinking as working paradigm (Llorens-Largo, 2015).

With the awareness of the importance of digital skills and related information technology (eSkills), there are several proposals worldwide about the need to include coding from the curriculum of non-university levels, starting since primary education (or sooner) (Balanskat & Engelhardt, 2015; Brown et al., 2013; García-Peñalvo, Llorens Largo, Molero Prieto, & Vendrell Vidal, 2017; Llorens Largo, García-Peñalvo, Molero Prieto, & Vendrell Vidal, 2017), because of the code-literacy skills are becoming understood as a core element for STEM (Science, Technology, Engineering, & Mathematics) subjects (Gelman & Brenneman, 2004; Weintrop et al., 2016), computational thinking may play an important role in K-12 STEM education because computational modelling is an effective approach for learning challenging science and math concepts (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009) and imaginative programming is the most crucial element of computing because it closely aligns mathematics with computing and, in this way, brings mathematics to life (Felleisen &

https://doi.org/10.1016/j.chb.2017.12.005
0747-5632/© 2017 Published by Elsevier Ltd.

Please cite this article in press as: García-Peñalvo, F. J., & Mendes, A. J., Exploring the computational thinking effects in pre-university education, Computers in Human Behavior (2017), https://doi.org/10.1016/j.chb.2017.12.005

Krishnamurthi, 2009).

A code-literate person means that can read and write in programming languages (M. Román-González, 2014), computational thinking is referred to the underlying problem-solving cognitive process that allows it. Thus, coding is a key way to enable computational thinking (Lye & Koh, 2014) and computational thinking may be applied to various kinds of problems that do not directly involve coding tasks (Wing, 2008).

2. What computational thinking is

Computational thinking has an increasing presence in the discussions about how to teach technology to pre-university students. However, there is not a consensus about what computational thinking is among computer scientists' community.

Jeannette M. Wing (2006) defined computational thinking as: "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science", with a very important message about this "computational thinking is a fundamental skill for everyone, not just for computer scientists". She revisited the topic and provided a new definition "Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2011).

Isbell et al. (2009) proposed a focus on providing services, interfaces, and behaviors that involve a more central role for modelling as a means of formulating relationships and identifying relevant agencies that are sources of change.

Moreover, Riley and Hunt (2014) asserted that the best way to characterize computational thinking is as the way that computer scientists think, the manner in which they reason.

Aho (2012) simplified this concept defining it as the thought processes involved in formulating problems so "their solutions can be represented as computational steps and algorithms".

García-Peñalvo (2016b) defined computational thinking as the application of high level of abstraction and an algorithmic approach to solve any kind of problems.

Barr and Stephenson (2011) provided an operational definition of computational thinking as a "problem-solving process that includes (but is not limited to) the following characteristics: formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; generalizing and transferring this problem solving process to a wide variety of problems".

Hemmendinger (2010) stated, the ultimate computational thinking should not be to teach everyone to think like a computer scientific nor to convert every child in a software engineer, but rather to teach them to apply these common elements to solve problems and discover new questions to explore within and across all disciplines. Close to this approach Sysło and Kwiatkowska (2013) also underlined that computational thinking is a set of thinking skills that may not result in computer programming, it should focus on the principles of computing rather than on computer programming skills.

And many more definitions and approaches to Computational Thinking (see more in (García-Peñalvo, Reimann, Tuul, Rees, and Jormanainen, 2016b)).

3. The core elements of computational thinking

Computational thinking really means an attempt to capture computing disciplinary ways of thinking and practicing. Thus, it is an active problem solving methodology where the students should use a set of concepts, such as abstraction, patterns matching, etc., to process and analyze data, and to create real or virtual artefacts. Computational thinking does not imply to use technology in a mandatory way to solve the problems, but it is oriented to student will be able to solve problems throughout the technology. For this reason, the goal of introducing ICT in pre-university curriculum is not the students become merely tool user but tool builders.

Different core computational thinking set of components are proposed to define specific computational thinking frameworks.

Barr and Stephenson (2011) present a structured model that emerged focused on identifying core computational thinking concepts and capabilities. The core concepts are data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization and simulation. The capabilities are computer science, math, science, social studies and language arts.

Brennan and Resnick (2012) propose a computational thinking where the components are classified into three dimensions:

1. Computational concepts, which are the concepts that students employ when they code: sequences, loops, events, parallelism, conditionals, operators, and data.
2. Computational practices, which are problem solving practices that occur in the process of coding: experimenting and iterating, testing and debugging, reusing and mixing, and abstracting and modularization.
3. Computational perspectives, which are the students' understandings of themselves, their relationships with others, and the digital world around them: expressing, connecting and questioning.

Gouws, Bradshaw, and Wentworth (2013) design a computational thinking framework to serve as foundation for creating computational thinking resources. This framework is a two-dimensional grid. One dimension gathers the skill sets that make up computational thinking: processes and transformations, models and transformation, patterns and algorithms, inference and logic, and evaluations and improvements. The other dimension means the different levels at which these skills may be practiced: recognize, understand, apply, and assimilate.

Zapata-Ros (2015) tries to connect computational thinking with the learning theories conceptualizations and thinking models, proposing the following computational thinking components: bottom-up analysis, top-down analysis, heuristics, divergent thinking, creativity, problem solving, abstract thinking, recursion, iteration, Successive approximation methods (trial and error), collaborative methods, patterns, synectics and metacognition.

TACCLE 3 Coding European project (García-Peñalvo, 2016a; García-Peñalvo, Rees, Hughes, Jormanainen, Toivonen, and Vermeersch, 2016a; TACCLE 3 Consortium, 2017) organizes its guidelines and resources over three main dimensions: the ability to interpret phenomena as computations (coding/programming), the ability to harness computations for solving problems (logical thinking), and the ability to design and control automation tasks (control technology).

4. Computational thinking practices

Due to computational thinking has different interpretations,

Please cite this article in press as: García-Peñalvo, F. J., & Mendes, A. J., Exploring the computational thinking effects in pre-university education, Computers in Human Behavior (2017), https://doi.org/10.1016/j.chb.2017.12.005

Download English Version:

https://daneshyari.com/en/article/6836294

Download Persian Version:

https://daneshyari.com/article/6836294

Daneshyari.com