Journal of Science and Technology
ISSN: 2456-5660 Volume 5, Issue 4, July-August 2020

www.jst.org.in *DOI: https://doi.org/10.46243/jst.2020.v5.i4.pp40-53*

Codecraft Competition: Learning to Code Through Contests Using Scratch

Anand Pultoo[1*], Anoop Sharma Bullee[2], Jean Norbert Meunier[2], Kaviraj Sheoraj[2], Shabneez Panchoo[2], Parveen Naseeven[2], Mohundranathsingh Ujoodha[2], Vikramsing Roocha[2], Hrishant Rajcoomar[2], Avinash Oojorah[2]

[1, 2]Mauritius Institute of Education, Reduit, Mauritius
**Corresponding Author: a.pultoo@umail.mie.ac.mu*

To Cite this Article
Pultoo, A., Bullee, A. S., Norbert, J. N., Sheoraj, K., Panchoo, S., Naseeven, P., Ujoodha, M., Roocha, V., Rajcoomar, R., & Oojorah, A. "Codecraft competition: Learning to code through contests using Scratch", Journal of Science and Technology, Vol. 05, Issue 04, July-August 2020, pp40-53

Article Info
Received: 17-03-2020 Revised: 10-06-2020 Accepted: 18-06-2020 Published: 22-06-2020

Abstract: *In this article, the design principles that guided the development of the Codecraft competition are discussed together with strategies for making the contest engaging for everyone. A random selection of 30 educators and 30 students who participated in the contests were also interviewed to find out their views about the competition and its impact on their computational thinking and creativity skills. Furthermore, 85 projects made up of codes from the Scratch program and submitted during the final round of the contest during the years 2016 to 2018 were processed using an online analytical tool known as Dr Scratch. To identify the progress made by the students, the different computational thinking concepts were analysed more deeply. The study revealed that over three years of the Codecraft competition, there was a significant improvement regarding the use of programming concepts, logic, and computational practices. Moreover, the average computational thinking (CT) score of the projects increased from 11.8 in 2016 to 14.2 in 2017 and 16.0 in 2018. Learning of computational concepts through an active contest-based approach also highlighted several advantages, such as motivation, fun, commitment, and enthusiasm, showing improvements related to creativity, computational thinking, and computational practices.*

Keywords*: Coding Contest, Scratch, Mauritius, Computational Thinking, Collaboration, Creativity*

I. Introduction

In a rapidly changing modern society and the computer-pervasive digital world with diverse opportunities and challenges, children need a new set of skills to succeed. Robotics, artificial intelligence, and automation technologies are among the high-growth businesses of the future [1], and people who will thrive in those industries need to be fluent in the coding languages behind them. However, the importance of learning programming languages goes beyond coding with fluency. The biggest issues are the development of analytical skills, systematic reasoning, and creativity. Learning programming languages makes people think in a very different way, algorithmically, to use abstraction to understand complex systems and the interconnection between them [2].

Coding competitions and games are a great way for students to apply their coding skills in a fun context, and to create something fascinating for them. Participants get that real-world experience of coding to solve a problem, which is essential to enhance their analytical skills and creativity [3-5]. The task can feel even more daunting in the open-ended challenges proposed during the finals of the competition, where the level of complexity is higher. Consequently, the perseverance and grit they develop can be of real benefit for life in the 21st-century.

1.1 Theoretical Background

It has become commonplace to refer to young people as 'digital natives' due to their apparent fluency with digital technologies [6, 7]. Young people are at ease browsing the Web, playing online games, sending text messages, and sharing videos. However, young people also need to learn some type of programming as it provides important benefits to them. For example, it adds the capability to independently learn and use new technology as it evolves, including the active use of computing [8]. It also expands the range of learning avenues. In particular, programming supports "computational thinking," helping people learn important problem solving and design strategies (such as modularization and iterative design) that carry over to non-programming domains. Wing [9] defined computational thinking (CT) as a skill that "*involves solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science*" and also stated "*Computational thinking is a fundamental skill for everyone, not just for computer scientists*" [10]. Šerbec et al., described CT skill as a problem-solving process that embraces logical, analytical, algorithmic thinking and dispositions, such as the talent to effortlessly deal with intricacy and open-ended problems [11]. Thus, CT skill is asignificant asset for learners. Educational institutions around the world are trying to include the development of this competence in schools [12].

When personal computers were first introduced in the late 1970s, there was initial enthusiasm for teaching all children how to program. Millions of students in secondary schools were taught to write simple programs in Logo or Basic. Seymour Papert, famous for constructionism (a learning theory that combined conceptual constructivism with concrete experience) presented in his book "*Mindstorms*" [13] Logo programming as a cornerstone for rethinking approaches to education and learning. Papert worked closely with Jean Piaget at the University of Geneva in Switzerland during the late 1950s and early 1960s. Papert based his theory of constructionism on the constructivist epistemology.

> *Constructionism - the N-word as opposed to the V-word - shares constructivism's view of learning as 'building knowledge structures' through progressive internalization of actions… It then adds the idea that this happens especially felicitously in a context where the learner is consciously engaged in constructing a public entity, whether it is a sandcastle on the beach or a theory of the universe* [14]. (Papert & Harel, p.1).

Constructionism views young learners as mini-scientists and inventors who develop their knowledge of how the world works by building theories and experimenting [15]. Constructionism maintains that learning can happen most effectively when people are also active in making real-world tangible objects. In the forward to the book "*Mindstorms: Children, Computers, and Powerful Ideas*", Papert [13] illustrated how his fascination as a child for model cars and how they work led him to experiment with gears. Papert went on to found the Life Long Kindergarten at MIT and to come up with Logo, a programming language for children.

In the 1980s, many started coding using Logo, the educational computer programming software that has the famous 'turtle' on the computer screen. Students and teachers were enthusiastic about thenew programming software. Through the 1990s it remained popular in schools, but since people started using computers mainly for other purposes than programming. Nowadays, though computers is omnipresent in children's lives, few of them use it for programming. Nowadays, computer programming is mostly viewed as a technical activity, suitable for only a small segment of the population. Baist & Pamungkas [16] pointed out that there are significant difficulties experienced by students in understanding the basic concept of programming and in finding faults of their own programs. What happened to the initial passion for introducing programming to children? Why did Logo and other programming software not live up to their initial potential?

There were several factors:

- Early programming languages were too challenging to use, and many students basically couldn't acquire the syntax of programming;
- Programming was frequently introduced with activities (such as generating curves and list of prime numbers) that were not in line with pupils' interests;
- No one could provide guidance when programming activities went wrong;
- Deeper explorations were not encouraged when things went right.

Papert claimed that programming languages should have a "low floor" (initially easy) and a "high ceiling" (become more complex over time). In addition, languages need "wide walls" (supporting many different types of projects so people with many different interests and learning styles can all become engaged) [17]. Thus, to have this combination of "low floor", "high ceiling", and "wide walls", the organizing committee decided that the participants in the Codecraft contest should use Scratch 2.0, a free block-based visual programming language and online community developed at the MIT Media Lab. Scratch 2.0 is inspired by programming languages for young people like LOGO and Squeak Etoys [18]. Although based on languages targeting the young generation, Scratch was designed to be different, less complex, and more instinctive [17, 19]. It enables programming interactive stories, games, and animations by simply using drag and drop blocks to perform different commands or actions (Figure 1) [20, 21]. Translated into 70+ languages, it is one of the most popular programming languages for young people with more than54 million users, over 53 million shared projects worldwideas of May 2020 [22-24]. The primary goal of Scratch is to help children (ages 8 and up) develop essential 21st-century learning competencies [25].

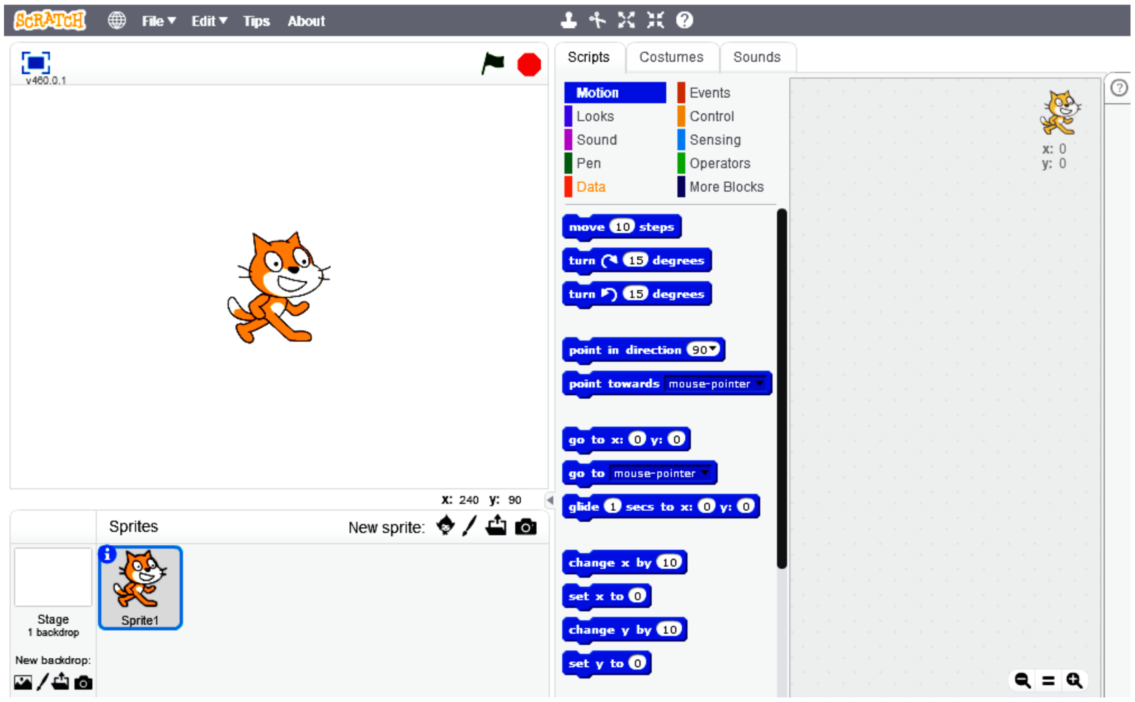


Figure 1: Scratch 2.0 programming environment

The Scratch grammar is based on a collection of graphical “programming blocks” that users snap together to create programs (Figure 1). The blocks have connectors indicating how they should be assembled together. Users can start by simply tinkering with the bricks, snapping them together in different sequences and combinations to see what happens. [19, 26] There is none of the obscure syntax or punctuation of traditional programming languages. The floor is low and the experience playful. Furthermore, the shape of Scratch blocks enables them to fit together only in ways that make syntactic sense. Control structures (like ‘forever’ and ‘repeat’) are C-shaped indicating that other blocks can be placed inside them. The shape of blocks with output values are related to the types of values they return: ovals for numbers and hexagons for Booleans. Conditional blocks (like ‘if’ and ‘repeat-until’) have hexagon-shaped cavities, indicating that a Boolean is needed.

Scratch is designed to be highly interactive - mixing graphics, animations, photos, music, and sound. As soon as a stack of blocks is clicked, it starts executing its code immediately. Users can even make changes to a stack as it is running, so it is easy to experiment with new ideas incrementally and iteratively. If users want to create parallel threads, they can simply create multiple stacks of blocks. This makes parallel execution as intuitive as sequential execution [27]. The scripting area in the Scratch interface is intended to be used as a physical desktop (see Figure 3). Users can even leave extra blocks or stacks lying around in case you need them later. The implied message is that it is OK to be a little messy and experimental. Scratch supports a wide variety of projects and interests (storytelling, gamification, animations, and simulations) [28]. Moreover, it is easy for people to personalize their Scratch projects by importing photos and music clips, recording voices, and creating graphics [29].

Though Scratch has the potential to be a powerful tool, it is often used as little more than a presentation tool in the classroom. Few Scratch projects use variables or control flow data structures. Many users do not advance to a higher level on their own, though the Scratch setting provides a 'low floor, high ceiling' that allows learners to explore the environment without frustration. However, tools like Scratch can empower students to display their creativity like never before; furthermore, the way these tools are taught by teachers and used by students significantly influences whether students move along the creativity continuum [28].

1.2 The Codecraft Competition

The competition aimed to develop computational thinking and 21st-century skills among secondary school learners of grade 9 (3rd year of secondary schooling). Undeniably, computational thinking has the potential to provide users opportunities to extend their creative expression to solve problems, create computational artifacts, and develop new knowledge. The ubiquitous nature of computing and availability of digital tools is also transforming education as pupils move from being mere consumers of content to producers in the subject matter by creating computational artifacts. There is a growing consensus that CT is a collective problem-solving skill that will benefit every student [30] and an all-embracing conceptual foundation that includes engineering, mathematics, and design thinking [31]. Scratch, which is one of the many tools designed to teach kids to code, provides students with an opportunity to express their creativity through stories, games, and animations [28].

While Scratch is widely used, we know little about how it influences students' creative thinking [28]. Thus, the Codecraft competition provided a platform for students to express their creativity, go beyond the low floor, immerse into the wide walls, and pave their way towards the high ceiling. Codecraft competition, with the use of Scratch 2.0 was thus used to offer students challenges in problem-solving combined with coding. The emphasis is on how students may learn to create software by thinking critically to solve a problem using Scratch codes. This can potentially enhance the competency and computational skills of students to adhere to better learning outcomes in higher education and the world of work.

Through the Codecraft competition, it was expected that the following improvements would occur in the learners:

i. enhancing creativity in a hands-on approach in the thick of the action experimenter and risk-taker-designer, editor and assembler;
ii. (switch from reading to writing digital programs through Scratch;
iii. improving engagement and motivation;
iv. enhancing problem-solving skills;
v. learning a subject by building an educational program about it;
vi. instilling persistence and resilience in learners;
vii. tolerating ambiguity;
viii. modelling critical thinking and behaviour;
ix. encouraging curiosity;
x. adopting an inclusive pedagogy in which teachers and learners enter a co-participating process around activities and explorations;
xi. asking questions and identifying problems and issues together with debating and discussing their thinking.

The coding competitions were scheduled over two phases. The first phase was a preliminary round where teams from colleges around the island were invited to demonstrate their coding skills through the completion of various tasks. The second phase, known as 'the finals', involved thirty best teams selected from the preliminary phase. Each team consisted of three students of grade nine and one educator who acted as a mentor. A maximum of two teams per school was allowed to register online. The teams had to bring their laptops with pre-installed offline Scratch program. During the preliminary phase, the teams were given a questionnaire in a CD asking them to execute a series of instructions using scratch 2.0. To motivate the participants, prizes were awarded as follows:

1st prize: MUR 45, 000 2nd prize: MUR 30, 000 3rd prize: MUR 21, 000

Furthermore, prizes of MUR 5,000, MUR 4,000 and MUR 3,000 respectively were proposed to the mentors of the winning teams.

The judges were from the University of Mauritius and University of Technology, Mauritius. The first edition of Codecraft was held in 2016 and required learners to develop an interactive storytelling project. The second edition held in 2017 focused on gamification of learning. Indeed, learners had to code a high-quality educational game. The 3rd edition in 2018 was to develop an interactive memory journey based on the 50th independence anniversary of Mauritius. Teams in the final were expected to construct a digital timeline after independence on a given theme. For the 2018 edition, the competition organised by the Mauritius Institute of Education benefitted from the support and collaboration of many partners such as the National Archives, the National Library, the Mahatma Gandhi Institute, and the Mauritius Broadcasting Corporation. The number of participants increased over the years from 105 in 2016 to 124 in 2017 and 135 in 2018 (Table 1).

Table 1: Design of the competition for the 2016, 2017, and 2018 editions

Year	**2016**	**2017**	**2018**
Themes	Interactive digital storytelling	Gamification of learning	Interactive memory journey on 50th independence anniversary of Mauritius
Number of participating teams	105	124	135

1.3 Mentor's Workshop and Training of Students

Educators are often concerned with their lack of ability and confidence [32] and feel isolated because they have to train themselves and find appropriate resources for students [33]. Therefore, mentor's trainings were conducted in which basic programming concepts and the Scratch environment. Using an active learning approach, an exemplar puzzle game was presented systematically and the mentors immediately applied them by creating the programs using Scratch. After the initial familiarization, mentors began training the students who were enlisted to participate in the competition. In order to give ideas, several examples of Scratch video tutorials, exemplar materials, and systematic guides were posted on the competition's Facebook page.These tutorials allow participants to develop an understanding of the learning content without external assistance. The difficulty progression successfully scaffolds abstraction in a way that validates its use and enables participants to transfer those skills in their Scratch projects.

1.4 Aim of the Study

This research sought to gain insights into how participation in the competition enhanced the 21st-century skills (collaboration, communication, critical thinking, and creativity) of the students. The study also examined the structure and quality of the projects created by the participants. The research questions were:

- To what extent did the competition enhance the 21st-century skills of the students?
- Which computational thinking concepts did the participants develop most?
- What was the overall quality of the programs created by the participants?

II. Material and Methods

Thesemi-structured interviews were carried out from July to October 2019 with randomly selected 30 educators (mentors) and 30 students who previously participated in the contests to find out their views about the competition. More specifically, the interview questions focused on the impact of the competition on the computational thinking skills and the 21st-century skills (collaboration, communication, critical thinking, and creativity) of the participants. Furthermore, 85 projects submitted during the final round of the competition were scrutinized to study the ways in which students used the Scratch program. Finally, the levels of computational thinking of the 85 projects were analysed by a web-based tool known as Dr Scratch,which is a very effective evaluation tool [34].

2.1 Analysis of Scratch projects with Dr Scratch web tool

Assessing the development of computational thinking is not a trivial issue, and several authors, like Resnick and Brennan, have proposed different approaches and frameworks to try to address the evaluation of this competence [18, 35]. New tools are being developed to support teachers in the assessment of the CT of students. One of the most relevant tools is Hairball [36], a static code analyser for Scratch projects that detects programming errors in the scripts of the projects. Dr. Scratch [37, 38] is a free/open-source web tool powered by Hairball. It analyses Scratch projects and automatically assign computational thinking (CT) score according to the level of competence shown by the creator on the following seven concepts: abstraction and problem decomposition, parallelism, logical thinking, synchronization, algorithmic notions of flow control, user interactivity, and data representation. To evaluate the level of development on each of these concepts, Dr Scratch implements an algorithm based on the rules in Table 2.

Table 2: Areas analyzed by Dr. Scratch

Area	**Score definition**
Abstraction and Problem Decomposition	Use of commands to create functions and use of sprites in an advanced way.
Parallelism	Use of 2 or more commands that execute at the same time.
Logical Thinking	Use of "if then else" commands.
Synchronization	Use of commands that wait or send messages to synchronize actions.
Flow Control	Use of "for", "while" commands.
User Interactivity	Use of commands that do some kind of interaction with the user; e.g. mouse use, keystroke.
Data Representation	Use of commands that modify actor, variable, and lists properties.

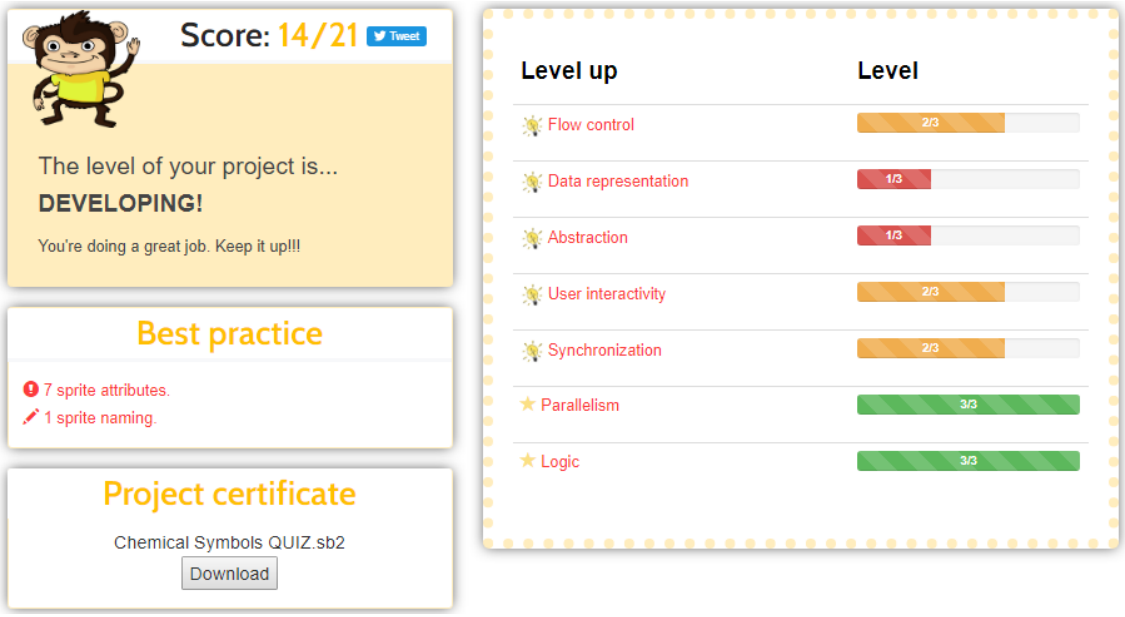
The overall CT score is obtained by adding up the scores of each CT concept. Projects with up to 7 points are given a *Basic* CT, while projects between 8 and 14 points are considered as *Developing*, and projects with more than 15 points are evaluated as *Proficient* (Table 3).

Table 3: Level of development for each computational thinking (CT) concept

CT Concept	**Basic**	**Developing**	**Proficiency**
Abstraction and problem decomposition	More than one script and more than one sprite	Definition of blocks	Use of clones
Parallelism	Two scripts on greenflag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is >%s, two scripts on when backdrop change to
Logical thinking	If	If else	Logic operations
Synchronization	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow control	Sequence of block	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

Dr. Scratch allows users to analyze Scratch projects by either uploading a .sb2 or just introducing the URL of the project. Figure 2 shows the output of the Dr. Scratch tool after performing the analysis on a Scratch project in the

Codecraft final 2018. An overall CT score point is computed, informing the user about the partial count of the different CT concepts.



Score: 14/21 Tweet

The level of your project is...

DEVELOPING!

You're doing a great job. Keep it up!!!

Best practice

7 sprite attributes.

1 sprite naming.

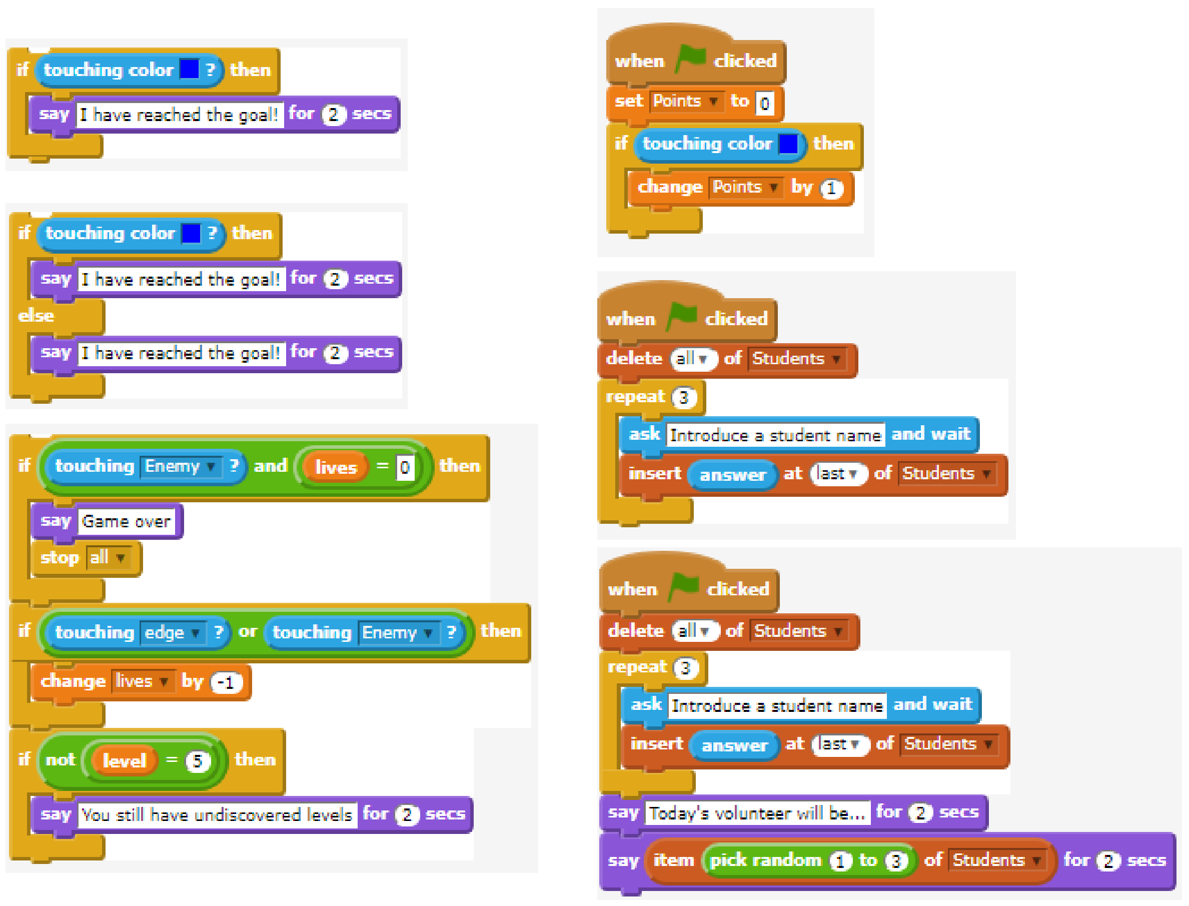
Project certificate

Chemical Symbols QUIZ.sb2

Download

Level up	Level
Flow control	2/3
Data representation	1/3
Abstraction	1/3
User interactivity	2/3
Synchronization	2/3
Parallelism	3/3
Logic	3/3

Figure 2: Dr. Scratch shows the computational thinking (CT) Score after analysing a Scratch project



```
if touching color ? then
    say I have reached the goal! for 2 secs

if touching color ? then
    say I have reached the goal! for 2 secs
else
    say I have reached the goal! for 2 secs

if touching Enemy ? and lives = 0 then
    say Game over
    stop all
if touching edge ? or touching Enemy ? then
    change lives by -1
if not level = 5 then
    say You still have undiscovered levels for 2 secs
```

```
when clicked
set Points to 0
if touching color then
    change Points by 1

when clicked
delete all of Students
repeat 3
    ask Introduce a student name and wait
    insert answer at last of Students

when clicked
delete all of Students
repeat 3
    ask Introduce a student name and wait
    insert answer at last of Students
say Today's volunteer will be... for 2 secs
say item pick random 1 to 3 of Students for 2 secs
```

Figure 3 (a). Different levels of development of data representation: basic (top), developing (center) and proficient (bottom)

Figure 3 (b). Different levels of development of logical thinking: basic (top), developing (center) and proficient (bottom)

III. Results and Discussion

3.1 Results of Semi-Structured İnterviews

The results of the semi-structured provided a better understanding of how the participating educators and students responded to the opportunities and challenges posed by the contest

Comments from educators
Instilled computational thinking skills among participants.
Bridge communication between educators and the Mauritius Institute of Education.
Videos on Scratch programming were very helpful.
You already have a good and dedicated team at the Mauritius Institute of Education.
Competition for different levels of students desirable - Grade 7 to Grade 13, with different levels of challenges.
More teams should participate per school. Teachers should get more training facilities, not just one day.
Keep the competition on-going. Such initiatives only do well to the teaching and learning communities.
I wish to thank the organizing team. They are excellent. Well done.
Codecraft should be opened to other grades as well.

The educators interviewed appreciated the training programs given to them and revealed that they trickled the knowledge gained to their students. They confirmed that the contest boosted collaboration and team spirit of participants. Furthermore, they suggested that the Codecraft competition should be extended to students of other grades also and with incrementally different levels of challenges. They also appreciated that the contest developed several skills of the students including computational thinking and creativity. These views were parallel with the study of Pinto & Escudeiro [39], which revealed that the use of Scratch motivates students and enhances the learning process. It also improves the level of concentration and encourages collaborative learning. Furthermore, Nouri et al., [40] stated that computational thinking, digital competence, and 21st-century skills of students were enhanced when they learnt programming.

Comments from students
Our computational thinking skills were enhanced.
We will definitely recommend our friends to participate next year.
We were highly motivated and had a lot of fun.
Fostered collaboration among us.
We were highly enthusiastic and committed to do our best.
The competition challenged our creativity skills.
The skills acquired during the contest are helping us in programming and robotics.

The interview results showed that students were highly enthusiastic to participate in the contest, which boosted the development of their computational thinking skills. The urge they showed to have more training is proof enough that the challenge is an appealing one and they are strongly willing to go forth and train their students for such events. They also admitted that the hands-on project work during the competition enhanced their critical thinking and creativity skills. They disclosed that they will recommend their friends to participate in the forthcoming competition and would be glad to guide them as they have gained a lot of experience after participating in the contest. The increase in the number of participants over the years from 105 in 2016 to 124 in 2017 and 135 in 2018 confirmed that the contest is gaining in popularity in the education sector. Our findings were congruent with the study of Nouri et al., [40] which highlighted that teaching programming to k-9 students not only develops CT skills but alsopromotes skills and attitudes of a more general character that are strongly associated with 21st-century skills and digital competence/literacy. Moreover, Turchi et al., [5] argued that embracing a playful experience in programming could support pupils' progress in STEM subjects by learning CT skills while enjoying the experience.

3.2 Analysis of Projects with Dr Scratch

Snapshots of some of the projects developed by the students are presented in Figure 4

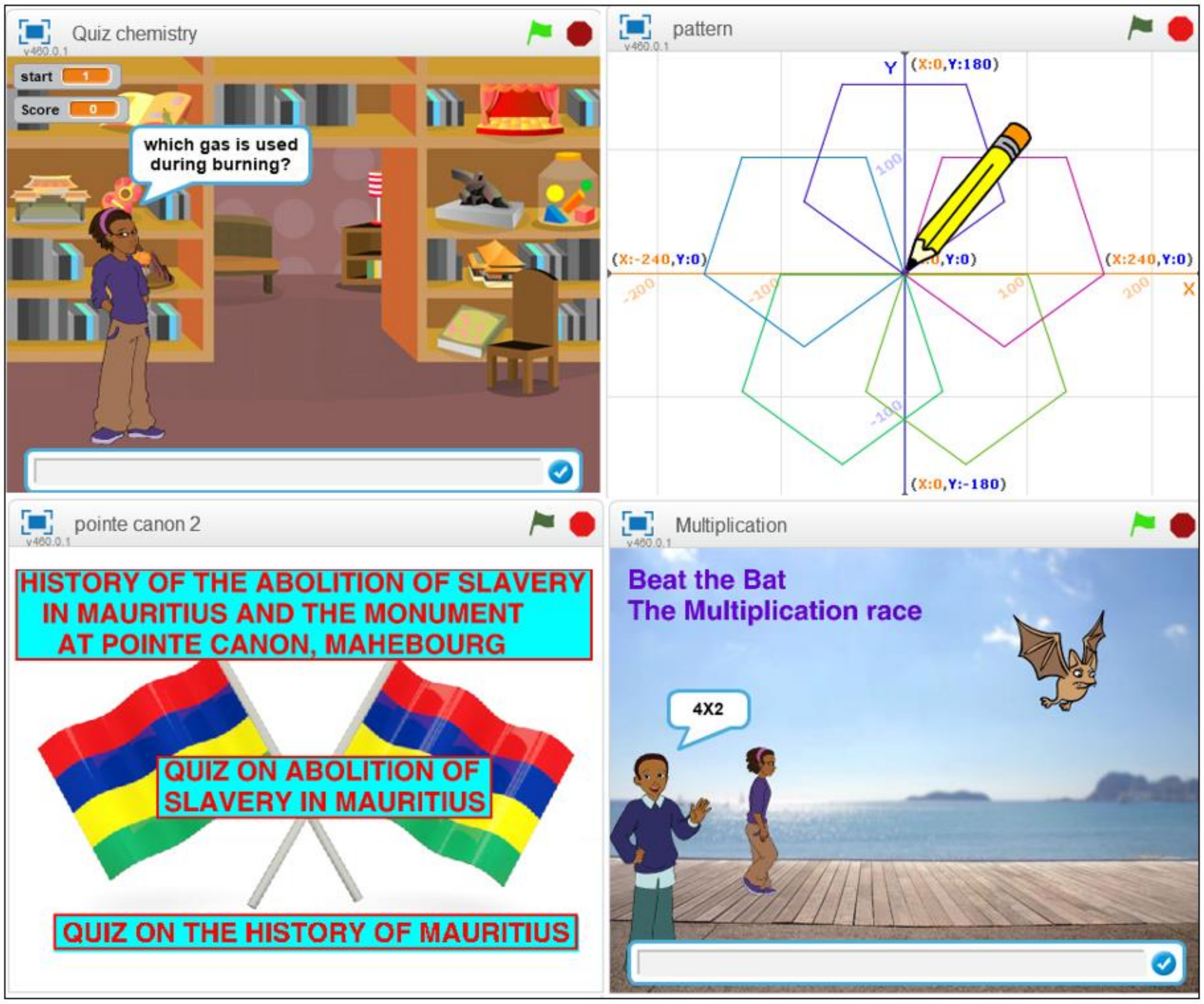

Quiz chemistry
start 1
Score 0
which gas is used during burning?
pattern
Y (X:0,Y:180)
(X:-240,Y:0)
(X:240,Y:0)
X
(X:0,Y:-180)
pointe canon 2
HISTORY OF THE ABOLITION OF SLAVERY IN MAURITIUS AND THE MONUMENT AT POINTE CANON, MAHEBOURG
QUIZ ON ABOLITION OF SLAVERY IN MAURITIUS
QUIZ ON THE HISTORY OF MAURITIUS
Multiplication
Beat the Bat
The Multiplication race
4X2

Figure 4: Snapshots of some projects submitted during the Codecraft competition

The CT score of the projects submitted in the preliminaries and finals of the Codecraft competition editions 2016, 2017, and 2018 were analysed using Dr Scratch. The average CT score increased from 11.8 in 2016, to 14.2 in 2017 and 16.0 in 2018 (Table 4). As can be seen in Figures 5-8, which presents the mean score for each of the CT components throughout the years, the concepts in which higher scores were obtained are flow control, abstraction, parallelism, and synchronization, while user interactivity and data representation got slightly lower scores. A study by Park & Shin [41] also pointed out that students using Scratch projects scored higher on average on parallelism, synchronization, and flow control.

Table 4: Average CT score of projects

Level of project (max score 21)	Year 2016 (N=29)		Year 2017 (N=27)		Year 2018 (N=29)	
	n (%)	Mean Score	n (%)	Mean Score	n (%)	Mean Score
Basic (0 to 7)	3 (10.34)	11.83	0 (0.00)	14.19	0 (0.00)	16.00
Developing (8 to 14)	24 (82.76)		15 (55.56)		5 (17.24)	
Master (15 to 21)	2 (6.90)		12 (44.44)		24 (82.76)	

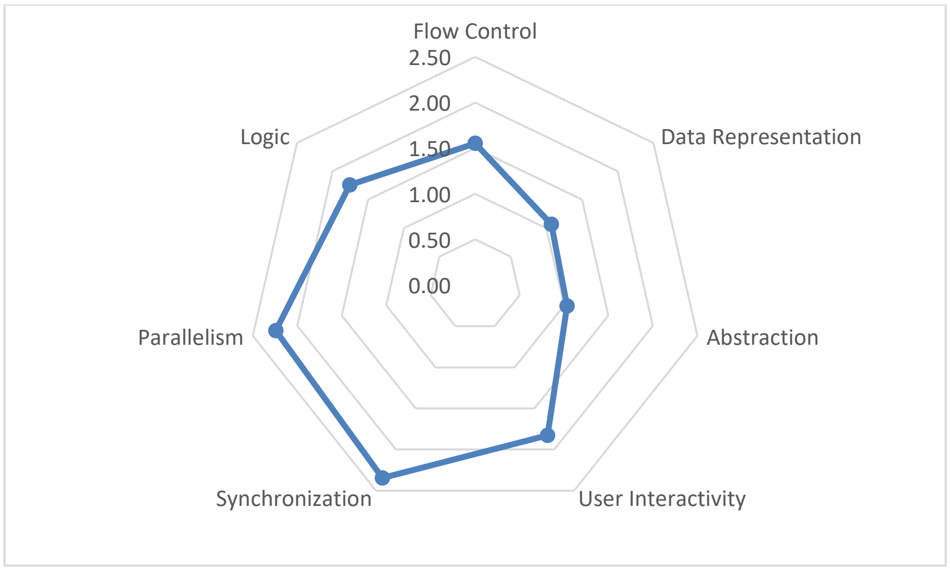


Figure 5: Mean score of constructs for 2016

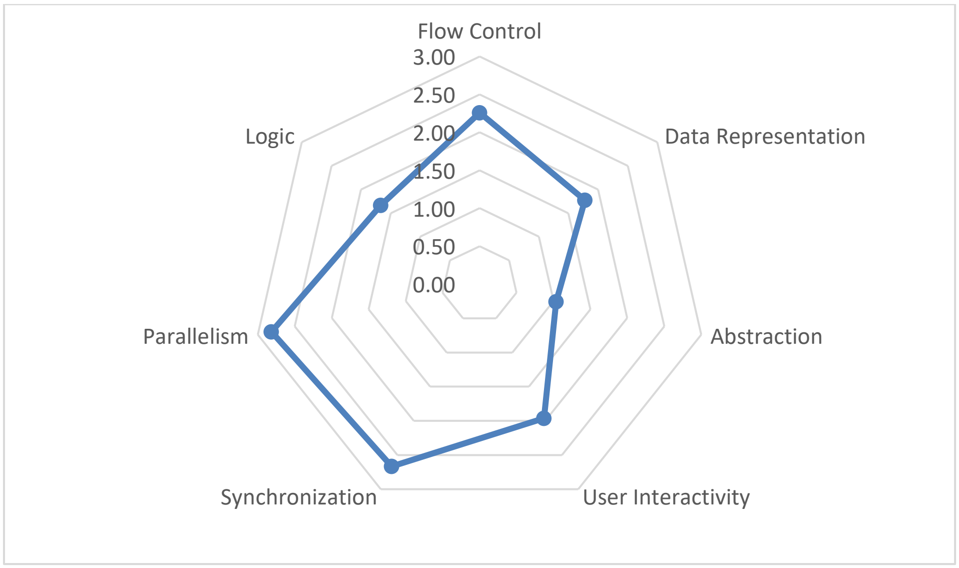


Figure 6: Mean score of constructs for 2017

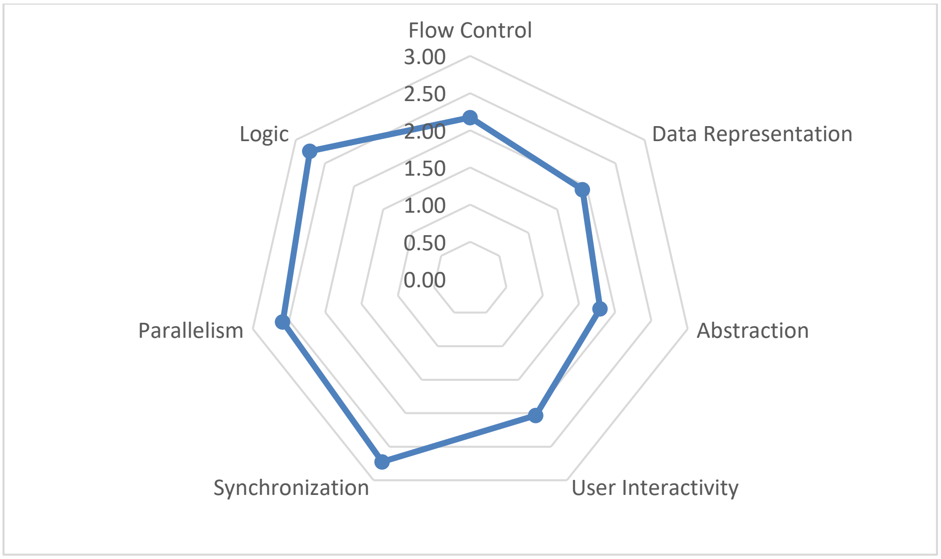


Figure 7: Mean score of constructs for 2018

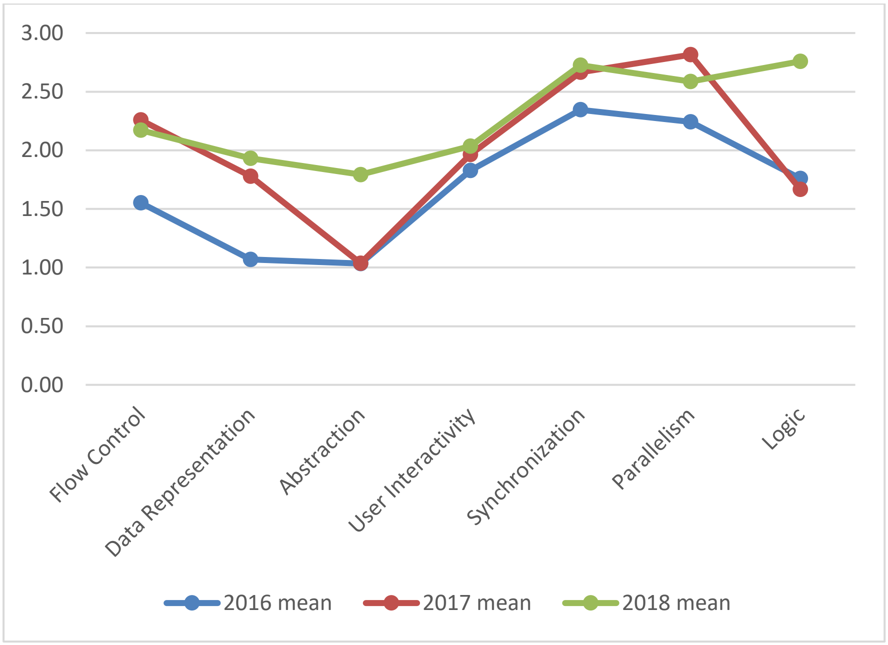


Figure 8: Mean score of constructs for 2016, 2017, and 2018

The mean scores of the CT concepts abstraction considerably from 2016 to 2018 (Figure 8).

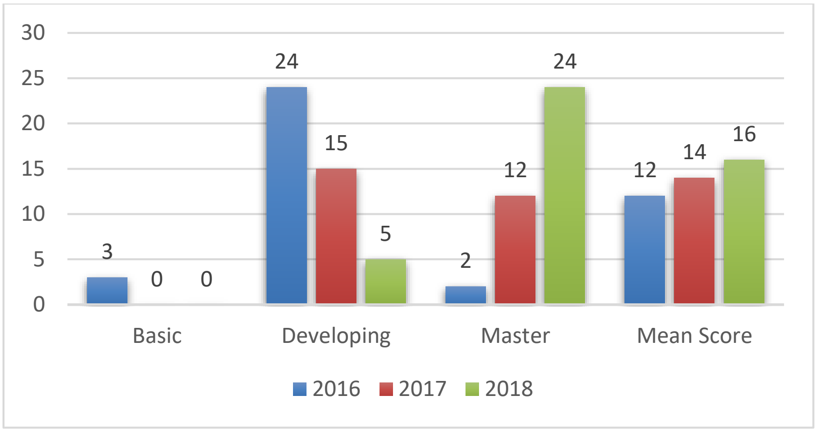


Figure 9: Progression of the Project Level and mean scores for 2016, 2017, and 2018

Analysis of the results revealed that the number of teams scoring the Master level increased significantly from 2 in 2016, to 12 in 2017 and 24 in 2018 (Figure 9). It indicated that the mentors gained experience through the various workshops they attended and successfully passed on the knowledge to their students. This support neo-Piagetian theory, which advocates that people, regardless of their age, progress through progressively abstract forms of reasoning as they gain expertise in a domain [42]. The experience acquired by contestants was also trickled down to the next batches indicating the enthusiasm and team-building spirit created by the competition. The results echoed the findings of Du et al., [26] which indicated that tutorials called 'Hour of Code' had a positive impact on students' attitudes toward programming and also improved their computing skills. Our findings were also in line with Bai et al., [3] and Baratè et al., [43] claiming that the use of gamification foster students' CT skills, engagement, and motivation in addition to their academic success at early ages.

IV. Conclusion

The study demonstrated that using contest-based learning as an interdisciplinary activity has the potential to enhance computational thinking, modeling, reasoning, and problem-solving skills among learners. Analysis of the Scratch projects showed that the average CT score of the Scratch projects increased from 11.8 in 2016, 14.2 in 2017, and 16.0 in 2018. The CT concepts in which students had higher scores were flow control, abstraction, parallelism, and synchronization, while user interactivity and data representation got slightly lower scores. Interviewing of the students revealed that the contest highly fostered their computational thinking, collaboration, critical thinking, and creativity skills. The educators appreciated that through the Codecraft competition, the Mauritius Institute of Education has adopted a brilliant approach to instill computational thinking and 21st-century skills among learners. For instance, the computational skills of participants grew significantly each passing year. The contest was gaining popularity in the education sector as the number of participating teams increased over the years. The results of the study revealed that the competition would have a greater impact on the education system if it is opened to all grades and with incrementally different levels of challenges.

Implications for practice or policy

- Computational thinking and 21st-century skills of students can be enhanced through participation in coding contests.
- Policymakers need to consider sponsoring coding contests such as Codecraft to make the competition even more attractive.
- The competition needs to be organized for students of all grades with incrementally higher levels of challenges.

Acknowledgement

The Codecraft competitions were sponsored by the Rotary Club of Port Louis, Mauritius.

References

[1]. Webster, C., & Ivanov, S. (2019). *Robotics, artificial intelligence, and the evolving nature of work*. In George, B., & Paul, J. (Eds.). Business Transformation in Data Driven Societies, Palgrave-MacMillan.

[2]. Mishra, P., & Yadav, A. (2013). Of art and algorithm: rethinking technology &creativity in the 21st century. *TechTrends, 57*(3), 10–14.https://doi.org/10.1007/s11528-013-0655-z

[3]. Bai, H., Pan, W., Hirumi, A., & Kebritchi, M. (2012). Assessing the effectiveness of a 3-D instructional game on improving mathematics achievement and motivation of middle school students. *British Journal of Educational Technology*, *43*(6), 993-1003.

[4]. Pinto, A. and Escudeiro, P. (2014) "*The use of scratch for the development of 21st century learning skills in ICT*", 9th Iberian Conference on Information Systems and Technologies (CISTI), IEEE. doi: 10.1109/CISTI.2014.6877061

[5]. Turchi, T., Fogli, D. & Malizia, A. (2017). Fostering computational thinking through collaborative game-based learning. *Multimedia Tools & Applications,78*,13649–13673.https://doi.org/10.1007/s11042-019-7229-9

[6]. Prensky, M. (2001). Digital natives, digital immigrants Part 1. *On the Horizon*, *9*(5), 1-6. http://dx.doi.org/10.1108/10748120110424816

[7]. Blotnick, E. (2017). *Digital Natives, mastering our world*. Firefall media, California, USA.

[8]. Seehorn, D., Carey, S., Fuschetto,B., Lee, I., Moix, D., O'Grady-Cunniff, D., Boucher Owens, B., Stephenson, C. & Verno, A. (2011). *CSTA K--12 Computer Science Standards*. ACM Digital Library Retrieved from http://scratch.ttu.ee/failid/CSTA_K-12_CSS.pdf

[9]. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35. https://doi.org/10.1145/1118178.1118215

[10]. Wing, J.M. (2011). Research Notebook: Computational Thinking - What and Why. Retrieved from http://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

[11]. Šerbec, I. N., Š Cerar., & A Žerovnik. (2018). Developing computational thinking through games in Scratch. In *Education and Research in the Information Society*, pp. 21-30.

[12]. European Schoolnet. (2014). *Computing our future. Computer programming and coding – priorities, school curricula and initiatives across Europe*. European Schoolnet, Brussels, Belgium

[13]. Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, New York.

[14]. Papert, S., & Harel, I. (1991). *Situating constructionism*. In S. Papert & I. Harel (Eds.), Constructionism. New York: Ablex Publishing.

[15]. Sabelli, N. (2008). *Constructionism: A New Opportunity for Elementary Science Education*. DRL Division of Research on Learning in Formal and Informal Settings.

[16]. Baist, A., & Pamungkas, A. S. (2017). Analysis of Student Difficulties in Computer Programming. *Volt Jurnal Ilmiah Pendidikan Elektro. 2*(2), 81-92.

[17]. Guzdial, M. (2004). *Programming environments for novices*. In Computer Science Education Research, Fincher, S. & Petre, M. eds. Taylor & Francis, Abingdon, U.K., 127–154.

[18]. Resnick, M. (2007) Sowing the seeds for a more creative society. *Learning and Leading with Technology*, *35*(4), 18–22. Retrieved from https://web.media.mit.edu/~mres/papers/Learning-Leading-final.pdf

[19]. Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The *Scratch language and environment. ACM Transactions on Computing Education*, *10*(4), 16.http://dx.doi.org/10.1145/1868358.1868363

[20]. Malan, D. J. and Leitner, H. H. (2007). *Scratch for budding computer scientists*. In Proceedings of the 38th SIGCSE technical symposium on Computer science education, 223-227, Covington, Kentucky, USA.

[21]. Monroy-Hernández, A., & Resnick, M. (2008). FEATURE empowering kids to create and share programmable media. *Interactions*, *15*(2), 50-53. http://doi.acm.org/10.1145/1340961.1340974

[22]. Scratch website. (2020). *Community statistics at a glance*. Retrieved from https://scratch.mit.edu/statistics/

[23]. Roque R., Rusk N., Resnick M. (2016) Supporting Diverse and Creative Collaboration in the Scratch Online Community. In: Cress U., Moskaliuk J., Jeong H. (eds) *Mass Collaboration and Education. Computer-Supported Collaborative Learning Series*, vol. 16. Springer, Cham.https://doi.org/10.1007/978-3-319-13536-6_12

[24]. Von Wangenheim, C. G., Alves, N. C., Rodrigues, P. E., & Hauck, J. C. (2017). Teaching Computing in a Multidisciplinary Way in Social Studies Classes in School--A Case Study. *International Journal of Computer Science Education in Schools*, *1*(2), 1-14. https://doi.org/10.21585/ijcses.v1i2.9

[25]. Rusk, N., Resnick, M. & Maloney, J. (2006). *Scratch and 21st Century Skills*. MIT Media Lab. US: Lifelong Kindergarten Group.Du, J.;Wimmer, H.; Rada, R. (2016). Hour of Code: Can it change students' attitudes toward programming? *Journal of Information Technology Education: Innovations in Practice*, 15, 52-73.

[26]. Resnick, M., Maloney, J., Monroy Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). *Scratch: Programming for all. Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

[27]. Yadav, A. & Cooper, S. (2017). Fostering creativity through computing. *Communications of the ACM*, *60* (2), 31-33.https://doi.org/10.1145/3029595

[28]. Peppler, K. & Kafai, Y. (2007). From Supergoo to Scratch: exploring creative media production in informal learning. *Journal on Learning, Media, and Technology*, *32*(7), 149–166. https://doi.org/10.1080/17439880701343337

[29]. Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*,*25*(2), 199–237. https://doi.org/10.1080/08993408.2015.1033142

[30]. Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking.*Educational Research Review*, *22*, 142–158. http://dx.doi.org/10.1016/j.edurev.2017.09.003

[31]. Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E., Belikov, O. M., Shoop, T. (2019). Coding in K-8: international trends in teaching elementary/primary computing. *TechTrends*, *63*(3), 311–329. https://doi.org/10.1007/s11528-018-0295-4

[32]. Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235–254. https://doi.org/10.1080/08993408.2016.1257418

[33]. Moreno-León, J., Robles, G., & Román-González, M. (2020). Towards data-driven learning paths to develop computational thinking with Scratch. *IEEE Transactions on Emerging, Topics in Computing*, *8*(*1*), 193-205. doi:10.1109/TETC.2017.2734818

[34]. Brennan, K. A. (2012). *New frameworks for studying and assessing the development of computational thinking.* Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.

[35]. Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). *Hairball: lint inspired static analysis of scratch projects*. In Proceedings of the 44th ACM Technical Symposium on Computer Science Education (pp. 215–220). doi: 10.1145/2445196.2445265

[36]. Moreno, J. (2014). Automatic detection of bad programming habits in scratch: a preliminary study.*Frontiers in Education Conference (FIE), IEEE*, 1-4. http://dx.doi.org/10.1109/FIE.2014.7044055

[37]. Moreno, J., & Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate scratch projects. *Proceedings of the Workshop in Primary and Secondary Computing Education on ZZZ - WiPSCE '15*, 132–133. https://doi.org/10.1145/2818314.2818338

[38]. Pinto, A., & Escudeiro, P. (2014). The use of Scratch for the development of 21st century learning skills in ICT. *9th Iberian Conference on Information Systems and Technologies* (CISTI), Barcelona.

[39]. Nouri, J., Zhang, L., Mannila L. & Norén, E. (2020): Development of computational thinking, digital competence and 21st century skills when learning programming in K-9, *Education Inquiry*. *11*(1), 1-17. https://doi.org/10.1080/20004508.2019.1627844

[40]. Park, Y. & Shin, Y. (2019). Comparing the Effectiveness of Scratch and App Inventor with Regard to Learning Computational Thinking Concepts. *Electronics*, *8(11)*, 2-12. https://doi.org/10.3390/electronics8111269

[41]. Lister, R. (2011). *Concrete and other neo-Piagetian forms of reasoning in the novice programmer*. In Proceedings of the Thirteenth Australasian Computing EducationConference (pp. 9–18).

[42]. Baratè, A., Ludovico, L. A., & Malchiodi, D. (2017). Fostering computational thinking in primary school through a LEGO®-based music notation. *Procedia computer science*, *112*, 1334-1344