Informatics in Education, 2019, Vol. 18, No. 1, 69–104
© 2019 *Vilnius University*
DOI: 10.15388/infedu.2019.04

# Lesson Planning by Computational Thinking Skills in Italian Pre-service Teachers

Lorella GABRIELE[*], Francesca BERTACCHINI,
Assunta TAVERNISE, Leticia VACA-CÁRDENAS,
Pietro PANTANO, Eleonora BILOTTA
*University of Calabria, Via P. Bucci, Cubo 17B, Arcavacata di Rende, Cosenza, Italy*
*e-mail: {lorella.gabriele, francesca.bertacchini, assunta.tavernise}@unical.it,*
*leticiavcec@yahoo.com, {pietro.pantano, eleonora.bilotta}@unical.it*

Received: May 2018

**Abstract.** In the last years, a growing trend in different educational contexts focused on Computational Thinking (CT) skills acquisition for both in-service teachers and students. But very low attention has been paid to pre-service teachers' education in regards to CT skills. To solve this issue, an empirical experimentation has been carried out with141 Italian pre-service teachers, that attended at a programming course, with the following aims: 1) provide them the main coding concepts by using Scratch 2.0; 2) offer practical advice on how to design educational applications (apps) to be applied into school context; 3) assess their apps by applying an already existing methodology, useful to give them feedback on their programming expertise and CT skills. Empirical findings showed that most of the participants achieved a medium-high level of CT skills, combining both design and programming skills in their school internship. Moreover, they reported a sense of greater self-esteem in teaching practice and a great emotional response from kids.

**Keywords:** computer uses in education, preservice teacher education, outcomes of education, coding, digital literacy.

## 1. Introduction

In contemporary society, digital literacy fosters the acquisition of Computational Thinking (CT) skills for everyone, allowing learners "to solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (Wing, 2006, pp 33). Broadly speaking, these abilities are also related to analytical thinking processes involved in modelling situations, designing and implementing systems. Nonetheless, the definition of these skills is still ongoing (Griffin, 2014). Similar approaches are adopted in the programming field (Tømte *et al.*,

[*] Corresponding author

2015, Kafai and Peppler, 2011), as well as in the Do-It-Yourself field (Guzzetti *et al.*, 2010; Bertacchini *et al.*, 2013; Bertacchini *et al.*, 2014; 2015; Bullock and Sator, 2015), in Digital Fabrication (Buechley and Eisenberg, 2008; Bilotta, Bossio, Pantano, 2010), and in Educational Robotics (Bilotta *et al.*, 2009; Bertacchini *et al.*, 2010; Gabriele *et al.* 2012; Gabriele and Bilotta, 2013; Gabriele *et al.*, 2017; Bertacchini, Bilotta, Pantano, 2017), just to cite a few. In many countries, CT formal courses in educational practice for both students and teachers have been carried out, investigating educational implications and assessment (`www.atc21s.org`). Hence, programming activities have been promoted for encouraging digital consumers to become educational "prosumers" (mixing "producers" and "consumers") (Bertacchini & Tavernise, 2014), thus entering in touch both with social media studies (Bertacchini *et al.*, 2018; Bertacchini *et al.*, 2018; Bilotta *et al.*, 1995) and contemporary complexity and chaos science (Bilotta, Lafusa, Pantano, 2003; Bilotta, Pantano, 2006; Bilotta, Pantano, Stranges, 2006; Bilotta *et al.*, 2007; Bilotta, Stranges, Pantano, 2007; Bertacchini *et al.*, 2007; Adamo *et al.*, 2010; Bilotta, Pantano, Vena, 2011; Bertacchini *et al.*, 2015).

*In this respect, it is quite interesting to discuss the Italian Digital Literacy Educational Reform and the pre-service teachers' educational context.*

In this work, the course on programming realized with preservice teachers, has been introduced into a curriculum that is not yet complete or that needs adjustments (Polenghi and Triani, 2014). Despite this, in 2014 the Italian Ministry of Education passed a law, according to which it is necessary that teachers of all levels learn to program in order to teach students the activities of Coding (Declaration N° 002937, September 23rd, 2014). Thus the need arose to make all pre-service teachers educated in High Order Thinking Skills or HOT Skills, an acronym synonymous with Computational Skills or CT (Heong *et al.*, 2012).

Whereas several studies affirm that pre-service teachers' technology-rich practices can be very beneficial for the use of programming in a school context (Coutinho, 2009; Lei, 2010; Vaca Cárdenas *et al.*, 2015; Bertacchini *et al.*, 2016; Vaca Cárdenas *et al.*, 2016; Bertacchini *et al.*, 2013; Bertacchini *et al.*, 2011), no Computer Science and Programming course for the Italian Primary Teacher Education Degree" has been scheduled. Digital literacy is instead acquired in an "Educational technology Lab course" by surfing online learning environments, searching the internet, using digital learning materials and Social Media and, finally, only a short introduction to Computational Thinking Skills. But, different authors found significant divide between the contents of "Educational technology lab course" and the use of these tools in real learning environments (Gunter, 2001; Angeli, 2004; Alì *et al.*, 2006; Gill and Dalgarno, 2008; Chai *et al.*, 2011; Tondeur *et al.*, 2012; Price *et al.*, 2012; Tondeur *et al.*, 2015; Baydas and Goktas, 2016; Blömeke *et al.*, 2016). The result is that pre-service teachers do not have a concrete education on programming, also fostering the idea that this activity is mysterious and difficult, or frustrating and boring (McInerney *et al.*, 2009).

To sum up, today, no specific educational program for pre-service teachers has been arranged to teach them "*how to teach*" HOTSkills or CT skills, and to provide them a satisfactory level of programming knowledge (Ertmer and Ottenbreit-Leftwich, 2010; Masterman and Manton, 2011; Moreno-León *et al.*, 2015).

To fill the gap that exists between the school education requirements in regards to "digital literacy and programming activities" and the current pre-service teachers' qualifications, some university courses need to be re-thought taking into consideration the Italian Educational Reform (Declaration N° 002937, September 23rd, 2014). Therefore, specific courses and laboratories for pre-service teachers should be implemented focusing on "Introduction to Programming" to provide them with an understanding of the role computation can play in solving problems and to help them feel justifiably confident of their ability to write small programs that allow them to accomplish useful goals. This educational path will support pre-service teachers in how to teach CT, how to include CT skills into teaching, and how to assess these abilities.

Starting from the specific problem – lack of effective teaching practices for digital literacy and programming – our work has the ambition to fill this gap by:

(i) Proposing a methodology to implement a Programming Laboratory to teach to novices' pre-service teachers how to develop an educational app.
(ii) Providing pre-service teachers with an educational methodology to use in their daily curriculum with their pupils (teach how to teach).
(iii) Combining two assessment methods, both found in literature, to measure the understanding and the use of different Computational Thinking concepts (What), and to detect the CT students' level (How) to identify their strengths and weaknesses.

The article starts by dealing with some key research carried on CT and Programming, looking at Scratch and its use in educational context methodologies found in literature to assess the apps (Section 2). Section 3 deals with the purpose of this work, the sample, the educational guidelines to implement the laboratory and the assessment methodology. Results are showed in Section 4 and discussed in Section 5. Finally, conclusions and further work are outlined in Section 6.

## 2. Previous Research

### 2.1. *CT and Programming*

According to both the International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA), CT skills include: problem formulation, logical organization and data analysis, data representation through abstractions, creation of computational automatic solutions, and problem generalization. As a result, they are a necessary prerequisite in digital society (Wing, 2006), and are critical to success (Lye and Koh, 2014). In fact, they imply a new and fundamental way of thinking (Curzon at al., 2014), involving thinking about a problem, coding, designing, analysing and applying some solutions to other contexts (Faraco and Gabriele, 2007; Barr *et al.*, 2016).

Federici, Gola, Brau, Zuncheddu (2015) highlighted how different countries included in their curricula computing, coding and Information Technology and how most teachers are not confident in this paradigm. These authors developed a new approach that allows teachers to approach coding without being necessarily experts.

Regarding CT Skills acquisition, various methods could be used to assess the acquired level of CT concepts in order to detect differences and advancements among students of various ages. As an example, the Progression of Early Computational Thinking model (Seiter and Foreman, 2013) could be considered useful to establish age-appropriate curricula, and to define lesson plans aligned with students' cognitive development stage.

Another method used to assess CT has been developed by Denner *et al.* (2012). It uses final products (e.g., games or models) to map students' computational thinking skill progression as marks of their higher level thinking, as they proceed through the curriculum. A similar approach was adopted by Repenning (2011) with Computational Thinking Pattern Analysis (CTPA).

"Dr. Scratch" developed by Moreno *et al.* (2015) carries out an automatic assessment of users' applications. As Repenning *et al.* (2015, p. 22) asserts, "the ability to automatically detect computational thinking patterns by analyzing student creations can give educators and learners greater insight into what concepts are understood and what concepts still need to be learned."

In regards to the arrangement of proper laboratories to teach CT skills at University level, various works introduce the general setting or qualitative results (McInerney *et al.*, 2009; Vaca-Cardenas *et al.*, 2015; Tavernise *et al.*, 2015). Indeed, pre-service teachers' University education on CT skills is fundamental for teaching in the elementary school, where children can efficiently implement computer games using customizable software developed for their level of experience (Baytak and Land, 2011). Furthermore, they can be acquainted with the following CT dimensions: computational concepts (concepts that programmer uses), computational practices (problem-solving practices that occur in the process of programming), and computational perspectives (students' understanding of their own learning, their relationships to others, and the technological context) (Brennan and Resnick, 2012).

Many studies involved the use Scratch with pre-service teachers programming courses. Bell, Frey, Vasserman (2014) reported the experience carried out by pre-service teachers, in a programming camp, as instructional team. They explored ways to incorporate Scratch into their daily curriculum, gradually acquiring skills in designing lessons into specific programming activities. Quan (2014) presented how students and pre-service teachers used computer projects into their lesson plans of second language teaching/learning for elementary, middle, or high school students. An and Lee (2014) describe an education course developed for pre-service teachers to allow them to understand and use computational thinking in their classrooms. Bean *et al.* (2015) underlined the benefit obtained applying their methodology, which improved learners' self-efficacy. Indeed, as Du Toit (2015) highlights, Information and Communication Technology (ICT) course for initial teachers is focused on computer literacy, but does not model effective teaching practices. Nevertheless, many other countries provide little or negligible teacher training related to ICT in education. For example, evidence from Europe shows that 70% and 65% of students in Lithuania and Romania, respectively, are taught by teachers for whom it is compulsory to participate in ICT training, compared to just 13% or fewer percentages of students in Luxembourg, Austria and Italy (European Commission 2013).

## 2.2. *Scratch: An Easy and Powerful Visual Programming Language*

In the last years, different easy-to-use visual programming languages for novice programmers were developed, among which Scratch, created by the Lifelong Kindergarten Group at the MIT Media Lab (Maloney *et al.*, 2008).

Scratch projects are composed of many little components called sprites; each sprite can be controlled by scripts. At the same time, each sprite has also one or more clothes to depict its visual status, and sound. Scratch 2.0 has a simple interface with a unique window with four panels; one panel has the command palate with buttons to select categories meanwhile the middle panel shows the scripts for the current sprite, with folder tabs to view and edit the costumes – images and sounds owned by that sprite. The panel on the upper right is the stage, where the action takes place. A button on the bar below the stage allows the stage to be displayed in full screen mode to show a finished project. The bottom-right panel shows thumbnails of all sprites in the project, with the currently selected sprite highlighted. Scratch is tinkerable: "tinkerability encourages hands-on learning and supports a bottom-up approach to write scripts where small chunks of code are assembled and tested, then combined into larger units" (Maloney *et al.*, 2010).

By using a drag and drop approach, novices can easily build a program, dragging command blocks from a palette into the scripting panel and assembling them to create "stacks" of blocks (Maloney *et al.*, 2010). However, in programming with Scratch, novices must understand some key building blocks categories. Blocks such as Statements, Boolean expressions, Conditions, Loops, Variables, Threads and Events, can be snap together only if they are syntactically appropriate. Moreover, certain blocks (e.g., conditions and loops) can be dynamically resized, to accommodate any number of nested blocks.

According to the international literature, Scratch software can be considered an attractive support for acquiring the skills linked to CT (Maloney *et al.*, 2008; Maloney *et al.*, 2010), providing both students and teachers with a chance to learn by doing (Vaca Cárdenas *et al.*, 2014; Tan and Kim, 2015).

Numerous papers confirm that the training and the educational activities based on Scratch "coding" can allow a fruitful production of digital contents (Wilson and Moffat 2010) as well as students' creative development through learning activities (Kobsiripat, 2015; Korkmaz, 2016) at different school levels, from elementary (Klaus-Tycho Foerster, 2016) to middle school students (Wolz *et al.*, 2011; Armoni, Meerbaum-Salant and Ben-Ari, 2015) and in adults (Chiu, 2014).

Moreover, Scratch promotes a friendly opportunity to coding due to the "remixing" approach. This characteristic refers to the possibility to use already available project on the Scratch Community, edit it, both modifying a part of the code and combining source materials, just giving credit to the original developer of the project (Monroy-Hernández, 2012).

## 2.3. *Designing Educational Apps with Pre-service Teachers*

While Italian teachers have an excellent pedagogical knowledge base, their role and skills in this rapidly evolving society, tied with the adoption of new technologies in the school has become a reasonable area of concern. In fact, it is not clear:

a. How to provide pre-service teachers with CT skills?
b. How to evaluate their acquisition?
c. How it is possible to develop CT in the teaching practice?

Furthermore, how to provide feedback to teachers has not yet been investigated. This point is very important for the education of pre-service teachers as it is about evaluating their programs, evaluating the acquired skills, and then considering a plan of CT further improvements.

The present work addresses these needs directly, by adopting an approach that allowed us to design first the teaching materials and then to evaluating teachers' products and skills acquisition.

The present work addresses these questions directly, by adopting an approach that allowed us to first design the teaching materials by using Scratch and then to evaluate teachers' products and skills acquisition.

The design of the app took place before doing an analysis of curricular contents, translating them into applications, which were then adopted in schools, linking the contents with the training results of children and pre-service teachers. The app design has been done through the following steps:

1. Analysis of curricular contents that are usually provided to children in primary school.
2. Choice of contents.
3. Application design.
4. Adoption of apps in schools, linking the contents with the educational outcomes both of children and pre-service teachers.

From the experimentation carried out in an empirical way, we realized that the products developed by the pre-service work and become really tools of knowledge for children if we carefully evaluate the steps of production of the programs, such as the cognitive articulation that led to the construction of the specific application. That is, if we evaluate the mental process carried out by the pre-service teachers in writing the various programs through the use of Scratch software. In this way, as several studies have shown, the results of understanding the programming processes implemented in the development of applications can directly give feedback on the effective construction of applications for educational purposes.

Therefore, the app design was evaluated by analysing the programs minutely, but specifically we evaluated:

a. The representation of the contents that the pre-service teachers have deployed in the applications and the quality and the didactic effectiveness of the applications; to achieve this goal, preservice-teachers must be able to create the agents of the application and be able to articulate events and phenomena in a system of temporal representation of events.

b. The levels of interaction that the pre-services have created to allow the subjects to interact with the knowledge bases represented in the first design step. Also for this purpose, the programming levels are certainly not simple. We expect advanced acquisition of major programming concepts to lead to high levels of interaction because pre-service teachers are able to develop many games and exercises to allow students to manipulate the concepts they are dealing with.
c. Kids 'learning assessment, by evaluating the performance of the various exercises / games proposed by the pre-service teachers.

From here on, a discussion follows on how to evaluate applications in the education sector. Good levels of teacher programming are also required for the implementation of learning assessment systems. But this is truly a challenge because it allows not only an assessment of the content you want to acquire, but also a measure of the performance of children in the use of the application.

From this analysis, we will choose some of the most appropriate methods to meet the needs of the pre-service teachers.

By analyzing which assessment methods best fit the needs of the pre-service teacher population and adopting some of them, we can provide a method that may be followed later. The second motivation on how to provide practical feedback to pre-service teachers on the achieved results. Knowledge of the level of programming achieved will enable them to better manage their future development.

In the following paragraph, we will present a non-exhaustive overview of the evaluation methods and those that seemed to us the most useful to provide effective feedback to teachers. Please, note that the choice was made based on the simplicity of the adopted methods and the clarity of the results obtained. The focus was to provide teachers with practical self-assessment tools. We will illustrate how these different methods are used (or in what contexts they are appropriate), and how they differ from each other to show that the two particular valuations methods we have chosen (i.e., Denner, Werner, Ortiz, 2012; Moreno *et al.*, 2015) are better fit with the than other methods for the purposes mentioned above. In this way, it will be possible to assess how appropriate their use is for the reported study.

## 2.4. *Evaluating Methods*

Different assessment methods are present in literature (Maloney, Peppler, Kafai, Resnick & Rusk, 2008; Denner, Werner, Ortiz, 2012; Brennan and Resnick, 2012; Meerbaum-Salant, Armoni, (Moti) Ben-Ari, 2013; Moreno *et al.*, 2015; Repenning, 2011; Koh, 2014). Some of these methods (Maloney, Peppler, Kafai, Resnick & Rusk, 2008; Denner, Werner, Ortiz, 2012) aim at identifying which programming language concepts are employed by learners to develop an application. If the student correctly uses some commands or blocks, it means he/she has acquired the mental model underlying the use of functional blocks. In 2008, Maloney, Peppler, Kafai, Resnick, & Rusk analysing the use of Scratch commands by novice learners, stated that the use of certain blocks in a given project is a proof that the related programming concept has been learnt. The authors evaluated the presence of the following functions of a coding language: User Interaction; Loops; Conditional Statements; Communications and Synchronization; Boolean Logic Variables; Random Numbers.

A similar method has been developed by Denner, Werner, Ortiz (2012). The authors note that learners' apps differ in their complexity, according to different users' mental models and cognitive profile. The authors provided a list of all commands and blocks, clustering them into three main categories of programming concepts that can be learned by using Scratch. In turn, each of these categories are organized in 17 subcategories: the first category, *Programming concepts*, is intended as the internal management of the program, through local and global variables, the use of parallelism functions and of conditional character interactions, the structure of the events happening in the apps, user interaction. The second category, *Code organization and documentation,* is concerned to the items related to the name given to the variables and to the general comments given by the user to the code. The third category, *Designing for usability*, takes into account the character features and its own main goal and functionality. Scores of 1 or 0 are assigned if each item is present or not in the developed program. Brennan and Resnick (2012) developed a more sophisticated approach. This method does not assess learners' basic programming concepts embodied into the programs. Instead, it evaluates any changes occurred in different versions of an app. Subsequently, authors interviewed the learners. From this, a complex evaluation model came out, with three steps: Approach #1 – Project Analysis: the collection of the releases of a project (how a project evolved over time), the analysis of the Scratchers' portfolio (online community) and a deep examination of a single final project. Approach #2 – Artefact-Based Interviews: to assess the development of computational thinking, interviewing the learners directly. Approach #3 – Design Scenarios: to test the conceptual understanding of how externally selected projects influenced the app under examination.

Meerbaum-Salant, Armoni, (Moti) Ben-Ari's (2013) evaluation method uses quantitative tools, integrated with a qualitative analysis based on observations carried out in educational settings. In particular, the authors, combining the revised Bloom's Observed Learning Outcome taxonomy, developed a new taxonomy composed by three super-categories – unistructural, multistructural, and relational. Each super category contains three sub-categories – understanding, applying, and creating. "Understanding is interpreted as the ability to summarize, explain, exemplify and classify Computer Science concepts (among which are programming constructs), and to compare them. Applying is interpreted as the ability to execute algorithms or code: to track them and recognize their goals. Creating is interpreted as the ability to plan and produce programs or algorithms" (Meerbaum-Salant, Armoni, (Moti) Ben-Ari, 2013, p. 72).

Other automatic assessment tools, such as the Computational Thinking Pattern Analysis (CTPA) (Repenning, 2011; Koh, 2014) and the open-source web application, developed by Moreno *et al.* (2015) are very interesting and easy to apply by non- expert. The CTPA is a learning data tool useful to measure student-learning skills and represents their achievements semantically through a phenomenological analysis in real-time. The outcomes of CTPA can be used to provide valid and useful educational feedback to educators and learners to measure and track student-learning outcomes. Dr. Scratch

(Moreno *et al.*, 2015) is a free/open-source web application developed to easily and automatically analyse Scratch projects, as well as to obtain feedback that can be used to improve programming skills. Dr. Scratch allows an automatic confirmation of the presence of 7 parameters (Abstraction, Parallelism, Logic, Synchronization, Flow control, User interactivity, Data representation), assigning a score from 1 to 3 to each parameter. Basic, Developing or Proficiency (Master) level is attributed to the app project, depending on the total final score. The ITCH system developed by Johnson (2016) is devoted to automated testing of small Scratch programs. However, it is not aimed at projects such as games, where continuous control is needed to navigate through the project. ITCH can aid students in the final development of more complex and creative assignments. Table 1 summarizes, in chronological order, the main features of the methods.

Table 1
App assessment methodologies

| | Assessment Criteria | Used in the context such as |
|---|---|---|
| Maloney, Peppler, Kafai, Resnick, Rusk (2008) | Evaluation of the following parameters:<br>• User Interaction<br>• Loops<br>• Conditional Statements<br>• Communications and Synchrony<br>• Boolean Logic, Variables<br>• Random Numbers | Computer Clubhouse (South Central Los Angeles) with youth ages 8–18 |
| Ericson, McKlin (2012) | Use of a pre and post questionnaire to evaluate which computing concepts students learned | Georgia Tech's camps reach a large range of students with 4th –12th grade |
| Denner, Werner, Ortiz (2012) | Three main categories and 17 subcategories.<br>1. Programming Concepts that foresee:<br>• Parallelism<br>• Use of random<br>• Variables<br>• Conditional character interaction<br>• Conditions or events<br>2. Code organization and documentation that foresee:<br>• Extraneous rules<br>• Character names<br>• Variable names<br>• Rule names<br>• Rule comments used<br>• Rule grouping<br>• Rule boxes<br>3. Designing for usability Incorporates that takes account of:<br>• Themes<br>• Character appearance<br>• Linking of stages<br>• Play Instructions clear<br>• Goal, Clear Functionality<br>Scores 1 or 0 are given according to the presence or the absence of each item | Voluntary after-school program focused on computer game programming with students that attended 6th grade |

Continued on next pagt

Table 1 – continued from previous page

| | Assessment Criteria | Used in the context such as |
|---|---|---|
| Brennan and Resnick (2012) | Tree assessment approaches:<br>**Approach #1** – Project Analysis, Collection of all releases of a program<br>**Approach #2** – Artefact-Based Interviews<br>**Approach #3** – Design Scenarios | Useful to assess the development of computational thinking in young people who are engaging in design activities with Scratch |
| Meerbaum-Salant, Armoni, (Moti) Ben-Ari (2013) | Three super-categories:<br>• unistructural<br>• multistructural<br>• relational<br>Each super-category contains three sub-categories – understanding, applying, and creating | Used with middle-school students during normal school hours and were taught by middle school teachers |
| Repenning (2011) Koh (2014) | Learning data tool useful to measure student-learning skills and represents their achievements semantically through a phenomenological analysis in real-time | Used with middle school and college student |
| Moreno *et al.* (2015) | Automatic confirmation of the presence of the following 7 parameters:<br>• Abstraction<br>• Parallelism<br>• Logic<br>• Synchronization<br>• Flow control<br>• User interactivity<br>• Data representation<br>Scores are from 1 to 3 for each parameter. The total assign the label of Basic, Developing or Proficiency (Master) level to the app | Used with students aged between 10 and 14 |
| Johnson (2016) | Automatic testing system for projects in the Scratch programming language | Used with university's pre-major Computer Science course to assess only small projects |

From the presented review of apps evaluation methods, we can say that all methods are excellent for assessing programs, as they make account for different aspects of the complex programming processes each learner exhibits while programming. As our sample of pre-service teachers are beginners, with no prior knowledge about programming concepts, we chose to adopt two different methodologies to assess the apps because the first one, developed by Denner *et al.* (2012), allows to identify **"What"** CT concepts (see Table 6) were learned by novice programmers; the second one, implemented by Moreno *et al.* (2015), allows to evaluate the *competence level* so **"How"** those CT concepts (see Table 7) were mastered by pre-service teachers.

Furthermore, meetings with pre-service teachers have also been made in order to give them feedback on their work, with the aim to allow a deeper analysis of their programming skills. By combining these two methods, we were able: 1) to measure the understanding and the use of different Computational Thinking concepts, explained directly by the learners in order to solve problems or particular events or variables they want to implement, and 2) to detect the CT students' level of understanding in order to identify their strengths and weaknesses.

## 3. Methodology

### 3.1. *Purpose of the Study*

This research aimed at the investigation of the CT skills gained by 141 pre-service teachers, who were novice programmers, after a laboratory on Scratch software.

We used two different methodologies to assess the Scratch projects (see section 2.3) in order to detect:

a. The *CT Skills learned (**What**)*, analysing the apps through Denner's criteria (Denner *et al.*, 2012).
b. The *competence level of CT skills (**How**)*, using Dr. Scratch Software (Moreno *et al.*, 2015).

Firstly, the work was guided by the following research questions:

1. *What are the programming concepts that pre-service teachers gained in developing an app?*
2. *What is the level of acquisition of CT skills?*
3. *How pre-service teachers perceived themselves as regards their programming knowledge level at the end of the laboratory?*

### 3.2. *Participants*

This study targeted 141 pre-service teachers (undergraduates) (F = 128; M = 13) enrolled at the, the curriculum for "Primary Teacher Education" (5-year Combined Degree courses. Students obtain a qualifications of Teaching for Early Childhood and Primary), at the University of Calabria (Italy).

Participants, aged between 18 and 40 (M = 26; SD = 5.93), freely formed 40 groups (5 groups = two participants; 13 groups = three participants; 18 groups = four participants; 4 groups = five participants).

In this degree course, pre-service teachers usually develop projects in team for different disciplines. In some cases, teams reflect students' preferences or demographic need (residence location). Hence, the different number of participants in each group reflects the social dynamics already existing, and the choice of the researchers not to interfere with these spontaneous dynamics. However, each group have a common goal, but each individual is stimulated to explore this goal according to his or her own context, teaching situations, individual beliefs, and so on.

An *initial demographic survey* collected information on age, sex and schooling. Sample self-evaluated their Digital Literacy background and their level of familiarity with technologies. Results showed that a very high percentage of participants had a humanistic high school background (see Table 2) that do not foresee a compulsory use of technology. Therefore, the majority of participants (93%) had a very low level of digital literacy.

Table 2

Participants' school background

| Italian secondary school | N° of participants | % of participants |
|---|---|---|
| High School Diploma in Classical studies | 79 | 56% |
| High School Diploma in Scientific studies | 10 | 7% |
| High School Diploma in Foreign Languages | 11 | 8% |
| High School Diploma in Pedagogical studies | 45 | 29% |

Table 3

Self- esteem on the acquired skills

| | |
|---|---|
| **Pre-service teachers career development** | |
| **Q1** | Personal satisfaction |
| **Q2** | Personal development |
| **Q3** | Pedagogical knowledge improved after CT skills acquisition |
| **Q4** | High capability to design educational stuff after CT skills acquisition |
| **Pre-service teachers teaching CT skills** | |
| **Q5** | Expertise improvement in teaching elements of CT to children |
| **Q6** | Children's motivation increase knowledge acquisition through technology |

At the end of the research, the students were asked to answer to the following question: "How do you perceive your own Programming knowledge after the Laboratory experience? Please, choose among "None knowledge", "Beginner", "Expert".

A survey of the pre-service teachers on self-esteem on the acquired skills, personal satisfaction and their improved pedagogical knowledge after CT skills acquisition was administered (see Table 3).

A four-point Likert items was used where respondents can choose among "Very good", "Good", "Fair" and "Poor".

## 3.3. *Didactic Guidelines to Implement the Laboratory*

Research had a duration of 10 months and consisted of four steps. A detailed description of the planned research activities is showed in Table 4.

In the first step, the experimental research plan was set up (i.e. activities to carry out, methodology to adopt) and the lecture material was arranged (power point presentations, introductory videos of the software, booklets, ad-hoc built apps as examples). This step had a duration of four months.

In the second step, we started with a pilot research that involved a small sample of participants in order to test the arranged material and to improve the experimentation, if the case. This step had a duration of three months.

Table 4
Planned timeTable of the laboratory

| Lectures | Lectures Overview | Didactic Material |
|---|---|---|
| **Week 1**<br>Laboratory activities overview: research aim and Scratch introduction | General overview on Laboratory activities: rules and setting<br>Scratch introduction: functionalities and application on Scratch 2.0<br>Guidelines on how to create an app | Power point presentations:<br>- Programming introduction<br>- General concepts: What is an algorithm? Structured programming<br>- Scratch and its functionalities<br>- Example of an app developed with Scratch<br>Video:<br>- Step by step analysis of an application made with Scratch |
| **Week 2**<br>Focus group | Students were given instructions on how to work in a team and how to develop the application. | Booklet with the instructions on how to work. |
| **Week 3**<br>Coding | Students were given instructions on how to create a didactic app (game, story, etc.).<br>Instructors explained them how to make the report and the app user's manual. | |
| **Week 4–8**<br>Designing and implementation of the didactic app | Students worked in groups with the instructors support. | |

In the third step, an eight weeks' laboratory on programming with ***Scratch 2.0*** was carried out. First, participants had to follow theoretical lectures on programming with Scratch 2.0 (four face-to-face lessons, two-hours each). Then, they had to design and develop an educational app. This step had a duration of two months. In the fourth step, the collected data was analysed (one month).

As above mentioned, during the third step the laboratory on programming with Scratch 2.0 was carried out. The laboratory activities had three phases (see Table 5):

1) Decision-making phase.
2) Implementation phase.
3) Follow-up phase.

In the decision-making phase, a brainstorming allowed the exploration and elaboration of the focus of the designed app; the feasibility of the project and its main objectives were analysed. User requirements were defined with the support of the official document "Italian National suggestions for the curricula" (`www.indicazioninazionali.it/documenti_Indicazioni_nazionali/indicazioni_nazionali_infanzia_primo_ciclo.pdf`). User requirements definition included: age of the final user; attended classroom; users' prerequisites (knowledge background).

Table 5

Procedure laboratory phases

| Laboratory phases |
|---|
| **1) Decision-making phase**<br>**A focus group session was carried out in order to:**<br>• Identify each team members' role<br>• Define the educational objective<br>• Define the user requirements (children age related to skills)<br>• Design the app functions |
| **2) Implementation phase**<br>This phase is subdivided into a development step and a testing step (the app was tested with end-users in a didactic context, in order to verify if is really user-centred) |
| **3) Follow-up phase: Report and handbook (guide) app**<br>In the report, each workgroup had to:<br>• List of the roles in the team<br>• Describe the App educational objectives of the app<br>• Report about User requirements<br>• End-users (children) age and possible skills<br>• Outline of the decision tree realized during the work by using a diagram<br>Each group had to assembly a user handbook (guide) app to explain the app function with images and text in order to allow to other people to run the app correctly and eventually to use the handbook app in educational context |

In the implementation phase, participants developed the app projects using Scratch software. Afterwards, the results were evaluated according to the list of requirements defined in the decision-making phase.

At last, the follow-up phase foresaw the writing of a report and a handbook app (guide on how to use the app) that provides instructions and help for end-users.

Summing up, each workgroup in developing their app has to follow specific methodological guidelines:

1) To choose a topic of a subject area (i.e. mathematics, science, foreign language).
2) To develop a user-centred app, taking in consideration the user requirements.
3) To use a storyboard to design the app.
4) To implement the topic with a high level of interactivity, by using for example interactive whiteboard or the webcam.
5) To test the app with the end-users.

## 3.4. *Assessment Method Adopted to Evaluate Reports and Apps*

Two kinds of the collected data were analysed:

1) The Report elaborated by each team.
2) The apps source files.

The following key information were extracted from the report: the main topics of the implemented app, the targeted age of the final user chosen by participants. Moreover, the presence in the report of the following characteristics was assessed:

1) The problem formulation and the didactic objective of the app (Goals).
2) The user requirements consideration.
3) A clear explanation of the steps to follow in order to run the app correctly. This information was included in the handbook app.

Regarding the app source files, they were analysed to detect the *CT Skills learned* (***What***) according to Denner's (Denner *et al.* 2012) and the *level of acquisition of CT concepts* (***How***) (Moreno *et al.* 2015).

Denner's criteria (see Table 6) define the categories of programming concepts that can be learned by using Scratch: Programming concepts; Code organization; Designing for usability (17 subcategories identify the three categories).

Table 6

CT skills assessment: categories and definitions for Scratch's projects (Source: Authors adapted schema created by Denner *et al.*, 2012)

| Categories and definitions for scratch's projects assessment | |
|---|---|
| Category | Definition |
| PROGRAMMING CONCEPTS | |
| 1. Sequence | Are the blocks in a systematic order to execute the program correctly? |
| 2. User interaction (e.g. Keyboard input) | Using blocks such as ask and wait prompts users to type in an answer |
| 3. Iteration (Loops) | Using loops forever and repeat to create iterations. |
| 4. Variables | Variables can be created within Scratch and then be used within programs. |
| 5. Conditional statements | Using if, forever if and if-else to check for conditions. |
| 6. List (arrays) | Allows for storing and accessing lists of strings and numbers. |
| 7. Coordination and synchronization (Parallelism) | Using blocks such as wait, broadcast and when I receive to coordinate the actions of multiple sprites. |
| 8. Random numbers | Pick Random is used to select random integers within any given range |
| 9. Boolean logic | Using and, or, not. True or false. |
| CODE ORGANIZATION | |
| 10. Extraneous blocks | Are there scripts, any blocks that are no initialized when the program is run? |
| 11. Sprite names (the default is overridden) | Are the sprite names rewrote or are used the default names? |
| 12. Variables names | Are the variables given meaningful names when set up? |
| DESIGNING FOR USABILITY | |
| 13. Functionality | There are few or no faults in the programming. |
| 14. Sprite customization | Is the sprite used a predefined sprite available in the library of the program or it has been customized by the developer? |
| 15. Stage customization | Is the stage used a predefined stage available in the library of the program or it has been customized by the developer? |
| 16. Clear instructions | Has the student defined how the game is supposed to run? |
| 17. App originality | Students create their own app according with the goal? |

With reference to the score, 1 point was attributed to the presence of a subcategory in the project, and 0 to the absence, for a maximum of 17 points.
For individuating the acquired *competence level* of CT skills (***how***), we used Dr. Scratch demo software (Moreno *et al.*, 2015). It allows an automatic confirmation of the presence of the following seven parameters:

1) **Abstraction and problem decomposition**: the ability of abstraction and decomposition of problems helps to break a problem into smaller parts that are easier to understand, program and debug (`http://www.drscratch.org/learn/Abstraction/`). For example, the character behaviour is controlled by different programs and each of these programs (decomposition of problems) deal with a particular issue.
2) **Parallelism**: is the possibility that different things happen simultaneously. Example: two characters execute an action while a character does several things at once.
3) **Logic**: permits to get dynamic projects; so that, they perform differently depending on the situation.
4) **Synchronization**: instructions associated to synchronization allow the characters to organize things to happen in the order we want. For example, "wait" ca be used to synchronize two characters to maintain a conversation, or "Wait until, when backdrop change to, broadcast and wait".
5) **Flow control**: means instructions related to algorithmic concepts of "flow control" to control the behaviour of the characters.

Table 7
Competence level for each CT concepts (source: Moreno *et al.*, 2015)

| CT Concept | Competence Level | | | |
|---|---|---|---|---|
| | Null (0) | Basic (1 point) | Developing (2 points) | Proficiency (3 points) |
| Abstraction and problem decomposition | - | More than one script and sprite | Definition of blocks | Use of clones |
| Parallelism | - | Two scripts on green flag. | Two scripts on key pressed, on sprite clicked on the same sprite | Two scripts on when I receive message, create clone, two scripts with conditionals. Two scripts on when background change to. |
| Local thinking | - | If | If else | Logic operations |
| Synchronization | - | Wait | Broadcast, when I receive message, stop all, stop program, stop | Wait until, when background change to, broadcast and wait |
| Flow control | - | Sequence of blocks | Repeat, forever | Repeat until |
| User interactivity | - | Green flag | Key pressed, sprite clicked, ask and wait, mouse blocks. | When %s is >%s, video, audio |
| Data representation | - | Modifiers of sprites properties | Operations on variables. | Operations on lists |

6) **User interactivity**: instructions that can help the Scratch projects to be more interactive. For ex-ample, the use of the keyboard or the mouse to move a character, to answer questions.
7) **Data representation**: Scratch projects need a set of information about the characters to run appropriately. Each character has a number of attributes. In addition, users can modify the attributes of the characters by using variables or lists to store information.

In particular, the software (Moreno *et al.*, 2015) assigns a score from 1 to 3 to each of the above parameters. A Basic, a Developing or a Proficiency (Master) level is attributed to the app project depending on the total final score (see the details in Table 7).

## 4. Findings

In this section, we present the findings and answer the research questions sketched in section 3.1.

We analyse 40 Scratch applications on Elementary School topics, developed by pre-service teachers. Each app deepened a specific topic (see Table 8) and the groups of pre-service teachers developed their own apps taking into account the age, the attended class, as well as the knowledge background of the final user.

The target age of the final users varied from three to seven years. In particular, 13 groups developed an app for final users aged between 3 and 5 years; 16 groups for a target aged between 6 and 7; 7 groups for a target aged between 8 and 9; and 4 groups developed an app for a final user aged between 10 and 11 years. Fig. 1 concerns the "Wash your teeth" app: (a and b) are some screenshots of the app and (c) concerns a photo of children that interact with the application, while testing the app's usability.

Table 8
Topics of the apps

| Number of Apps | Topic |
|---|---|
| 4 | Recognize and name the colours |
| 6 | Mathematics |
| 5 | The alphabet (Italian and English) |
| 12 | English |
| 1 | Days of the week |
| 3 | Seasons and fruits |
| 1 | Visual attention and reaction speed games |
| 3 | Geography |
| 1 | Music |
| 1 | Sports |
| 1 | Education |
| 1 | Emotions |
| 1 | Science |

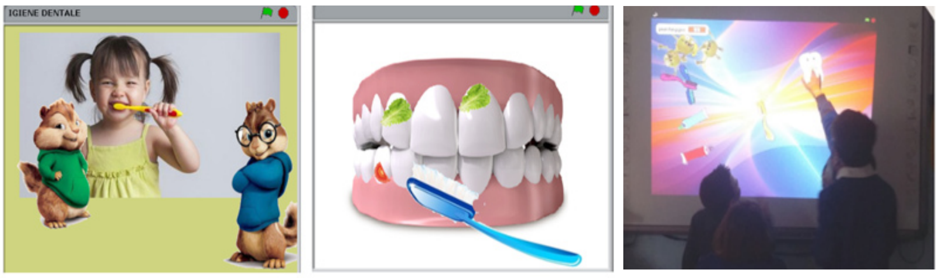
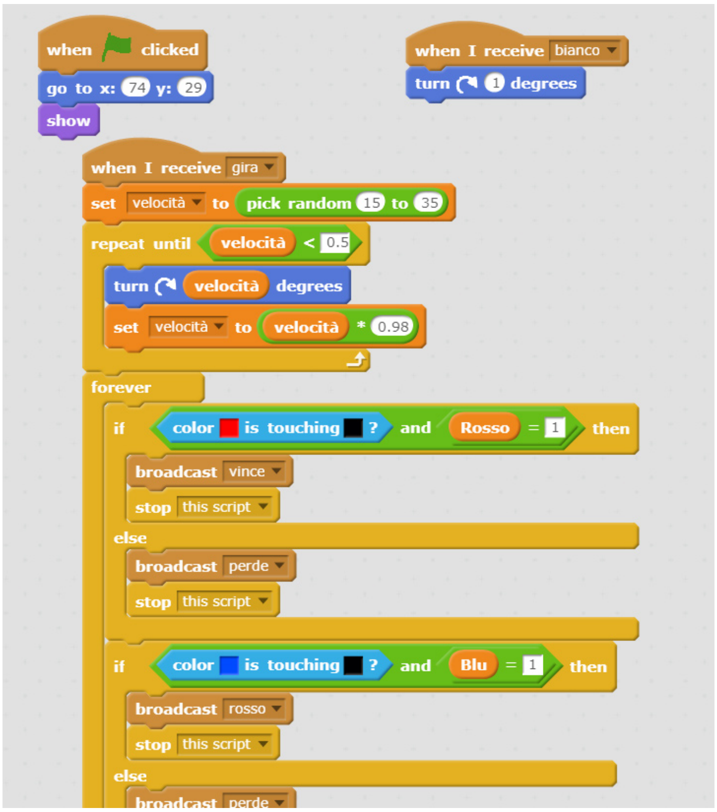


Fig. 1. Three pictures related to the app “Wash your teeth“. a) shows how to correctly clean teeth. b) shows an interactive game. By using an interactive whiteboard, children can clean teeth with handling the toothbrush. c) shows another interactive game, children obtain a point for each good element (toothbrush, toothpaste) collected.

Fig. 2 show one of the script of *Lucky wheel* app. The app includes Flow control, Data representation, Abstraction, User interactivity, Synchronization, Parallelism. Students duplicated two scripts, used variables, and defined two sprite attributes.



```
when [flag] clicked
go to x: 74 y: 29
show

when I receive bianco
turn ↻ 1 degrees

when I receive gira
set velocità to pick random 15 to 35
repeat until velocità < 0.5
    turn ↻ velocità degrees
    set velocità to velocità * 0.98
forever
    if color ■ is touching ■ ? and Rosso = 1 then
        broadcast vince
        stop this script
    else
        broadcast perde
        stop this script
    if color ■ is touching ■ ? and Blu = 1 then
        broadcast rosso
        stop this script
    else
        broadcast perde
```

Fig. 2. The script of *Lucky wheel* app.

### 4.1. *Results of CT Skills Learned by Novice Programmers: Programming Concepts, Code Organization Designing for Usability*

Analysing the report and the handbook (guide) app that each group delivered at the end of Laboratory, we collected some useful evidences on the process of app design and development.

We found that all teams started the work by clearly identifying the "Problem to solve" (the topic that each app deepened): 97.5% of the teams detailed the Goal of the app (specific knowledge to delivery through the app). All workgroups took into account the *user requirements*. They collected images, sounds, voice records, customized pictures and all the digital materials useful to assembly the final app. At last, 80% of the teams fulfilled a complete user handbook app, defining how the game is supposed to run, specifying and explaining all the apps' functionalities; hence, 95% of the apps were equipped with clear instructions.

### 4.2. *What are the Programming Concepts that Pre-service Teachers Gained in Developing an App?*

The implemented forty applications were analysed to detect the *CT Skills learned* (***What***), according to Denner's (Denner *et al.* 2012) criteria.

Those criteria refer to Programming concepts, Code Organization and Design for Usability. Table 9 shows all the results' analysis.

Table 9

Results of CT skills learned by novice programmers: programming concepts, Code Organization Designing for usability

| Scratch's projects assessment | % |
|---|---|
| **Programming Concepts** | |
| 1. Sequence | 100 |
| 2. User interaction (e.g. Keyboard input) | 88 |
| 3. Iteration (Loops) | 78 |
| 4. Variables | 20 |
| 5. Conditional statements | 63 |
| 6. List (arrays) | 0 |
| 7. Coordination and synchronization (Parallelism) | 83 |
| 8. Random numbers | 8 |
| 9. Boolean logic | 13 |
| **Code Organization** | |
| 10. Extraneous blocks | 5 |
| 11. Sprite names (the default is overridden) | 25 |
| 12. Variables names | 20 |
| **Designing for Usability** | |
| 13. Functionality | 97.5 |
| 14. Sprite customization | 72.5 |
| 15. Stage customization | 82.5 |
| 16. Clear instructions | 95 |
| 17. App originality | 97.5 |

In particular, as regards the "Programming concepts", 100% of the projects used Sequences, 78% Loops and 88% User Interaction. The apps developed by different teams had different programming complexity. In particular, Communication and Synchronization commands were used significantly (83%). These commands have a critical role when users build more structured and complex projects in Scratch, since they allow to switch from a functionality to another. Moreover, Boolean operations (13%), Variables (20%), and Random numbers (8%) are concepts that undoubtedly are difficult to learn on their own. 63% of the apps had Conditional statements. These blocks are used, for example, when the sprite has to move around and if it touches an object then it causes an event, i.e., the sprites rebound borders or change stages.

Even though all these apps are runnable, only 5% of them include extraneous blocks (any blocks that are no initialized when the program is run): these bugs do not affect the general app functioning. For example, the app can run even if the programmer forgot some unrunning blocks like the comments.

In 25% of the apps the default sprite name was changed; 20% of the projects included variables, and all of them had meaningful variables names (name linked to the functionality).

In their own reports, novice programmers stated that they tested many times their own app with children, in order to develop a user-centred app, according the user requirements. This statement is supported by the results' analysis for the category "Designed for usability".

Findings show that 97.5% of the apps had a high level of functionality and were developed according to the goal. This criterion measures the originality of an app.

Moreover, sprites (72.5%) and stages (82.5%) were customized; hence, programmers did not use predefined sprites or stages, but accurately sketched these elements.

## 4.3. *What is the Level of Acquisition of CT Skills?*

The *level of acquisition of CT* concepts (***How***) gained by pre-service teachers was measured by analysing the 40 developed applications with Dr. Scratch demo software.

According to Dr. Scratch method, a score was assigned to each of the seven CT concepts present in the app and each programmer can reach from 0 to 21 points (Moreno *et al.*, 2015).

As showed in Table 10, the most used coding concepts were flow control (61.67%) and user interactivity (60.83%). It means that the projects have a sequence of instructions according to specific conditions and foresee users' interaction.

Parallelism was used by 48.33% of the apps, which means that several sprites (characters) were activated at the same time.

The other used CT concepts are: synchronization (45.83%); logic concepts (38.33%) (and, or, not), including conditional commands (if, if – else); data representation (37.50%); and abstraction (33.33%).

Dr. Scratch demo software takes also into account some bad habits of programming or possible errors, such as non-significant sprite names, repetition of code, code never executed and the incorrect initialization of object attributes.

Table 10
Competence level of the scratch projects

| CT concept | % |
|---|---|
| Flow Control | 61.67 |
| Data Representation | 37.50 |
| Abstraction | 33.33 |
| User Interactivity | 60.83 |
| Synchronization | 45.83 |
| Parallelism | 48.33 |
| Logic | 38.33 |

Table 11
Dr. Scratch assessment

| Level | % of projects |
|---|---|
| Basic | 30 |
| Developing | 60 |
| Master | 10 |

According to the points obtained in each CT concept (see Table 7), a programmer can reach the following levels: *Basic; Developing*; *Master*.

The level gotten mirrors the CT concepts used from the user in the app. Hence, developers can try to improve their projects using the guidelines and tips offered by the tool. The 30% of the developed apps were "Basic", 60% were "Developing" and 10% were "Master" (see Table 11).

### *Final examination results*

The impact of the course of Computational thinking on the skills of pre-service teachers has been successful. In fact, most of the pre-service teachers gained a very high quality performance.

In fact, at the end of this course, marks were assigned to each pre-service teacher according to the level of knowledge and skills acquired. In particular, 21% of pre-service teachers received an "Excellent" mark, 43% received a "Very good" mark; 22% received a "Good" mark; 6% received a "Discrete" mark and only 8% received a "Sufficient" mark on the final examination.

## 5. Discussion

Curzon *et al.* (2014) discussed that the CT skills acquisition (framework) encloses four interconnected stages, were "Assessment" is the fourth one, after "Definition", "Concepts" and "Classrooms techniques". Moreover, they underlined the importance of assessing the increasing competence of learners in CT and mapping the learning outcomes.

The twofold assessment methodologies that we adopted (Denner *et al.* 2012; Moreno *et al.* 2015) in order to evaluate the developed applications, sketched a composite framework with specific indicators that allowed to assess "What" and "How" coding concepts have been understood. In particular, the 40 didactic apps implemented by the pre-service teachers were assessed according to Denner's (Denner *et al.*, 2012) and Moreno's (Moreno *et al.*, 2015) methodologies. While the first one allows to detecting the CT concepts acquired by pre-service teachers through the designing and the implementation of their applications (***What***), the second one allows to finely ascertain their CT level competence (***How***).

The adopted didactic methodology, aimed to motivate pre-service teachers, allowed us to achieve a very good result. Our sample composed of 141 pre-service teachers were able to implement 40 applications (two examples are described in Appendix).

In a very short time, pre-service teachers acquired both basic and advanced programming concepts. The designed Laboratory methodology provided learners with specific suggestions and detailed methodological steps to follow: STEP 1 – Decision-making: they defined the educational objectives, the user requirements (children age related to skills) and designed the app functions; STEP 2 – Implementation phase; STEP 3 – Testing phase: they tested the app with final users in a didactic context, in order to verify if it is really user-centred.

During each step, in case of difficulties or doubts, learners could rely on the researchers involved in the laboratory activities. In this way, they had the certainty to receive the necessary support or feedback or encouragement to go on with the assigned work. In this way, the risk was that learners were more confident in the suggestions of the researchers than in their abilities. That is why we chose to guide them to find new solutions and never provide just the answer they are looking for.

However, regarding the first research questions (section 3.1) "*What are the programming concepts that pre-service teachers gained in developing an app*?", we found that, after the face-to-face lessons, participants discovered and used sequence, user interaction, loops, conditionals, and finally communication and synchronization in different scripts. The fact that some blocks were correctly used in a project means that a programming concept was understood and learned. Then, the use of variables, Boolean logic, and random numbers was less common but two teams (according to the findings) learned to use these concepts over time. We found that arrays (Data menu), absolute value and, square root (Operators menu) never appeared in the projects since these are advanced CT concepts.

One of the main characteristics of the Scratch approach is the projects "remixing", that is, the opportunity to use a project already available on the Community, edit it, modify a part of the code, combining source materials, giving credit to the creator of the original project (Monroy-Hernández 2012). In some cases, participants had useful help in adopting the "remixing" approach to implement their app, downloading and modifying scripts from the sample projects.

The opportunity to find other projects and verify how some scripts can be arranged requires a great ability. In fact, pre-service teachers unconsciously used a top-down approach to coding, decomposing the whole app in small scripts and analysing the functionality of the different scripts.

Our findings show that Scratch projects were implemented by applying the concept of "designed for usability". Students settled the way in which the app had to run correctly, they personalized the stages and sprites according to the educational goal and user requirements. Learners customized sound, images, audio file, and so on, only 29% of the projects used backgrounds images from Scratch folder, the other teams created specific digital materials. Students settled the appropriate way in which the app had to run, they personalized the stages and sprites according to the educational goal and user requirements. Moreover, each workgroup tested their own app with the targeted final users. This approach, allowed teams to develop a *user-centred app* with a high level of usability, as results from the assessment. We explained to the pre-service teachers the importance to involve final users in the developing process of the app and to take into account users' needs; if their app has been effective, efficient, engaging, error tolerant, easy to learn.

The implemented apps allowed to easily handling modularity, due to the ease of use of Scratch. According to Wing (2006), modular application allows programmers to decompose the whole program in smaller pieces of code (scripts); hence, novices can easily manage interdependencies between the parts of the project, and assemble very complex applications in a reliable way. In fact, pre-service teachers created, shared, and reused objects and components; they broke the main problem into smaller components or modules and developed the scripts for each sprite independently and asynchronously before assembling them, in line with Brennan and Resnick findings (Brennan and Resnick 2012).

As regards the second research questions "*What is the level of acquisition of CT skills*?"

From a CT point of view, the app' score reflects the cognitive competence level reached by the programmers; hence, our findings show that our pre-service teachers sample gained a medium-high level of competences. In fact, 60% of the projects was categorized as developing, and 10% of the apps got the master level. Only 30% of the apps were basic. In particular, this latter 30% of the projects with a Basic competence level used only sequence blocks. They used the command "wait" for synchronization, and they used only the green flag to implement the interactivity.

As regards the medium-high level, 60% of the projects labeled as "Developing", had a better definition of blocks, and used conditionals like "if-else", and loops like "repeat" and "forever". The interactivity was applied by using pressed keys or clicked sprites. Finally, only few of them used variables. Hence, these teams were able to manage a more complex script, by using, for example, more complex interactivity functions than green flag click. 10% of the projects that obtained a Master competence level, used all the programming concepts explained during the lessons, except lists or arrays. The apps had audio records, logic operations, and the same code for different sprites.

Summing up, the used criteria to evaluate the student's work was easy to use and helpful to assess the increasing knowledge. *These criteria can be used easily also by pre-service teachers to evaluate the children competences.*

The adopted assessment method allows pre-service teachers to become more qualified and objective in judging the quality of a Scratch project, reducing also the effort and the subjectivity in the evaluation of the student's work (Brennan, 2013; Rehmat and Bailey, 2014).

*How pre-service teachers perceived themselves as regards their programming knowledge level at the end of the laboratory? In addition, how do they judge this experience?*

Analysing learners' answers, we noticed that 82% of the students perceived themselves as "experts" in using Scratch and the other software used to assemble the digital materials for the app's implementation. The other 18% of the students perceived themselves as "beginners".

Pre-service teachers judged their experience with Scratch as excellent and good (91.67%), able to stimulate their creativity (83.33%), offered them a good possibility to interact with their colleagues improving their work competences (79.17%). During the project development, students experienced some emotions, as a desire to finish the project, satisfaction, joy, and someone a little confusion. Pre-service teachers were satisfied and very satisfied with the setting applied to develop their own Scratch projects (81.82%).

As results of the investigation (see Table 3), each work team of pre-service teachers esteemed this experience useful for their career development. In particular, 85% of them judged "Personal satisfaction" positively, in the same way, 92.5% of them considered "Personal development" as a good and excellent experience". 85% of them estimated "Pedagogical knowledge improved after CT skills acquisition" and "High capability to design educational stuff after CT skills acquisition" "Very Good" and 15% selected the "Good" option.

The second section of the survey was focused on "Pre-service teachers teaching CT skills". 92.5% of pre-service teachers considered "Very good" their "Expertise improvement in teaching elements of CT to children" and 97.5% of them confirmed that "Children's motivation increase knowledge acquisition through technology" was "Very good".

A Pearson's correlation (two-tailed test) was run to determine the relationship between "High capability to design educational stuff after CT skills acquisition" (Question 3 of the survey – Q3) and "Pedagogical knowledge improved after CT skills acquisition" (Question 4 of the survey – Q4).

The calculation was run with SPSS (IBM statistic SPSS 24) software. The Pearson correlation coefficient value of 1.00 (see Table 12) confirms that there was a very strong, positive correlation between "High capability to design educational stuff after

Table 12
Correlations between Q3 – "High capability to design educational stuff after CT skills acquisition" and Q4 – "Pedagogical knowledge improved after CT skills acquisition"

| | | Q3 | Q4 |
|---|---|---|---|
| Q3 | Pearson Correlation | 1 | 1.000** |
| | Sig. (2-tailed) | | .000 |
| | N | 40 | 40 |
| Q4 | Pearson Correlation | 1.000** | 1 |
| | Sig. (2-tailed) | .000 | |
| | N | 40 | 40 |

** Correlation is significant at the 0.01 (2-tailed).

CT skills acquisition" and "Pedagogical knowledge improved after CT skills acquisition" (r = 1.00; N = 40; p =< .001). The direction of the relationships is positive meaning that these variables tend to increase together.

In the opinion of some pre-service teacher: "After completing this project we were very pleased with the results. We realized that the use of new kind of teaching tools is very important because we have enriched our cultural background and we understood that the initial scepticism shown towards this type of activity was totally unfounded. We have to be more flexible and elastic, not to limit the imagination of our pupils and to provide them with more tools for their learning process". Others: "We were given the opportunity to interact casually with colleagues. This opportunity has stimulated and motivated to reflect on the disadvantages and advantages of teamwork. Each meeting produced either uniformity of thoughts or divergence of opinions; however, it was an innovative training experience". "Scratch offers the opportunity not only to use it but also to create our own app, so that adults and children are not only users but also digital manufacturers".

In conclusion, we can say that programming with Scratch was an interesting and fun experience. We believe that this application is suiTable and useful for learning the notions of programming and construction of short stories involving children with a wide range of age. In addition, as future teachers, Scratch serves us to engage even more students to technology and, of course, to use this software in our teaching practice. Scratch is available to everyone, so even adults can use it to create nice scenes, presentations and games". In any case, engaging people in coding does not mean to turn them into expert programmers, but it means to foster the acquisition of CT skills and abilities, to promote their digital literacy and information technology knowledge, introducing them in the technological and software world (Polly 2015; Bertacchini *et al.* 2014). This remark is very important since all pre-service teachers involved in this research had never programmed before this experience.

## 6. Conclusions and Further Perspectives

The effects of the programming course using Scratch had two consequences on pre-service teachers. On the one hand, they questioned their teaching tools, because they had to compare technically and from the point of view of programming (to realize the lessons in Scratch); on the other hand, they were confronted with their specific pedagogical and disciplinary knowledge. In a nutshell, the pre-service teachers have been able to reflect on their knowledge and ability to foster their knowledge base for teaching.

The benefits consist in the practical applicability of their applications. In fact, all pre-service teachers have been able to directly administer to the students the applications developed in Scratch during their school internship hours, (about 2 hours per week, for all the weeks of the academic year). In this context, preservice teachers can interact from an educational point of view with the children, in accordance with the curricular planning that the tenured teacher does.

The challenges of the course concern the design of teaching units with Scratch realized according to three specific types of activities:

1. The representation of the contents, using the narration and the representation of events and phenomena to simulate the main concepts and phenomena that we want to communicate and make children acquire. This part seemed very stimulating to the preservice as they found themselves having to narrate in a new way, using a whole series of characters, environments, objects, objects and tools that normally do not use in the construction of traditional teaching units.
2. The construction of interaction environments for children. This type of activity has become very stimulating for the pre-service, as to realize this interactive activity they had to build a series of games of manipulation of data with the Software Scratch, which in turn also corresponds to a conceptual manipulation by children.

A coding learning laboratory has been arranged at the University of Calabria (Italy) to define guidelines on how to implement a Programming Laboratory for pre-service teachers, using Scratch with the aim to foster CT acquisition and to achieve the 21st skills.

The complexity of the developed app reflects the learners coding skills. Hence, the efficacy of our didactic methodological guidelines were measured by individuating the learned programming concepts and the competence level of CT skills acquired by the pre-service teachers.

According to McComas (2013), an imperative need of our society is to prepare teachers to efficaciously delivery skills and contents, and this requires an adequate curriculum experience and a reasonable time of practice. In our opinion, this is also particularly true for pre-service teachers, to prepare them on how to teach 21st Century Skills.

Our results suggest that the planning laboratory activities had a successful effect on pre-service teachers' digital literacy. In fact, our findings showed a full engagement of novice programmers with coding. This was probably due to the possibility to verify in real time the output of the scripts on the screen. This facility is particularly useful for all types of didactic projects (Koh 2013).

Regarding the practical importance of this investigation, we conclude that in a few weeks, students were able to acquire the key elements of coding needed to develop Scratch projects according to the suggested educational perspective. Furthermore, this group of pre-service teachers was able to face the 21st century requirements, gaining programming concepts, problem-solving practices, and computational skills. Going on with the work, using new technological tools, learners were able to develop different abilities such as creativity, critical thinking, and decision-making, sharing at the same time knowledge and responsibilities in the group.

The presented didactic methodology for pre-service teachers has a twofold advantage:

1. It represents a useful and repeaTable example that pre-service teachers could apply as guidelines in their future classrooms and for their curriculum. In fact, we showed them how to include the coding activities into the didactic planning, giving them some didactic methodological guidelines.
2. It is "friendly". In a very short time, thanks to the collaborative and cooperative learning approach adopted, with a constructivist perspective and fostering a

learning-by-doing skills acquisition, novice programmers reached a medium-high competence level in coding.

The present paper sums up a long work carried out at different levels: bureaucratic level, didactic level and individual level to better motivate students to face challenges and the general change. The approach adopted in our university, starts from the general observation that there exists a gap between university education in Computer Science for Elementary teachers, and the school needs, as remarked also by the National Law. Hence, after long discussions on this problem, we had the opportunity to launch a new programme for the Laboratory on Educational Technology for pre-service teachers: a new experimental programme course based on Scratch. A strong work has been carried out to arrange didactical materials and to choose the right educational approach. However, good results were achieved at the end of the course, as highlighted in the results here presented, and during the European Researchers' Night 2016 at University of Calabria, when pre-service teachers showed their developed app to the visitors.

Why our small research is important? Porter, Lee, Simon, and Guzdial (2017) underline the need and the importance "to emphasize best practices underrepresented groups".

*Our approach starts from the low because it takes into account the real learners' requirements that have to approach to the programming. These latter have not a basic computer science background and in some case, they are adult learners that need to be greatly motivated.*

In our opinion, the results presented in this paper, focused on measuring the efficacy of our didactic methodological guidelines, allows to collect important information from an academic point of view. In particular, both positive and negative laboratory outcomes might lead to: re-arrange some key University courses; rethink Italian pre-service teachers' curricula, in order to be really centred on the needs of the future in-service teachers.

Educational paths focused on 21st Century Skills acquisition means to promote the attainment of life skills, workforce skills, applied skills, personal and interpersonal skills as well as non-cognitive skills (Trilling and Fadel 2009). The acquisition of these abilities represents a great challenge for each educational system since these skills are the scaffolding that allows everyone to face the changes that occurred in the learning processes, in the educational and social life aspects, in the last years.

In our opinion, it would be interesting to investigate, in a future work, also pre-service teachers' motivation: for example, to verify how motivation changes the educational paradigm, administrating a pre and post-test.

## References

Adamo, A., Bertacchini, P.A., Bilotta, E., Pantano, P., Tavernise, A. (2010). Connecting art and science for education: learning through an advanced virtual theater with "talking heads". *Leonardo*, 43(5), 442–448.

Alì, G., Bilotta, E., Gabriele, L., Pantano, P. (2006). An e-learning platform for academy and industry networks. In: *PerCom Workshops 2006. Fourth Annual IEEE International Conference on* (Pervasive Computing and Communications Workshops, 2006). IEEE, 4-pp.

An, S., Lee, Y. (2014). Development of pre-service teacher education program for computational thinking. In: *Proceedings of Society for Information Technology & Teacher Education International Conference.*

2048–2052.

Angeli, C. (2004). The effects of case-based learning on early childhood pre-service teachers' beliefs about the pedagogical uses of ICT. *Journal of Educational Media*, 29(2), 139–151.

*Assessment & Teaching of 21st Century Skills*. Retrieved September 21, 2016 from `http://www.atc21s.org/`

Armoni, M., Meerbaum-Salant, O., Ben-Ari, M. (2015). From Scratch to "Real" Programming. *Trans. Comput. Educ.* 14(4), Article 25 (February 2015), 15 pages. DOI: 10.1145/2677087

Barr, D., Harrison, J., Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, 38(6), 20–23.

Baydas, O., Goktas, Y. (2016). Influential factors on preservice teachers' intentions to use ICT in future lessons. *Computers in Human Behavior*, 56, 170–178.

Baytak, A. Land, S.M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development*. 59(6), 765–782. DOI: 10.1007/s11423-010-9184-z

Bell, S., Frey, T., Vasserman, E. (2014). Spreading the word: introducing pre-service teachers to programming in the K12 classroom. In: *Proceedings of the 45th ACM technical symposium on Computer science education* (SIGCSE'14). ACM, New York, NY, USA, 187–192. DOI: 10.1145/2538862.2538963

Bean, N., Weese, J., Feldhausen, R., Bell, R.S. (2015). Starting from scratch: Developing a pre-service teacher training program in computational thinking, In: *IEEE Frontiers in Education Conference (FIE)*, El Paso, TX, 1–8, DOI: 10.1109/FIE.2015.7344237

Bennett, L. (2014). Learning from the early adopters: developing the Digital Practitioner. *Research In Learning Technology, 22*. DOI: 10.3402/rlt.v22.21453

Bertacchini, F., Gabriele, L., Pantano, P.S., Olmedo-Vizueta, D., Giaquinta, A., Tavernise, A., Bilotta, E. (2018). A facial emotions recognition application for subjects with Autism Spectrum Disorder, *EDULEARN18 Proceedings*, 4005–4011.

Bertacchini, F., Giglio, S., Gabriele, L., Pantano, P.S., Bilotta, E. (2018). New technologies for improving tourism students training, *EDULEARN18 Proceedings*, 4155–4162.

Bertacchini, F., Bilotta, E., Pantano, P. (2017). Shopping with a robotic companion. *Computers in Human Behavior*, 77, 382–395.

Bertacchini, F., Bilotta, E., Caldarola, F., Pantano, P., Bustamante, L.R. (2016, October). Emergence of linguistic-like structures in one-dimensional cellular automata. In: *AIP Conference Proceedings*. AIP Publishing, 1776(1), 090044.

Bertacchini, F., Bilotta, E., Gabriele, L., Vizueta, D.O., Pantano, P., Rosa, F., Tavernise, A., Valenti, A. (2013). An emotional learning environment for subjects with Autism Spectrum Disorder. *International Conference on Interactive Collaborative Learning (ICL)* (2013), 653–659. DOI: 10.1109/icl.2013.6644675

Bertacchini, F., Bilotta, E., Carini, M., Gabriele, L., Pantano, P., Tavernise, A. (2014). Learning in the Smart City: A Virtual and Augmented Museum Devoted to Chaos Theory. *Lecture Notes in Computer Science New Horizons in Web Based Learning* (2014), 261–270. DOI: 10.1007/978-3-662-43454-3_27

Bertacchini, F., Tavernise, A. (2014). Knowledge sharing for Cultural Heritage 2.0: prosumers in a "digital agora". *International Journal of Virtual Communities and Social Networking (IJVCSN)*, (6)2, 24–36.

Bertacchini, F., Bilotta, E., Gabriele, L., Pantano, P., Tavernise, A. (2015). Designing an educational music software using a student-centred strategy. In: Roberta V. *Nata Progress in Education*, New York: Nova Science Publishers.

Bertacchini, F., Bilotta, E., Gabriele, L., Pantano, P., Tavernise, A. (2013). Toward the use of Chua's circuit in education, art and interdisciplinary research: Some implementation and opportunities. *Leonardo*, 46(5), 456–463.

Bertacchini, F., Bilotta, E., Pantano, P., Tavernise, A. (2012). Motivating the learning of science topics in secondary school: A constructivist edutainment setting for studying Chaos. *Computers & Education,* 59(4), 1377–1386.

Bertacchini, F., Gabriele, L., Tavernise, A. (2011). Bridging educational technologies and school environment: implementations and findings from research studies. *Educational theory*, 63–82.

Bertacchini, F., Bilotta, E., Gabriele, L., Pantano, P., Servidio, R. (2010). Using Lego MindStorms in higher education: Cognitive strategies in programming a quadruped robot. In: *Workshop proceedings of the 18th international conference on computers in education, ICCE*. 366–371.

Bertacchini, F., Bilotta, E., Gabriele, L., Mazzeo, V., Pantano, P., Rizzuti, C., Vena, S. (2007). Imagination-TOOLS™: Made to play music. In: *International Conference on Technologies for E-Learning and Digital Entertainment*. Springer, Berlin, Heidelberg, 369–380.

Bertacchini, F., Bilotta, E., Gabriele, L., Pantano, P., Tavernise, A. (2015). Designing an educational music software using a student-centred strategy. *Progress in Education*, 33, 89–99.

Bilotta, E., Bossio, E., Pantano, P. (2010). Chaos at school: Chua's circuit for students in junior and senior high school. *International Journal of Bifurcation and Chaos*, 20(01), 1–28.

Bilotta, E., Di Blasi, G., Stranges, F., Pantano, P. (2007). A gallery of Chua attractors: part IV. *International Journal of Bifurcation and Chaos*, 17(04), 1017–1077.

Bilotta, E., Di Blasi, G., Stranges, F., Pantano, P. (2007). A gallery of Chua attractors: part VI. *International Journal of Bifurcation and Chaos*, 17(06), 1801–1910.

Bilotta, E., Fiorito, M., Iovane, D., Pantano, P. (1995). An educational environment using WWW. *Computer Networks and ISDN Systems*, 27(6), 905–909.

Bilotta, E., Gabriele, L., Servidio, R., Tavernise, A. (2009). Edutainment robotics as learning tool. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5940 LNCS, 25–35. DOI: 10.1007/978-3-642-11245-4_3.

Bilotta, E., Lafusa, A., Pantano, P. (2003). Life-like self-reproducers. Complexity, 9(1), 38–55.

Bilotta, E., Pantano, P. (2006). Structural and functional growth in self-reproducing cellular automata. *Complexity*, 11(6), 12–29.

Bilotta, E., Pantano, P., Bertacchini, F., Gabriele, L., Longo, G., Mazzeo, V., Vena, S. (2007). Imagination-TOOLS (TM)-A 3D environment for learning and playing music. In: *Eurographics Italian Chapter Conference*. 139–144.

Bilotta, E., Pantano, P., Cupellini, E., Rizzuti, C. (2007, April). Evolutionary methods for melodic sequences generation from non-linear dynamic systems. In: *Workshops on Applications of Evolutionary Computation*. Springer, Berlin, Heidelberg, 585–592.

Bilotta, E., Pantano, P., Stranges, F. (2006). Computer graphics meets chaos and hyperchaos. Some key problems. *Computers & Graphics*, 30(3), 359–367.

Bilotta, E., Pantano, P., Vena, S. (2011). Artificial micro-worlds: part I: A new approach for studying life-like phenomena. *International Journal of Bifurcation and Chaos*, 21(02), 373–398.

Bilotta, E., Stranges, F., Pantano, P. (2007). A gallery of Chua attractors: part III. *International Journal of Bifurcation and Chaos,* 17(03), 657–734.

Blömeke, S., Busse, A., Kaiser, G., König, J., Suhl, U. (2016). The relation between content-specific and general teacher knowledge and skills. *Teaching and Teacher Education*, 56, 35–46.

Brennan, K.A., Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In: *Proceedings of the 2012 annual meeting of the American Educational Research Association*, (2012). Vancouver, Canada, 1–25.

Brennan K.A. (2013). Best of Both Worlds: Issues of Structure and Agency in Computational Creation, in and out of School. PhD Thesis. Massachusetts Institute of Technology.

Buechley, L., Eisenberg, M. (2008). The LilyPad arduino: Toward wearable engineering for everyone. *IEEE Pervasive Computing*, 7(2), 12–15. DOI: 10.1109/mprv.2008.38

Bullock S.M., Sator, A.J. (2015). Maker pedagogy and science teacher education. *Journal of the Canadian Association for Curriculum Studies*, 13(1), 60–87.

Chai, C.S., Ling Koh, J.,H., Chin-Chung Tsai, Lynde, L., Wee Tan (2011). Modeling primary school pre-service teachers' Technological Pedagogical Content Knowledge (TPACK) for meaningful learning with information and communication technology (ICT). *Computers & Education*, 57(1), 1184–1193. DOI: 10.1016/j.compedu.2011.01.007

Chiu, C. (2014). Use of problem-solving approach to teach scratch programming for adult novice programmers (abstract only). In: *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (SIGCSE'14). ACM, New York, NY, USA, 710–711. DOI: 10.1145/2538862.2544284

Choi, H. (2013). Pre-service teachers' conceptions and reflections of computer programming using Scratch: Technological and pedagogical perspectives. *International Journal for Educational Media and Technology*, 7(1), 15–25.

Coutinho, C.P. (2009). Challenges for teacher education in the learning society: case studies of promising practice. *Handbook of Research and Practices in e-Learning: Issues and Trends*, 385–401. DOI: 10.4018/978-1-60566-788-1.ch023

Curzon, P., Dorling, M., Ng, T., Selby, C., Woollard, J. (2014). *Developing Computational Thinking in the Classroom: A Framework*. Swindon, GB, Computing at School.

Dasgupta, S., Resnick, M. (2014). Engaging novices in programming, experimenting, and learning with data. *ACM Inroads*, 5(4), 72–75.

Denner, J., Werner, L. (2011). Measuring computational thinking in middle school using game programming. In: *Proceedings of the Annual Meeting of the American Educational Research Association*.

Denner, J., Bean, S., Martinez, J. (2009). The girl game company: Engaging Latina girls in information technology. *Afterschool Matters*, 8, 26–35.

Denner, J., Werner, L., Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education*, 58(1), 240–249.

Denner, J., Werner, L. (2011). Measuring computational thinking in middle school using game programming. In: *Proceedings of the Annual Meeting of the American Educational Research Association*.

Du Toit, J. Teacher Training and Usage of ICT in Education. *New Directions for the UIS Global Data Collection in the Post-2015 Context*. UNESCO Institute for Statistics. Retrieved March 2017, from `http://www.uis.unesco.org/StatisticalCapacityBuilding/Workshop%20Documents/Communication%20workshop%20dox/Paris%202014/ICT-teacher%20training-use_EN.pdf`

Ericson, B., McKlin, T. (2012). Effective and sustainable computing summer camps. In: *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (SIGCSE '12). ACM, New York, NY, USA, 289–294. DOI: 10.1145/2157136.2157223

Ertmer, P., Ottenbreit-Leftwich, A. (2010). Teacher technology change: how knowledge, confidence, beliefs, and culture intersect. *Journal of Research on Technology in Education*, 42(3), 255–284.

Faraco, G., Gabriele, L. (2007). Using LabVIEW for applying mathematical models in representing phenomena. *Computers & Education*, 49(3), 856–872.

Federici, S., Gola, E., Brau, D., Zuncheddu, A. (2015). Are educators ready for coding?. In: Markus Helfert, Maria Teresa Restivo, Susan Zvacek, and James Uhomoibhi (Eds.) *Proceedings of the 7th International Conference on Computer Supported Education* – Volume 1 (CSEDU 2015). SCITEPRESS – Science and Technology Publications, Lda, Portugal, 494–500. DOI: 10.5220/0005491604940500

Fesakis, G., Serafeim, K. (2009). Influence of the familiarization with "scratch" on future teachers' opinions and attitudes about programming and ICT in education. In: *Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE '09). ACM, New York, NY, USA, 258–262. DOI: 10.1145/1562877.1562957

Foerster, K. (2016). Integrating programming into the mathematics curriculum: Combining Scratch and geometry in Grades 6 and 7. In: *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16).* ACM, New York, NY, USA, 91–96. DOI: 10.1145/2978192.2978222

Gabriele, L., Tavernise, A., Bertacchini, F. (2012). Active learning in a robotics laboratory with university students. In: Wankel, C., Blessinger, P. (Eds.) *Increasing Student Engagement and Retention Using Immersive Interfaces: Virtual Worlds, Gaming, and Simulation (Cutting-edge Technologies in Higher Education, Volume 6 Part C).* Emerald Group Publishing Limited, 315–339. DOI: 10.1108/s2044-9968-(2012)000006c014

Gabriele, L., Bilotta, E. (2013). Robot for thinking: cognitive strategies to experiment the use of educational robotics in classroom. *EDULEARN13 Proceedings*, 5801–5810.

Gabriele, L., Marocco, D., Bertacchini, F., Pantano, P., Bilotta, E. (2017). An educational robotics lab to investigate cognitive strategies and to foster learning in an arts and humanities course degree. *International Journal of Online Engineering*, 13(4), 7–19. DOI: 10.3991/ijoe.v13i04.6962

Gill, L., Dalgarno, B. (2008). Influences on pre-service teachers' preparedness to use ICTs in the classroom. *Hello! Where are you in the landscape of educational technology? Proceedings Ascilite Melbourne 2008*.

Griffin, P. (2014). Performance assessment of higher order thinking. *Journal of Applied Measurement,* 15(1), 1–16.

Gunter, G.A. (2001). Making a difference: Using emerging technologies and teaching strategies to restructure an undergraduate technology course for pre-service teachers. *Educational Media International*, 38(1), 13–20.

Guzzetti, B.J., Elliott, K.F., Welsch, D. (2010). *DIY Media in the Classroom: New Literacies across Content Areas*. Teachers College Press.

Heong, Y.M., Yunos, J.M., Othman, W., Hassan, R., Kiong, T.T., Mohamad, M.M. (2012). The needs analysis of learning higher order thinking skills for generating ideas. *Procedia-Social and Behavioral Sciences*, 59, 197–203.

Indicazioni nazionali per il curricolo della scuola dell'infanzia e del primo ciclo d'istruzione. 2012. Retrivied September 11, 2016 from `http://www.indicazioninazionali.it/documenti_Indicazioni_nazionali/indicazioni_nazionali_infanzia_primo_ciclo.pdf`

Johnson, D.E. (2016). ITCH: Individual Testing of Computer Homework for Scratch Assignments. In: *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (SIGCSE '16). ACM, New York, NY, USA, 223–227. DOI: https://doi.org/10.1145/2839509.2844600

Kafai, Y.B., Peppler, K.A. (2011). Youth, technology, and DIY developing participatory competencies in creative media production. *Review of Research in Education*, 35(1), 89–119.

Kobsiripat, W. (2015). Effects of the Media to Promote the Scratch Programming Capabilities Creativity of Elementary School Students. *Procedia-Social and Behavioral Sciences*, 174, 227–232.

Koh, K. (2013). Adolescents' information-creating behavior embedded in digital Media practice using scratch. *Journal of the American Society for Information Science and Technology*, 64(9), 1826–1841.

Koh, K.H. (2014). Computational thinking pattern analysis: a phenomenological approach to compute computational thinking. Diss. *Computer Science Graduate Theses & Dissertations*. 86. `http://scholar.colorado.edu/csci_gradetds/86`

Korkmaz, Ö. (2016). The Effects of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *Education*, *3*(4), 13–15.

Lei, J. (2010). Quantity versus quality: A new approach to examine the relationship between technology use and student outcomes. *British Journal of Educational Technology*, 41(3), 455–472. DOI: 10.1111/j.1467-8535.2009.00961.x

Lye, S.Y., Koh, J.H.L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?, *Computers in Human Behavior*, 41, 51–61.

Lynde, T., Kim, B. (2015). Learning by doing in the digital media age. In: *New Media and Learning in the 21st Century*. Springer, Singapore, 2015, 181–197. DOI: 10.1007/978-981-287-326-2_12

Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M., Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, 40(1), 367–371.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 16.

Masterman, E., Manton, M. (2011). Teachers' perspectives on digital tools for pedagogic planning and design. *Technology, Pedagogy and Education*, 20(2), 227–246.

McComas, W.F. (2013). The Language of Science Education: an Expanded Glossary of Key Terms and Concepts in Science Teaching and Learning, Rotterdam: Sense Publishers.

McInerney, D.M., Gavin, B.TL., Liem, G.A.D. (2009). Student perspectives on assessment: What students can tell us about assessment for learning. Vol. 9. IAP, Charlotte, NC.

Meerbaum-Salant, O., Armoni, M., (Moti) Ben-Ari, M. (2013). Learning computer science concepts with Scratch, *Computer Science Education*, 23(3), 239–264, DOI: 10.1080/08993408.2013.832022

Monroy-Hernández, A. (2012). *Designing for remixing: Supporting an online community of amateur creators.* PhD Thesis. Massachusetts Institute of Technology.

Moreno-León, J., Robles, G., Román-González, M. (2015). Dr. Scratch: análisis automático de proyectos Scratch para evaluar y fomentar el pensamiento computacional. *Revista de Educación a Distancia*, 46(10), 15-Sept.-2015, DOI: 10.6018/red/46/10, `http://www.um.es/ead/red/46/moreno_robles_es.pdf`

Papert, S. (1980). *Mindstorms: Children, Computers, And Powerful Ideas*. New York: Basic Books

Polenghi, S., Triani, P. (2014). Teacher training and profession in Italy. Today's situation after a 250 years history. In: Gabriella Pusztai, Įgnes Engler. *Teacher Education Case Studies in Comparative Perspective.* Center for Higher Education Research and Development-Hungary.

Polly, D. (2015). Elementary Education Pre-service Teachers' Development of Mathematics Technology Integration Skills in a Technology Integration Course. *Journal of Computers in Mathematics and Science Teaching*, 34(4), 431–453.

Porter, L., Lee, C., Simon, B., Guzdial, M. (2017). Education. Preparing Tomorrow's Faculty to Address Challenges in Teaching Computer Science. Using a "boot camp" workshop for new faculty orientation. *Communications of the ACM*, 60(5), 25–27. DOI: 10.1145/3068791

Polly, D. (2015). Elementary Education Pre-service Teachers' Development of Mathematics Technology Integration Skills in a Technology Integration Course. *Journal of Computers in Mathematics and Science Teaching*, 34(4), 431–453.

*Programma il Futuro – Code.org* (2016). Retrieved September 21, from `http://www.programmailfuturo.it/`

Rehmat A.P., Bailey J.M. (2014). Technology Integration in a Science Classroom: Preservice Teachers' Perceptions. *Journal of Science Education and Technology* 23(6), 744–755. DOI: 10.1007/s10956-014-9507-7

Repenning, A., Webb, D.C., Koh, K.H., Nickerson, H., Miller, S.B., Brand, C., Many Horses, I.H., Basawapatna, A., Gluck, F., Grover, R., Gutierrez, K., Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *Trans. Comput. Educ.,* 15(2), Article 11 (31 pages). DOI: 10.1145/2700517

Repenning, A., Webb, D.C., Koh, K.H., Nickerson, H., Miller, S.B., Brand, C., ... Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE)*, 15(2), 11.

Repenning, A. (2011). Making programming more conversational. In: *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, VL/HCC '11, (Pittsburgh, PA, USA Sept. 18–22, 2011). IEEE Computer Society, Los Alamitos, CA.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60–67. DOI: 10.1145/1592761.1592779

*Scratch – Imagine, Program, Share* (2016). Retrieved September 21, 2016 from `https://scratch.mit.edu/`

Seiter, L., Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. In: *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research* . ACM, New York, NY, USA, 59–66. DOI: 10.1145/2493394.2493403

Tavernise, A., Bertacchini, F., Pantano, P.S., Bilotta, E. (2016). Implementing a new Class-Lab: guidelines for integrating innovative devices in pre-service teachers' practice. *International Journal of Digital Literacy and Digital Competence (IJDLDC)*. Special Issue on "School revolution? Let's start from teachers' digital literacy and competences!". Hershey, USA: IGI Global Publishing. DOI: 10.4018/IJDLDC.

Tømte, C., Enochsson, A.B., Buskqvist, U., Kårstein, A. (2015). Educating online student teachers to master professional digital competence: The TPACK-framework goes online. *Computers & Education*, 84, 26–35.

Tondeur, J., Aesaert, K., Pynoo, B., Braak, J., Fraeyman, N., Erstad, O. (2016). Developing a validated instrument to measure preservice teachers' ICT competencies: Meeting the demands of the 21st century. *British Journal of Educational Technology*. DOI: 10.1111/bjet.12380

Tondeur, J., Van Braak, J., Sang, G., Voogt, J., Fisser, P., Ottenbreit-Leftwich, A. (2012). Preparing pre-service teachers to integrate technology in education: A synthesis of qualitative evidence. *Computers & Education*, 59(1), 134–144.

Trilling, B,. Fadel, C. (2009). *21st century skills: learning for life in our times*, San Francisco: Jossey-Bass.

Vaca Cárdenas, L.A., Bertacchini, F., Gabriele, L., Tavernise, A., Olmedo, D., Pantano, P., Bilotta, E. (2015). Surfing virtual environment in the Galápagos Islands. In: *Remote Engineering and Virtual Instrumentation (REV), 2015 12th International Conference on*, 25–27 Feb. 2015. IEEE, 192–198. DOI: 10.1109/REV.2015.7087291

Vaca Cárdenas, L.A., Olmedo, D., Tavernise, A., Gabriele, L., Bertacchini, F., Pantano, P., Bilotta, E. (2014). Darwin has come back to the Galápagos Islands: An educational journey to discover biological evolution. *EDULEARN14 Proceedings*, 6088–6095.

Vaca-Cárdenas, L.A., Bertacchini, F., Tavernise, A., Gabriele, L., Valenti, A., Olmedo, D., Pantano, P., Bilotta, E. (2015). Coding with Scratch: The design of an educational setting for Elementary pre-service teachers. In: *Interactive Collaborative Learning (ICL), 2015 International Conference on*, 20–24 Sept. 2015. IEEE, 1171–1177. DOI: 10.1109/ICL.2015.7318200

Vaca-Cárdenas, L., Tavernise, A., Bertacchini, F., Gabriele, L., Valenti, A., Pantano, P., Bilotta, E. (2016). An educational coding laboratory for elementary pre-service teachers: A qualitative approach. *International Journal of Engineering Pedagogy (iJEP)*, 6(1), 11–17. `http://dx.doi.org/10.3991/ijep.v6i1.5146`

Wilson, A., Moffat, D. C. (2010). Evaluating Scratch to introduce younger school children to programming. In: *Proc. of the Psychology of Programming Interest Group Workshop.* Madrid/Espanha, 64–75.

Wing, J.M. (2006). Computational thinking. *Commun. ACM* 49, 3 (March 2006), 33–35. DOI: 10.1145/1118178.1118215

Wolz, U., Stone, M., Pearson, K., Pulimood S.M., Switzer, M. (2011). Computational thinking and expository writing in the middle school. *Trans. Comput. Educ.* 11(2), article 9 (22 pages). DOI: 10.1145/1993069.1993073.

**L. Gabriele** is with the Physics Department, University of Calabria, Rende, Italy. She is a post-doc Fellow and lecturer and is interested in the following topics: psychology of programming, educational technologies and their use in didactic context, learning, and techniques of Human Computer Interaction, communications systems and their use for educational purposes. She is an occasional referee of the journals Computers & Education, Scientific Research and Essays, and member of the Editorial Board of the Journal of Applied Research in Higher Education (JARHE). She worked in different national and international projects.

**F. Bertacchini** is with the Mechanical, Energy and Management Engineering Department, University of Calabria, Rende, Italy. She is a Ph.D in "Psychology of Programming and Artificial Intelligence", and a Post-Doc Fellow. She is a member of the research group ESG, and works in the Laboratory of Psychology and Cognitive Science at University of Calabria, designing educational-entertaining environments for science communication and education. Among her research interests we can cite the followings: Educational Technologies, Art & Science, Scientific Visualization, Psychology of Music.

**A. Tavernise** is member of both the Laboratory of Psychology and Cognitive Science and the research group called Evolutionary Systems Group at the University of Calabria. The focus of her research is the design and implementation of educational settings/paths in technology-enhanced contexts in order to investigate the acquisition of high-level cognitive skills (as critical thinking, problem solving, creativity, collaboration) and, in particular, learning. Many of her research results were achieved in national and international projects, as for example Connecting European Culture through New Technology – NetConnect (EU project, financed in the programme Culture2000) and Virtual Museum System (national projects POR 2000–2006).

**L. Vaca-Cárdenas** is Ph.D graduated and collaborated with the Laboratory of Psychology and Cognitive Science. She is a Systems Engineer and she has a Master degree in University Teaching and Educational Research. Her research interests are focused in the following topics: E-learning Systems, Virtual Learning platforms, Information and Communication Technologies (ICTs), Internet of Things (IoT), Virtual Reality, Games and programming tools for education.

**P. Pantano** is with the Physics Department, University of Calabria, Rende, Italy. He is a full Professor of Physics and Mathematics at University of Calabria. His main research interests concern several topics according to an interdisciplinary approach, among which we can cite the followings: e-Learning systems; scientific communication; Artificial life. He is a member of several national and international scientific communities, scientific committees and organized several congresses and workshops. From 2011 to 2016 was the director of a Doctoral School at University of Calabria. He participated in national and international projects as responsible of local research unit as well as international coordinator.

**E. Bilotta** is with the Physics Department, University of Calabria, Rende, Italy. She is a full professor of General Psychology at University of Calabria, Italy, where she is also the director of the Cognitive Psychology Laboratory. Her research interests concern various scientific topics from an interdisciplinary point of view and comprise the following areas: Human Computer Interaction, Psychology of Programming, Information and communication technologies, Educational Technologies. She published more than 200 scientific papers and is the author of various books. She was the coordinator of (or participated in) national and international projects. Moreover, she is a member of several national and international scientific communities.

## APPENDIX: Two Apps in Depth

In this Appendix, to better illustrate the findings, we describe in details two apps and their assessment, to show how in a very short time, pre-service teachers acquired both basic and advanced computer science concepts thanks also the methodology adopted during the Lab.

The first app is "***Little Red Riding Hood in an apple garden***" and was developed by a group of three pre-service teachers.

The educational objective was: improving the ability to count from 1 to 10, in pre-school children (aged from 4 to 5). In this app, the main character (that is, the sprite) is a young girl known as Little Red Riding Hood (the fairy tale character). There are different scenes (a project with different backdrops, like a story with multiple scenes). In the first scene, the title of the application and the image of the garden apples are present. In the second scene, Little Red Riding Hood talks to her grandmother (called Granny), who asks her to go to the garden and collect 10 apples for making a cake. In the third scene, the user can move Little Red Riding Hood using the direction keys, while a recorded voice explains the concept of tens.

In Fig. 3 a screenshot of the app "Little Red Riding Hood in the apple garden" (scene 3) is showed.

In the same scene, Little Red Riding Hood and Granny ask the user to repeat a numerical sequence. In the last scene, Granny explains to Little Red Riding Hood the quickest way to reach her house, and the user has to follow her instructions, following a specific path and counting the trees. If the user takes the wrong way, all the operations are automatically reset, and he/she has to start again the same scenario; if the user follows the instructions correctly, he/she will reach the target (Granny's home and the apple pie).

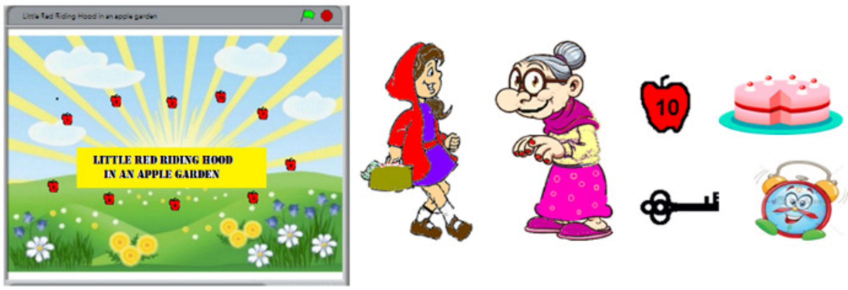

Little Red Riding Hood in an apple garden
LITTLE RED RIDING HOOD IN AN APPLE GARDEN
10

Fig. 3. A screenshot of the app "Little Red Riding Hood in the apple garden". The educational objective of the app is the improvement the ability to count from 1 to 10 in preschool children.

*According Denner's (2012) evaluation criteria, the sprites and* scenes, *in the game are controlled by the arrow keys and by the keys 1, 2, 3 and 4. The app uses conditions such as: if the red apples are touched by Little Red Riding Hood a sound is played; if she collects an apple, the apple disappears.*

*"Little Red Riding Hood in an apple garden" has not variables or random numbers, but it presents statements of sequence, user interaction, Boolean expressions, conditions, loops and threads control commands and also voice records and sounds.*

*According to Dr. Scratch software, this app has a "**developing**" level. All the CT concepts are used, the feedback of the program reports that all 33 sprites have the default name, that should be changed for a better organization. The concepts on data representation, abstraction and synchronization should be improved.*

The second app is "***Uncertainty Mathematics: From Luckyville to Sharpsville***" and was developed by a group of three pre-service teachers.

The educational objective was the improving of the concept of "probability calculation" in elementary school children (aged from 9 to 11); users' prerequisite was the basic knowledge of Computer Science.

Eight apps with different difficulty levels are developed and linked through a community. The apps are also customized for children with learning difficulties: dyslexia (audio messages allow to overcome the obstacle of reading the instructions); dyscalculia (only few symbols and mathematical formulas are used). From the main, user can choose among eight game levels: 1) Certainty, feasibility, impossibility; 2) Objects matrix; 3) Probability; 4) Combinatory; 5) Lucky wheel; 6) Spider web; 7) Butterflies; 8) Monty Hall.

1) **Certainty, feasibility, impossibility.** The aim of this app is to teach the difference between events that can be predicted and events that cannot be. In order to understand if an event is predicTable or not, children have to make decisions based on the experience and the circumstances in which the event occurs (the variables).
2) **Objects matrix.** Children have to identify object properties and operate their combinations on a matrix. Users learn that the knowledge of variables allows the prediction of the even, as well as the use of a double entry Table to compare the probability of events and to highlight correlations.
3) **Probability.** The aim of the app is the comparison with the events in order to reflecting on their degrees of probability. Children have to distinguish between probable and improbable events. Children have to compare the probability in the space of the events, understanding that a different distribution of the variables creates different probabilities.
4) **Combinatory.** The aim of the app is the measurement of the probability taking into account the ratio between favourable and possible cases. Children have to work on the space of the event, aggregate and eliminate cases.

5) **Lucky wheel.** Children have to make decisions using the calculation of probabilities.
6) **Butterfly.** Children have to understand the probability of an event using the mathematical fraction, switching from visual and semantic representations to symbolic ones (mathematical representations).
7) **Spider Web.** Three concrete events are presented; children have to understand which is the most probable. Hence, they learn to make decisions using the calculation of probabilities, and controlling multiple variables simultaneously.
8) **Monty Hall.** Children have to recognize and define the space of events in complex situations. They have to solve problems, identify appropriate strategies, justify the followed procedure, and use the learned concepts. In particular, behind three locked doors there are a car and two goats. The user wins when he finds the car. First, the user must choose a door, that remains closed, while another door opens. The user can decide to change or keep the object behind the closed door. 1 point is assigned for each right choice (the car); the aim of the game is to score 3 points. The user has to understand if the change of the chosen door increases the probability of winning. In Fig. 4, a screenshot of the app "Monty Hall" is showed.

*Applying Denner's (2012) evaluation criteria, "Uncertainty Mathematics: From Luckyville to Sharpsville" the following concepts were learned by pre-service teachers: statements of sequence, user interaction, Boolean expressions, conditions, loops and threads control commands and also voice records, sounds and very original handmade images. According with Dr. Scratch "Monty Hall" app has a* ***master level****, because uses Flow control, Data Representation, Abstraction, user interactivity, synchronization, parallelism and logic, sprite attributes.* From an educational point of view, this collection of apps had a great success with children, arousing enthusiasm and stimulating curiosity in the targeted users.

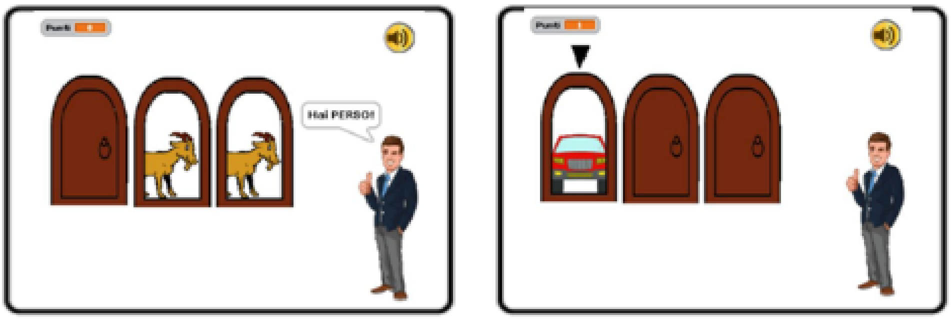


Fig. 4. A screenshot of the app "Monty Hall". Users have to recognize and define the space of events in complex situations.