Going beyond digital literacy to develop computational thinking in K-12 education,

Divya Menon, Sowmya Bp, Margarida Romero, Thierry Viéville

▶ **To cite this version:**

Divya Menon, Sowmya Bp, Margarida Romero, Thierry Viéville. Going beyond digital literacy to develop computational thinking in K-12 education,. Linda Daniela. Smart Pedagogy of Digital Learning, Taylor&Francis (Routledge), 2019, 9780367333799. hal-02281037

HAL Id: hal-02281037

https://inria.hal.science/hal-02281037v1

Submitted on 18 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Going beyond digital literacy to develop computational thinking in K-12 education

Divya MENON[1,2,3], Sowmya BP[1,3], Margarida ROMERO[1], Thierry VIEVILLE[2,1].
[1] Laboratoire d'Innovation et Numérique pour l'Education http://unice.fr/laboratoires/line
margarida.romero@unice.fr
[2] INRIA, Mission de Médiation Scientifique thierry.vieville@inria.fr
[3] MSc #CreaSmartEdtech http://app.univ-cotedazur.fr/smartedtech
divya.menon@etu.univ-cotedazur.fr sowmya.bp@etu.univ-cotedazur.fr

Abstract

In the past decade, digital learning has contributed to the transformation of K-12 education by using a variety of technology-enhanced pedagogical approaches, and it helps understand the basics of computational thinking (CT). In the area of CT for young learners, educators are experimenting with digital or digital-inspired methods to go beyond digital literacy, towards also improving other skills, such as problem-solving, logical thinking and abstraction. By improving these skills, we aim to empower learners with the required knowledge as technology users and to aid in mastering the technology to develop their creative and citizenship potential through them. This chapter will provide a literature review on studies conducted to teach computer programming and computational concepts to K-12 students using visual programming tools, unplugged activities and educational robotics while evaluating how it can also help improve CT skills.

Keywords: creative programming, unplugged activities, educational robotics, computational thinking, 21st-century skills, coding for young learners

Introduction

Across different OECD educational systems worldwide, K-12 education has transformed from being a textbook-based methodology to learning-focused pedagogy. To develop a learner-centered perspective, certain Technology-Enhanced Learning (TEL) practices have been implemented for the development of competency-based education aiming to go beyond cognitive learning objectives (Motschnig-Pitrik & Standl, 2013). In this context, the selected TEL practices aim to empower learners with a creative and socio-critical perspective of digital technologies (Resnick, 2017; Romero et al., 2017). In some cases, a learner-centered TEL approach has been facilitated by the possibilities of digital learning, along with learning resources and activities. The personalisation and creation of digital-inspired pedagogical concepts could provide learners with better access to and control over their learning experience. It could also provide educators with enhanced participation with subjects and a better awareness of learners' activities through learning analytic approaches (Siemens & Long, 2011).

When analysing the development of TEL approaches from a learning sciences perspective, certain studies focus on creating and using materials and tools in digital learning (Pachler et al., 2010), while others consider TEL from a creative perspective or as a way to develop computational thinking (CT) competency and a more enlightened approach to digital citizenship (Israel et al., 2015). This chapter goes beyond the idea of using existing digital learning materials (DLMs) and tools, to consider how learners can understand the underlying concepts of digital culture through the development of their CT competency in terms of formal systems (programming languages) and physical systems (sensors, actuators, and connectors, etc.) which interconnect as software and hardware.

Learning to code and CT development is a growing trend internationally (Chalmers, 2018; Faber et al., 2017). While it was earlier regarded as a suitable skill for advanced learners aiming for an IT career, today it is seen as an essential 21st-century skill that can be taught even to elementary students (Grover & Pea, 2013). From this perspective, projects such as Class'Code, a French initiative to introduce programming in schools, have developed training for educators, resources and an interdisciplinary learning community (Canellas et al., 2016). At the international level, numerous experiments have taken place in different contexts, although their systematic analyses have not been conducted. To support these learnings, this chapter will provide a reading and synthesis of different studies produced in educational science. Before introducing three different approaches in the way K-12 learners are introduced to CT, the concept definition and a few worldwide initiatives to introduce programming at K-12 will be presented in the next section.

From coding to the development of CT competency

Coding could be learned as a technological skill in which a set of instructions are organised into a program. When learning to code, learners are acquiring knowledge about some concepts and procedures through which they develop an awareness about formal coding system, but they do not develop CT competency as described by Wing (2006), who defines it as "solving problems, designing systems, and understanding human behavior, by drawing on the concept fundamental to computer science" (p. 33). When we engage learners beyond learning to code, we should provide them with problem situations so that they get to analyse the situation, select the type of code and physical systems to use and develop an iterative process of improvement.

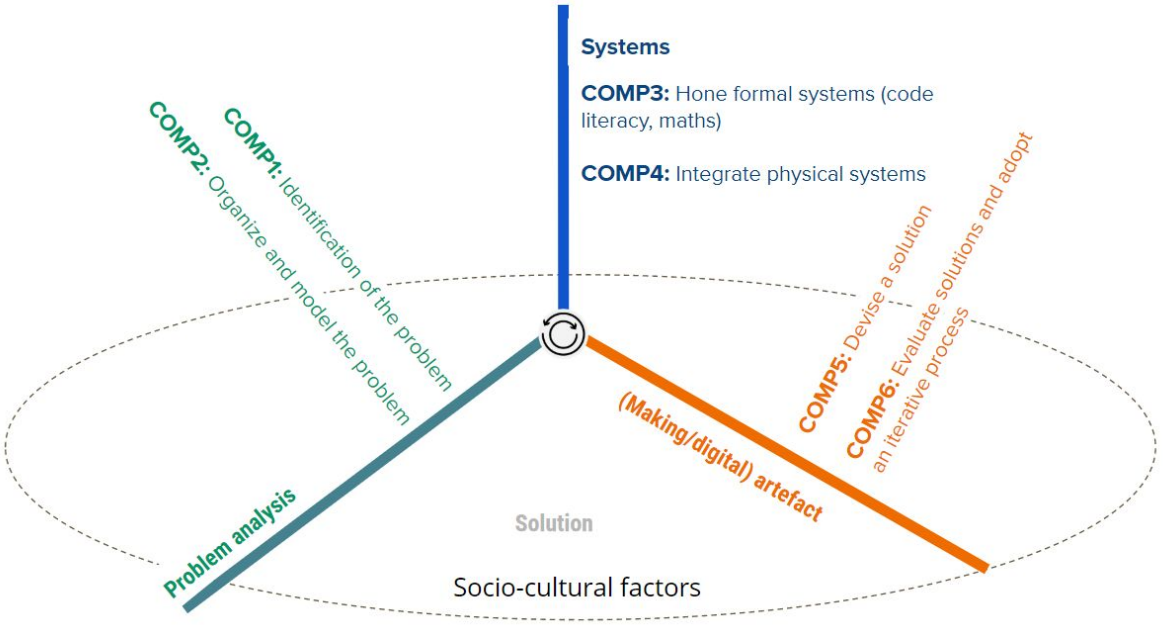

Figure 1. CT components (Romero et al., 2017)

CT engages components related to analysis of the problem situation and the way subjects organise and model the problem (problem analysis axis), honing formal systems with the use of a certain programming language and integration of physical systems (systems axis) and devices of an intermediate solution, its evaluation and improvement (creation axis). When learners are only engaged in coding they develop knowledge related to systems, but they do not engage in the full process of analysis, modeling and iterative creation of a solution (Romero et al., 2017).

Worldwide initiatives to introduce programming at K-12

Computer programming was first attempted to be integrated with the school curriculum in the 1980s (Popat & Starkey, 2019), but it started making an impact from 2010. Today, countries are adopting varied strategies when introducing coding to different learners. Chalmers (2018) mentions a research study conducted on four Australian primary school teachers to include coding and robotics in their classrooms through LEGO® WeDo® 2.0 kits. New College Worcester, England conducted a beta test with children with visual disabilities through Code Jumper, developed by Microsoft (Peters, 2019), a tool that uses "pods" or blocks with various shapes that can be connected into patterns and provides audio outputs. In another project by the British library's Young Researchers programme, two boys (aged 13 to 14 years) created a game in Mission Maker based on the poem *Beowulf* (Paula et al., 2018). The purpose of this activity was to foster learner creativity while integrating problem-solving skills with knowledge in subjects of Arts and Humanities through CT. Vaidyanathan (2013) also talks about a Bring-Your-Own-Device session she conducted with teachers at Los Altos School District in the USA using a "show-and-tell" method to emphasise that coding could be easily understood using everyday devices. Another innovative medium is used by Earsketch.gatech.edu, a virtual programming environment, that teaches coding in Python and JavaScript through music composing and runs annual competitions for interested students in the USA. There are others like KidsWhoKode.org, an Indian non-profit organisation, who follow a blended learning approach to teach coding to children aged 10 to 14 years from economically disadvantaged sections of society. They combine unplugged activities, online learning, classroom training, along with guest lectures and industry visits to make learners digitally empowered for the future (Kumari, 2018). In a very similar way, Kids Code Jeunesse in Canada supports teachers and educational professionals in developing CT skills (Romero et al., 2016). Another example from India demonstrates the educational value and socio-cultural usage of CT as a way-of-life (Dharavidiary.org; Chandran, 2016). After being mentored and tutored by a filmmaker, a group of adolescent girls have learned to code using a donated laptop, created an app for the betterment of their community and succeeded in expanding the project to more than 400 learners.

Through these different worldwide initiatives, we can observe the different approaches used by students to learn to code in schools and informal settings. The literature references for this chapter were acquired from sources, such as ScienceDirect.com and ResearchGate.net primarily from the years 2006 till current date. Special emphasis was laid on collecting information with real data analysis regarding the progress made in the teaching of computer programming and on CT using digital pedagogical methods. In the next section, we will describe the three approaches identified from these initiatives and a literature review on the way kids learn programming in schools.

Diversity in the way kids learn programming in schools

While continuing to grow at a fast pace, CT education also exhibits advanced and innovative implementation methods (Heintz et al., 2016), and educational sciences can now work on this object of study. We consider three approaches that are used to introduce kids to programming at the school level: unplugged computing, visual programming tools and educational robotics.

The first approach focuses on learning CT through *unplugged computing*. Cicirello (2013) describes unplugged activities in computer science as a way of teaching CT concepts with hands-on tools and activities rather than digital devices. Some studies are now interested in how CT can be imparted without the use of technology. They focus on how unplugged activities use pedagogy-based methods that help reduce the mental block most learners seem to have about code learning (Grover et al., 2018). Unplugged activities can help demystify CT for young and old learners. For example, KidsWhoKode.org conduct unplugged activities to start every coding session with games such as "Simon Says" to introduce conditional actions, get learners to write their daily activities to help

identify sequences and loops or use arts and crafts or music composition to understand basics of programming. Brackmann et al. (2017) also describe how methods, such as decomposition activity (breaking down real-world tasks into required steps), map activity (using four-directional arrow keys to move objects from point A to point B on a map), children's song activity (converting a song into an algorithm to identify variables, repetition and conditionals), etc., were tried on students aged 10 to 12 years from Madrid to introduce CT concepts. Most researchers seem to agree that unplugged activities can help learners understand these concepts (Bell et al., 2009; Moreno-Leon & Robles, 2015).

The second approach considers using *visual programming tools*, with an intent to develop creative computing skills, rather than just a "software developer". Since their introduction as a way to improve CT skills (Wilson & Moffat, 2010; Brennan & Resnick, 2012), a variety of programming tools, such as Scratch, Hopscotch and Kodable have come up in recent years. Among these, studies are focused primarily on Scratch (Moreno-León & Robles, 2015; Resnick & Siegel, 2015) and how it is used. Rose et al. (2017) conducted a study with a group of 40 children aged 6 and 7 years on whether programming tools can help develop CT skills in young learners. They used ScratchJr, a version of Scratch designed for younger learners (aged 5 to 7 years), and Lightbot, an educational puzzle that helps learners program a robot using block-based instruction. Since Lightbot only contains a subset of commands compared to Scratch and doesn't allow learners to explore the tool freely, the study concluded that Scratch provides learners with more freedom to experiment, thus leading to improved problem-solving ability. Similarly, Baker (2017) provides a comparison of features and capabilities between Scratch and Blockly. It appears that there are features provided by Blockly (including the option to translate programming blocks directly into other languages, such as JavaScript, PHP and Python) which might be lacking in the current version of Scratch. On the other hand, Scratch provides additional features, such as the ability to share code for completed projects within the user community (which would lead to a sense of contributing, knowledge sharing and self-directed learning), and the ease of using coding to create animations (which might easily attract and engage young learners).

The third approach is *educational robotics*, which offers the additional potential to learn how to interact with a connected or programmed object. Educational robotics not only allow improving skills regarding formal systems (component 3) but also manipulating and understanding physical systems (component 4). Several studies available establish the feasibility of making robotic devices with children. They show some virtues and limits too. Using robotics in educational activities seems to engage learners more than many of the other instructional methods (Papastergiou, 2009) because they get to play an active role in the learning process while using their creativity and problem-solving ability, thus increasing learner motivation. According to Carbonaro et al. (2004), learners get the opportunity to imbibe knowledge better since they get to collaborate with and showcase their robotic projects to a wider audience. In the process, learners also seem to acquire additional skills. For example, Benitti (2012) mentions that apart from helping learners understand robotics, it can also help improve skills, such as problem-solving, logical thinking and scientific inquiry. Castro et al. (2018) highlight another interesting observation that in a sample of 389 students selected to learn educational robotics, the success level of both girls and boys appeared to be the same. But while a few (Whittier & Robinson, 2007; Schopler & Toplis, 2008) experiment with using robotics to integrate subjects that are not related to physics and mathematics, most of the learning with educational robotics revolves around robotic concepts, such as "robot programming, robot construction, artificial intelligence, algorithm development and mechatronics" (Mitnik et al., 2008, p. 1). Some of the other possible drawbacks of educational robotics are covered by McNally et al. (2006), who categorise them into logistical and pedagogical disadvantages. Logistical disadvantage covers issues of cost and accessibility (it can only be offered to learners in limited numbers and controlled environments) and pedagogical disadvantage focuses on the limited concepts that are currently being covered through robotics-based education. Benitti (2012) also mentions that educational robotic activities predominantly use different versions of LEGO robots which are designed for children who are aged 7

years and older. It often appears that because of the ‘fun’ nature of these educational activities, they are mostly conducted as after-school or summer camp activities, and hence, might lack the quantitative data required to understand how educational robotics can help with the learning of subjects other than STEM better.

For an analysis on how learning sciences has approached CT (Wing 2008; Barr & Stephenson 2011), we will review studies in each of these three approaches to learn programming in K-12. The present review cannot claim to be exhaustive but aims to be representative of available knowledge. Because of a bias that favours these learnings, special attention has been paid to the few "critical" studies that also show limitations and disadvantages of these approaches. Meta-analyses or reviews of only studies with real factual results have been considered here, rather than "testimonials" as explained by Hickmott et al. (2017). It should be noted that all studies are cautious, including the earliest studies in education science on Logo programming language proposed by Papert (Clements & Meredith, 1992).

Approach 1: Unplugged activities to develop CT competencies

Until a few years ago, there seemed to be a lack of interest among students to learn computer science. This could be attributed to a few factors, including the assumed complexity in understanding computer science, the ‘let’s jump into programming’ learning approach used by most educators, and the fact that it was regarded by most learners as a highly intellectual and therefore unattainable skill and something only suitable for male learners (Taub et al., 2012; Bell et al., 2009). But with the persistent integration of unplugged activities, these assumptions seemed to be finally changing for a majority of learners (Bell et al., 2009; Bell, Witten, & Fellows, 1998). During unplugged or computerless activities, children learn concepts at the heart of computing in general or robotics in particular through play. Unplugged activities have been observed to be effective for learning CT (Brackmann et al., 2017).

According to Tricot (2017), computer thinking skills include knowing how to break down a problem into simpler subproblems, thinking about tasks to be done to solve a problem in terms of steps and actions (algorithm), describing problems and solutions at different levels of abstraction, which makes it possible to identify similarities between problems and subsequently to be able to reuse solution elements. Such improvement of these skills is confirmed by Moreno-Leon and Robles (2015) on about sixty pupils aged 9 to 11 years, after a work including an initiation into programming turned into a learning of algorithmic thought, with a key element: these teachers had just been trained in this new learning, and this was conducted in an English learning class. To establish these CT skills, Moreno-Leon et al. (2015) link each competency to the use of Scratch language constructions. They then propose a tool (Dr.Scratch.org) to assess the level of competence by using such constructions from the framework proposed by Brennan and Resnick (2012) included in a manual, such as Huseyin et al. (2017) to name only the most recent. In these studies, the leverage is in the shift from procedural learning of programming to multi-disciplinary integration of creative programming, as explained by Romero (2016). An example of skill transfer between learning to programme and other learning is given by the significant improvement of left-right distinction after a robot activity (Romero et al., 2016), and while other types of transfers are fairly likely, they are yet to be established. This positive effect on the ability to solve problems and, to a lesser extent, on reasoning and spatialisation is established on groups of pupils at the end of primary and middle school (Román-González et al., 2017). These results are consistent with the work of Chen et al. (2017), which include the use of educational robotics. The competency assessment on this subject has been finely studied by Lepage and Romero (2017) and Romero et al. (2017). An evaluation grid tool is used to assess and validate participants' engagement in solving problems of a certain complexity and authenticity in a critical, empathic and creative way while using strategies and processes of computer science to create one or more solutions.

Approach 2. Visual programming or learning to programme with Scratch

Scratch uses constructivist pedagogy which encourages learners to work with an artifact and tinker with it while completing a task, thus using that as a basis for building knowledge. Resnick et al. (2009) mention that most Scratch users belong to the age group of 8 to 16 years, though there are a lot of adult users too. They also define the purpose of Scratch as a medium not aimed at creating professional programmers, but one that could enable learners to use programming skills to interpret real-world situations creatively and systematically, while "using programming to express their ideas" (p.1).

Wilson and Moffat (2010) study 20 children aged 8 years in disadvantaged areas of Glåsgow and observe a positive experience through learning with Scratch to overcome frustrations and anxieties of learning, including children who learn well despite having difficulties. Gulbahar and Kalelioglu (2014) observe primary school youth for whom learning Scratch can increase programming skills, but with no increase in problem-solving skills in general, unless the authors' literature review shows it to design specific problem-solving activities from Scratch. Also, the use of Scratch on a group of college freshmen compared to traditional methods of learning on paper is not superior to the level of the results (Tekerek & Altan, 2014), but at the level of pleasure to learn and in generating student engagement. This study also shows the lack of difference in performance between girls and boys. The commitment of students is widely used in the studies of learning with Scratch (Ortiz-Colon & Romo, 2016) on a college-level class, even though the evaluation results are too scattered to lead to a definitive conclusion. What seems well established is Scratch's didactic value in terms of computer literacy as reported by Korkmaz (2016), wherein about 50 university entrance students were able to learn a language of professional programming, with this learning being significantly improved by prior use of Scratch. It has also been observed (Fesakis & Serafeim, 2009) that Scratch helped some post-baccalaureate students increase their self-confidence in digital tools and led to a positive influence on their digital uses in education. At the interface between programming learning and "tangible" computing, Horn et al. (2009) show that introducing programming using tangible objects (like a building set) through family workshops in a museum helped increase the performance of learners. This result is to be compared to what Boissel (2017) obtained with a tangible system for school initiation to programming for sighted and visually impaired children. Another study concerns the learning of programming with tangible objects and Correll et al. (2012) show the positive contribution on the commitment to study the sciences through questionnaires. It should also be noted that Scratch can lend itself to pedagogical approaches that are engaging for the child, such as the Scratch Community Blocks system, which allows children to analyse their learning data (Dasgupta & Hill, 2017). Learning to programme in Scratch is sometimes coupled with unplugged computing activities, tangible devices or participatory approaches, and in its broader conditions, there is a positive differentiation.

Approach 3. Educational robotics or usefulness of playful and educational robotics

Let us first consider available studies that analyse the feasibility of introducing educational robotics even to very young children. Sklar et al. (2003) attribute this to the ease with which "off-the-shelf robotic kits, such as LEGO Mindstorms, FischerTech-nik Mobile Robot and Elekit SoccerRobo" can be bought and introduced into classrooms (p. 5). As before, robotic activities are attractive but do not systematically offer leverage to help with other learning, especially if we are left with the acquisition of technological know-how. Gordon et al. (2015) conducted a study on how the use of a robot to help children learn something promotes/encourages their curiosity. The systematic literature review in educational robotics of Benitti (2012) simplifies our analysis as it shows that real improvements in learning are observed, but this is not always the case, the differentiating factor being the methodological (teacher's attitude and adapted workspace), and curriculum (which skills are targeted). The danger seems to be to make robotics for robotics, to remain at the level of a work of copying

technology. Few studies look, beforehand, if the handling of the robotic system is easy, contrary to Desprez et al. (2018) who take the trouble to validate the usability of their device. This study also shows that out of 70 publications analysed, only around 10 or so include a real quantified study beyond statements a priori, or occasional testimonials, such as Robinson (2005), which are not devoid of interest, but whose validity, in general, is questionable. More recently, Kim et al. (2015) show how robotic experimentation is used to teach and learn science, technology, engineering and mathematics, linking these disciplines, and resulting in the motivational commitment of the teacher. The link between educational robotics and CT (see next section) is made by Atmatzidou and Demetriadis (2016) who note the need for long-time learning, with consistency in quality of performance regardless of age or gender, with the fact that girls take longer to complete a process (we note here an over-interpretation of the authors who refer to girl students and say that "they need more time to achieve the same result" while facts show "that they spend more time", and if the time had been constrained there is no proof that the results would have been less good). In a smaller study, Bers et al. (2014) establish the capacity of about fifty children (kindergarten level) to appropriate the first notions of robotics adapted to their age. A qualitative observation (Sullivan et al., 2013) confirms the feasibility of introducing the youngest (in this case, pre-kindergarten level) to educational robotics. More specifically, based on interviews, Highfield et al. (2008) show learning in spatialisation and geometry from the manipulation of a programmable object and activity involving the body. More recently Spolaôr and Benitti (2017) make a systematic literature review of the use of educational robotics for other learning and conclude on the real potential, established in several cases, especially when the approach is combined with a constructivist pedagogical approach. In his work on IniRobot resources with Thymio-II, Roy (2015) shows a real increase of skills on a group of 24 primary-level children with a final score of 93% at the post-test establishing that learning is accessible to everyone. The educational work with Thymio-II demonstrating its ability to engage primary and middle school children was established by the creators of the robot (Riedo et al., 2013), including the ability to appropriate basics of robotics (Magnenat et al., 2012).

Discussion

Comparison criteria	**Unplugged activities**	**Visual programming tools**	**Educational robotics**
Tools required	● Hands-on tools and activities ● No digital tools ● No internet	● Digital software ● Internet	● Educational robot (usually a programmable toy) ● Digital software ● Internet
Suitability	Suitable for learners of all ages	Suitable for learners aged 7 years and above	Suitable for learners aged 7 years and above
Prior knowledge required	No prior knowledge of programming or computers required	Requires basic knowledge about computers from a user perspective	Requires basic knowledge about programmable interfaces (such as Thymio or Beebot)
Investment required	Very inexpensive compared to other approaches	Requires investment in computers	Requires investment in educational robots (and computers, if software is required)
CT components covered	All CT components could be introduced, some of them in a metaphorical way (physical systems - hardware)	All CT components except physical systems (hardware), but there is often a focus on formal systems (code literacy)	All CT components could be covered
Benefits	● Useful in introducing initial	● Focuses on developing	Could offer the following benefits:

	coding concepts ● Could help in reducing inhibitions that learners might have about code learning ● Can be used anywhere with any material available (free resources for activities available online)	creative computing skills ● Programming tools, such as Scratch, are free ● Has the potential to be a fun and engaging method for learners ● Learners get real-time feedback through a visual representation of their activities ● Can be used to integrate STEM subjects	● Provide advanced learning about coding and hardware ● Allow learners to display their creation to others ● Help improve other skills, such as problem-solving, logical thinking and scientific inquiry
Limitations	● Doesn't offer any experience with hardware or software ● Doesn't provide hands-on practice in coding ● Unless learning objectives are briefed, learners might not understand the concepts being represented through the unplugged activities	● Even if software is free (e.g. Scratch), computers have a cost that might not be suitable for learning environments with limited monetary resources ● Doesn't cover physical systems ● While learners may be able to reuse or modify existing codes, they might be unable to write own codes	● Involves higher cost than other approaches ● Can only be offered to a few learners at a time ● Most of the learning centres around robotic concepts

Figure 2: Features, advantages and limitations of the three approaches to CT development

The above table presents a comparison of features, benefits and limitations between the three approaches used in developing CT skills among K-12 learners. The review conducted in this paper shows that all three approaches have certain limitations as well as unique benefits. Educators need to take into account a few critical factors when considering the most suitable approach in their educational environment. The first and primary limitation faced by most educators is in terms of availability of resources and existing infrastructure. For schools from economically disadvantaged sections of society, it may be a challenge to use the educational robotics approach, since they require more investment. In such cases, using unplugged activities or visual programming tools would be more suitable for their purpose. One also needs to consider the CT competencies that are to be integrated and the level of expertise that the educator wants learners to master. From the mentioned examples, it appears that while unplugged activities might be a good approach to introduce CT competencies to learners, visual programming tools help learners understand them and practice their use in creative ways, and educational robotics provide them with an advanced learning experience.

By learning how to "use computers in the solution of life problems for production purposes" (Korkmaz et al., 2017) learners, particularly school children, will gain something essential regarding their education, and much more. That is why CT initiation is beyond learning to programme or controlling robots. This statement has to be supported with concrete proof, and the previous literature review brings some positive answers to this statement. Indeed, for this statement to be meaningful, the concept of CT must be well defined. ISTE (2015) defines CT as the common reflection of creativity, algorithmic thinking, critical thinking, problem-solving, cooperative thinking and communication skills. When these skills are taken into consideration together, they explain a brand-new thinking skill that is called CT (Korkmaz et al., 2017). When considering the potential of the three approaches to aid in developing CT skills that go beyond technical skills and integrate other essential skills mentioned above, it can be seen from the reviewed studies that all the three approaches appear to be promising.

What can and cannot be expected from CT learning

The present review tends to show that learning to code can promote autonomy and develop skills

regarding problem-solving and research approach, but it is not automatic as it requires support in these two components. For instance, one track is to explain a solution to students to enable them to build the next one, such as a variant, and allowing them to take inspiration from existing solutions to be adapted, which could help reduce the large gap imposed by autonomous learning. Learning to code is considered by some authors as a relevant and effective skill that requires the know-how to manipulate. However, learning CT also requires notional knowledge learning, thus being cognitively active is very important. Group and project learning can address multiple skills, but can also fail, if not well managed. This learning can be fostered by helping students organise themselves, or by giving them roles and defining progressive stages. Such an approach makes it possible to engage pupils in the activities, and their perception a posteriori is very positive. Better engagement of students in their learning activities, thanks to the interest generated by CT activities, may be considered as the ultimate positive effect of such a paradigm.

Conclusion

When we go beyond simple learning of programming, for example through unplugged computing activities or by promoting learning of algorithmic thinking, we can achieve a real positive effect at the primary level and beginning of college, including with newly trained teachers. The leverage here is the transition from procedural learning of programming to multi-disciplinary integration of creative programming. The studies covered in this chapter point to positive learning outcomes in terms of CT, but further research needs to be developed to analyse the learning transfer to other disciplines. The analysis of current practices in the way programming is introduced in K-12 education has led to a huge diversity in terms of pedagogical strategies.

In many cases, all three approaches are combined in practice, such as by performing an unplugged activity or manipulating a pedagogical robot, before learning programming. However, to our best knowledge, the combination of these three approaches has not been studied formally. To this end, future work is needed to design a paradigm to study how to combine these different approaches in an integrated environment to engage learners using unplugged activities along with other methods, such as by using tangible robotic programming tasks. We hope to be able to do this through an escape-game inspired experience. This is the goal of a future ANR #CreaMaker/AIDE project structured to observe the creative aspects of CT as well as to collect automatic logs that will track the components of CT competencies. Similarly, efforts in the emerging field of computational learning sciences are also combining selective loosely-defined tasks within a context in which the analysis of sensors and logs would permit integration of machine learning approaches to firstly analyse user competencies and secondly to help learners develop their learning process further by providing cues at appropriate moments for external and internal learning regulations. These few results in the learning sciences seem to confirm the combined approach of unplugged computing activities, visual programming tools and educational robotics projects such as Class'Code.

Acknowledgement

We aim to acknowledge the support of EducAzur cluster, INRIA and the ANR #CreaMaker (ANR-18-CE38-0001) project for supporting this chapter development.

References

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Baker, S. (2017). Blockly VS Scratch: What's best for me? Retrieved 13 April 2019, from

https://www.robotlab.com/blog/blockly-vs-scratch-whats-best-for-me
Bell, T. C., Alexander, J., Freeman, I., & Grimley, M. (2009). (PDF) Computer Science Unplugged: school students doing real computing without computers. Retrieved 12 April 2019, from ResearchGate website: https://www.researchgate.net/publication/266882704_Computer_Science_Unplugged_school_students_doing_real_computing_without_computers
Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, *58*(3), 978–988. https://doi.org/10.1016/j.compedu.2011.10.006
Barr, Valerie, and Chris Stephenson. 2011. "Bringing Computational Thinking to K-12: What Is Involved and What Is the Role of the Computer Science Education Community?" *ACM Inroads* 2 (1): 48. https://doi.org/10.1145/1929887.1929905.
Wing, Jeannette M. 2008. "Computational Thinking and Thinking about Computing." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 366 (1881): 3717–3725.
Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020
Boissel, S. (2017). Programmer les yeux fermés. Retrieved 5 April 2019, from binaire website: http://binaire.blog.lemonde.fr/2017/05/22/apprendre-a-programmer-les-yeux-fermes/
Brackmann, C. P., Roman-Gonzalez, M., Robles, G., & Moreno-Leon, J. (2017). (PDF) Development of Computational Thinking Skills through Unplugged Activities in Primary School. http://dx.doi.org/10.1145/3137065.3137069
Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. 25.
Canellas, C., De La Higuera, C., Peinchaud, É., & Roche, M. (2016). Class' Code a un an... et c'est un commencement. *1024 – Bulletin de La Société Informatique de France*, (9), 19–24.
Carbonaro, M., Rex, M., & Chambers, J. (2004). *Using LEGO robotics in a project-based learning environment*. 7.
Castro, E., Cecchi, F., Valente, M., Buselli, E., Salvini, P., & Dario, P. (2018). Can educational robotics introduce young children to robotics and how can we measure it? Retrieved 12 April 2019, from ResearchGate website: https://www.researchgate.net/publication/327470707_Can_educational_robotics_introduce_young_children_to_robotics_and_how_can_we_measure_it
Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93–100. https://doi.org/10.1016/j.ijcci.2018.06.005
Chandran, R. (2016). Girls code in India to tackle challenges of slum living - Reuters. Retrieved 13 April 2019, from https://in.reuters.com/article/india-children-tech/girls-code-in-india-to-tackle-challenges-of-slum-living-idINKCN0YI20F
Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, *109*, 162–175. https://doi.org/10.1016/j.compedu.2017.03.001
Cicirello, V. (2013). (PDF) A CS Unplugged Activity for the Online Classroom. Retrieved 12 April 2019, from ResearchGate website: https://www.researchgate.net/publication/244988711_A_CS_Unplugged_Activity_for_the_Online_Classroom
Clements, D. H., & Meredith, J. S. (1992). *Research on Logo: Effects and Efficacy*. 15.
code.org. (n.d.). Retrieved 12 April 2019, from https://code.org/curriculum/unplugged
Correll, N., Wailes, C., & Slaby, S. (2012). A One-Hour Curriculum to Engage Middle School

Students in Robotics and Computer Science Using Cubelets. *DARS*. https://doi.org/10.1007/978-3-642-55146-8_12

Dasgupta, S., & Hill, B. M. (2017). Scratch Community Blocks: Supporting Children as Data Scientists. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*, 3620–3631. https://doi.org/10.1145/3025453.3025847

Paula, B. H., Burn, A., Noss, R., & Valente, J. A. (2018). Playing Beowulf: Bridging computational thinking, arts and literature through game-making. *International Journal of Child-Computer Interaction*, *16*, 39–46. https://doi.org/10.1016/j.ijcci.2017.11.003

Desprez, T., Noirpoudre, S., Segonds, T., Caselli, D., Roy, D., & Oudeyer, P.-Y. (2018). *Design and Evaluation of Pedagogical Robotic Kits*. Retrieved from https://hal.archives-ouvertes.fr/hal-01688310

Dharavidiary.org. (n.d.). Retrieved 13 April 2019, from https://www.dharavidiary.org/

Dr.Scratch.org. (n.d.). Retrieved 4 April 2019, from http://www.drscratch.org/

earsketch.gatech.edu. (n.d.). Retrieved 9 April 2019, from http://earsketch.gatech.edu/landing/#/

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017). Teaching Computational Thinking to Primary School Students via Unplugged Programming Lessons. *Journal of the European Teacher Education Network*, *12*, 13–24.

Fesakis, G., & Serafeim, K. (2009). Influence of the Familiarization with 'Scratch' on Future Teachers' Opinions and Attitudes About Programming and ICT in Education. *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education*, 258–262. https://doi.org/10.1145/1562877.1562957

Gordon, G., Breazeal, C., & Engel, S. (2015). Can Children Catch Curiosity from a Social Robot? *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction - HRI '15*, 91–98. https://doi.org/10.1145/2696454.2696469

Grover, S., Jackiw, N., Lundh, P., & Basu, S. (2018). (PDF) Combining Non-Programming Activities with Programming for Introducing Foundational Computing Concepts. Retrieved 12 April 2019, from ResearchGate website: https://www.researchgate.net/publication/326539412_Combining_Non-Programming_Activities_with_Programming_for_Introducing_Foundational_Computing_Concepts

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Gulbahar, Y., & Kalelioglu, F. (2014). (PDF) The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. Retrieved 5 April 2019, from ResearchGate website: https://www.researchgate.net/publication/271746747_The_Effects_of_Teaching_Programming_via_Scratch_on_Problem_Solving_Skills_A_Discussion_from_Learners'_Perspective

Heintz, F., Mannila, L., & Färnqvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K-12 education. *Frontiers in Education Conference (FIE), 2016 IEEE*, 1–9. IEEE.

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2017). (PDF) A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms. Retrieved 11 April 2019, from ResearchGate website: https://www.researchgate.net/publication/321760448_A_Scoping_Review_of_Studies_on_Computational_Thinking_in_K-12_Mathematics_Classrooms

Highfield, K., Mulligan, J., & Hedberg, J. (2008). Early mathematics learning through exploration with programmable toys. *Proceedings of the Joint Meeting of PME*, *32*, 169–176. Citeseer.

Horn, M. S., Solovey, E. T., Crouser, R. J., & Jacob, R. J. K. (2009). Comparing the use of tangible and graphical programming languages for informal science education. *Proceedings of the 27th International Conference on Human Factors in Computing Systems - CHI 09*, 975. https://doi.org/10.1145/1518701.1518851

Huseyin, O., Gary, W., & Tugba, O., H. (2017). *Teaching Computational Thinking in Primary*

Education. IGI Global.
Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, *82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022
ISTE. (2015). Computational Thinking Leadership Toolkit. Retrieved 5 April 2019, from https://id.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf
KidsWhoKode.org. (n.d.). Retrieved 10 April 2019, from <KidsWhoKode /> website: https://www.kidswhokode.org/
Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers & Education*, *91*, 14–31. https://doi.org/10.1016/j.compedu.2015.08.005
Korkmaz, Ö. (2016). The Effects of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *International Journal of Modern Education and Computer Science*, *8*(1), 16–23. https://doi.org/10.5815/ijmecs.2016.01.03
Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005
Kumari, B. (2018, April 9). KidsWhoKode aims to be a student-driven movement: Code of small things [News]. Retrieved 10 April 2019, from Bangalore Mirror website: https://bangaloremirror.indiatimes.com/bangalore/cover-story/kidswhokode-aims-to-be-a-student-driven-movement-code-of-small-things/articleshow/63672465.cms
Lepage, A., & Romero, M. (2017). (PDF) Évaluation par compétences d'activités de programmation créative avec l'outil #5c21. Retrieved 5 April 2019, from ResearchGate website: https://www.researchgate.net/publication/320310837_Evaluation_par_competences_d'activites_de_programmation_creative_avec_l'outil_5c21
Magnenat, S., Riedo, F., Bonani, M., & Mondada, F. (2012). A programming workshop using the robot "Thymio II": The effect on the understanding by children. *2012 IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO)*, 24–29. https://doi.org/10.1109/ARSO.2012.6213393
McNally, M., Goldweber, M., Fagin, B., & Klassner, F. (2006). Do LEGO Mindstorms robots have a future in CS education? *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '06*, 61. https://doi.org/10.1145/1121341.1121362
Mitnik, R., Nussbaum, M., & Soto, A. (2008). (PDF) An autonomous educational mobile robot mediator. Retrieved 10 April 2019, from ResearchGate website: https://www.researchgate.net/publication/200744655_An_autonomous_educational_mobile_robot_mediator
Moreno León, J., Robles, G., & Román-González, M. (2016). Code to Learn: Where Does It Belong in the K-12 Curriculum? *Journal of Information Technology Education: Research*, *15*, 283–303. https://doi.org/10.28945/3521
Moreno-León, J., & Robles, G. (2015). Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects. *Proceedings of the Workshop in Primary and Secondary Computing Education*, 132–133. ACM.
Moreno-Leon, J., & Robles, G. (2015). (PDF) Computer programming as an educational tool in the English classroom: a preliminary study. http://dx.doi.org/10.1109/EDUCON.2015.7096089
Moreno-Leon, J., Robles, G., & Roman-Gonzalez, M. (2015). (PDF) Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. Retrieved 5 April 2019, from ResearchGate website: https://www.researchgate.net/publication/281714025_Dr_Scratch_Automatic_Analysis_of_Scratch_Projects_to_Assess_and_Foster_Computational_Thinking
Ortiz-Colon, A. M., & Maroto Romo, J. L. (2016). Teaching with Scratch in Compulsory Secondary Education. *International Journal of Emerging Technologies in Learning (IJET)*, *11*(02), 67.

https://doi.org/10.3991/ijet.v11i02.5094
Pachler, N., Cook, J., & Bachmair, B. (2010). Appropriation of mobile cultural resources for learning. *International Journal of Mobile and Blended Learning*, *2*(1), 1–21.
Papastergiou, M. (2009). Exploring the potential of computer and video games for health and physical education: A literature review. *Computers & Education*, *53*(3), 603–622. https://doi.org/10.1016/j.compedu.2009.04.001
Peters, A., (2019, January 22). These tactile blocks teach blind kids to code. Retrieved 8 April 2019, from Fast Company website: https://www.fastcompany.com/90294418/these-tactile-blocks-teach-blind-kids-to-code
Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128*, 365–376. https://doi.org/10.1016/j.compedu.2018.10.005
Resnick, M. (2017). Fulfilling Papert's Dream: Computational Fluency for All. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 5–5. ACM.
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … others. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60–67.
Resnick, M., & Siegel, D. (2015, November 10). A Different Approach to Coding. Retrieved from Bright website: https://medium.com/bright/a-different-approach-to-coding-d679b06d83a#.b9jcw2js5
Riedo, F., Chevalier, M., Magnenat, S., & Mondada, F. (2013). (PDF) Thymio II, a robot that grows wiser with children. http://dx.doi.org/10.1109/ARSO.2013.6705527
Robinson, M. (2005). Robotics-Driven Activities: Can They Improve Middle School Science Learning? *Bulletin of Science, Technology & Society*, *25*(1), 73–84. https://doi.org/10.1177/0270467604271244
Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047
Romero, M. (2016a, May 5). La pensée informatique. Retrieved 5 April 2019, from #CoCreaTIC website: https://margaridaromero.wordpress.com/2016/05/05/la-pensee-informatique/
Romero, M. (2016b, May 25). De l'apprentissage procédural de la programmation à l'intégration interdisciplinaire de la programmation créative. Retrieved 5 April 2019, from #CoCreaTIC website: https://margaridaromero.wordpress.com/2016/05/25/de-lapprentissage-procedural-de-la-programmation-a-lintegration-interdisciplinaire-de-la-programmation-creative/
Romero, M., Davidson, A.-L., Cucinelli, G., Ouellet, H., & Arthur, K. (2016). Learning to code: from procedural puzzle-based games to creative programming. *Learning and Teaching Innovation Impacts*. Presented at the International Conference on University Teaching and Innovation, Barcelona, Spain.
Romero, M., Dupont, V., & Pazgon, E. (2016). À gauche ou à droite du robot ? Test de perspective décentrée gauchedroite par le biais d'une activité sur papier et d'une activité de robotique pédagogique. Retrieved 5 April 2019, from CIRTA website: http://www.cirta.org/index.php/50-banque-de-textes-actes-colloque-2016/221-a-gauche-ou-a-droite-du-robot-test-de-perspective-decentree-gauche-droite-par-le-biais-d-une-activite-sur-papier-et-d-une-activite-de-robotique-pedagogique
Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*(1), 42. https://doi.org/10.1186/s41239-017-0080-z
Romero, M., Lille, B., & Patino, A. (Eds.). (2017). *Usages créatifs du numérique pour l'apprentissage au XXIe siècle* (Vol. 1). Québec: Presses de l'Université du Québec.
Rose, S. P., Habgood, M. P. J., & Sheffield, T. J. (2017). *(PDF) An Exploration of the Role of Visual Programming Tools in the Development of Young Children's Computational Thinking*. Retrieved from www.ejel.org/issue/download.html?idArticle=602

Roy, D. (2015). *Optimisation des parcours d'apprentissage à l'aide des technologies numériques* (Phdthesis, CNAM, Paris). Retrieved from https://tel.archives-ouvertes.fr/tel-01252695/document

Schopler, E., & Toplis, R. (2008). Journal of Autism and Developmental Disorders. Retrieved 10 April 2019, from ResearchGate website: https://www.researchgate.net/publication/314067414_Journal_of_Autism_and_Developmental_Disorders

Siemens, G., & Long, P. (2011). Penetrating the fog: Analytics in learning and education. *EDUCAUSE Review*, *46*(5), 30.

Sklar, E., Eguchi, A., & Johnson, J. (2003). RoboCupJunior: Learning with Educational Robotics. In A. Bredenfeld, A. Jacoff, I. Noda, & Y. Takahashi (Eds.), *RoboCup 2005: Robot Soccer World Cup IX* (Vol. 4020, pp. 238–253). https://doi.org/10.1007/978-3-540-45135-8_18

Spolaôr, N., & Benitti, F. B. V. (2017). Robotics applications grounded in learning theories on tertiary education: A systematic review. *Computers & Education*, *112*, 97–107. https://doi.org/10.1016/j.compedu.2017.05.001

Sullivan, A., R. Kazakoff, E., & Umashi Bers, M. (2013). The Wheels on the Bot go Round and Round: Robotics Curriculum in Pre-Kindergarten. *Journal of Information Technology Education: Innovations in Practice*, *12*, 203–219. https://doi.org/10.28945/1887

Taub, R., Armoni, M., & Ben-Ari, M. (2012). (PDF) CS Unplugged and Middle-School Students' Views, Attitudes, and Intentions Regarding CS. Retrieved 12 April 2019, from ResearchGate website: https://www.researchgate.net/publication/241623893_CS_Unplugged_and_Middle-School_Students%27_Views_Attitudes_and_Intentions_Regarding_CS

Tekerek, M., & Altan, T. (2014). The effect of Scratch environment on student's achievement in teaching algorithm. *Educational Technology.*, 7.

Tricot, A. (2017). L'innovation pédagogique : Mythes et réalités. Retrieved 5 April 2019, from http://www.cafepedagogique.net/LEXPRESSO/Pages/2017/09/05092017Article636401937485284884.aspx

Vaidyanathan, S. (2013, December 9). We Need Coding in Schools, but Where are the Teachers? - EdSurge News. Retrieved 9 April 2019, from EdSurge website: https://www.edsurge.com/news/2013-12-09-opinion-we-need-coding-in-schools-but-where-are-the-teachers

Whittier, E., & Robinson, M. (2007). Teaching Evolution to Non-English Proficient Students by Using Lego Robotics. Retrieved 10 April 2019, from ResearchGate website: https://www.researchgate.net/publication/234707215_Teaching_Evolution_to_Non-English_Proficient_Students_by_Using_Lego_Robotics

Wilson, A., & Moffat, D. C. (2010). *Evaluating Scratch to introduce younger schoolchildren to programming*. 12.