Comparison of the learning curve and adaptive behavior from kids to adults using computational thinking with Block-Programming to Technology Enhanced Learning

1st Felipe Moreno-Vera
School of Computer Science
Universidad Nacional de Ingeniería
Lima, Perú
felipe.moreno.v@uni.pe

2nd Leonardo León-Vera
School of Computer Science
Universidad Nacional de Ingeniería
Lima, Perú
lleonv@uni.pe

3rd Juan Moreno-Motta
School of Informatics and System Engineering
Universidad Nacional Mayor de San Marcos
Lima, Perú
juan.moreno2@unmsm.edu.pe

4th Juan Guizado-Vasquez
School of Physics Engineering
Universidad Nacional de Ingeniería
Lima, Perú
jd.guizado.v@uni.pe

5th Michael Vera-Panez
School of Physics Engineering
Universidad Nacional de Ingeniería
Lima, Perú
mvera@uni.pe

***Abstract*—In this article we present a study on how much the learning speed differs and how much information retention capacity children between 6 to 9 years old, adolescents between 10 to 13 years old, young people between 14 to 17 years old and adults from 18 years old Who have in the same conditions of learning, same environment, same classes, same tools and the same methodology, such as mental maps, computational thinking and scripting language questions with the aim of learning concepts about algorithms, objects and classes applied to create robots, IoT concepts and Electronic hardware concepts through different online platforms using Block Programming. In addition, the way in which the learning curve is measured is evaluating the ability to retain, skill gained and how much they have learned during the course of a certain time, also the methodology that they use to solve problems.**

***Index Terms*—Learning, Education, Robotics, Kids, Teenagers, Adults, Block, Programing languages, Computer games, Block programming, Technology, Computational Thinking.**

I. Introduction

A. Present context

In the context of Peru, the education methodology is not enough to a complete a certain level of knowledge. Our target study subjects are people between the ages of 9 and 45, because is the 57.232% of the total population (information was collected from [2]). Kids (6-9 years old) have more easily way to learn and play with new tools [3], Teens (10-12 years old) and Juniors (13-17 years old) have the same capacity to learn, but they have another distractions that causes a little reduction in the learning curve for new things [4].

Adults (from 18 years old and on) have a different way to learn things about technology, some of them have experience working with computers but others do not use computers in daily life. We have approximately 100 children, 100 teens, 100 junior and 100 adults, each group is divided into a group of 25 people with the same curriculum, the same materials, the same teachers and the same 16 lessons (each lasts 3 hours a day), based on our previous work [1] we continue and complete our research.

978-1-7281-1666-2/19/$31.00 2019 IEEE

B. Description of the current situation of education and accessibility to technology

Currently, in Per we have a lot of devices distributed in all families, adults, teens and kids. Is very common see a kid with a tablet or mobile phone at 8 years old playing games or using social networks (with parental control like facebook) or watching vides about youtubers (gamers with minecraft channels, fornite, etc).

In this context, we have technology in our hands every day any time, but in education until this time, schools separate technology for the classrooms, schools don't use technology to teach and don't tech how to use technology responsibly. So, this create a disorder in the generation gap between people who use technology only for play games and those who do not use technology to improve techniques or methodologies to teach or learn new things.

In other words we have poblation that explote technology in any another aspect except in education and our consequence is that schools don't teach programming at early ages, our young people learn computer concepts or how to programming in the university at age between 20 and 22 years old. We try to break the gap through free courses of computational thinking and algorithm design concepts hidden in courses called "Robotics

for everyone" or "game development for everyone" using different methodologies describe in the present work, preparing people of all ages to a better understand of computer concepts and improve computer skills.

C. Research objective and motivation

For this situation with help of our universities we organize a course that involves learning computer science concepts to develop simple programs using C and python language and build robots with Arduino board.

For that, we measure how fast is the adaptive behavior of our students and how fast increase the understanding of these concepts. We hide the concept of Computational Thinking inside of "how to teach computer science using block programming" because computational thinking is the way that any person can interprete the world in a computer and how can extrapolate these concepts to real world. So while we teach about how interpret computer science concepts, we teach how is computational thinking works in our lives [5].

II. Definitions and techniques

We need to understand important concepts those we use in our research, those concepts are very useful when we try to measure and expose about the progress or difficulties of the students while they are learning new things with enhanced methods through technology. We use *Learning Curve* to define how they understand new concepts and *Adaptive Behavior* to define performance and attitude against new concepts.

A. Learning Curve

We introduce the concept of *Learning Curve* as the representation how to increases the learning based on experience. also, measuring if the student solve problems better than before times.

B. Adaptive Behavior

We introduce the concept of *Adaptive Behavior* as how students accept or reject new concepts. Also, we know that adults, kids and teens have different ways to learn and understand things. The adaptative behavior is notorious when in the learning process they can use examples based on computer science concepts or make jokes with computers or with some new concepts than they never used before.

C. Script Language

Script Language is a technique used to explain concepts in an very short time with simple examples, script languages are defined as a normal conversation with specific questions, the answers could be any but in all of these answers, there is a concept hidden inside.

D. Metaphors

Metaphors is used to complete the understanding of the concepts givins another examples based on the previous one, metaphors is very commonly used in the daily activity, in any situation because is the most easy way to explain new things. we use metaphors to exaplin computer science concepts with common examples.

E. Mind maps

Mind maps is a technique based in drawing a map based on a brain storming with connected ideas like a graph [6]. We use Mind maps to teach about mathematics concepts, programming concepts and algorithm design concepts [7].

F. Block Programming

Block Programming is a technique to encapsule code programming in a simple sentence and the way to programming is just joining the blocks following a flow diagram or main idea. In this research we try to test how efficient is teach computer science concepts and examples using block programming in a first steps and how fast our students can improve skills programming to jump from blocks to code in simple programs like programming sensors or programming games, in this work we encourage to our students and our readers to practice to thinking in blocks [8] and think how easy could be the learning for future generations topics like IoT or Data Science or Machine Learning with blocks, that research field or develop is part of challenges to learn and understand block programming and how you can add new libraries into a sequence of blocks [9].

III. Methodology

We started the classes with the empirical methodology based on Metaphors [10] because or students are novices in the five stage of learning programming skills: novice, advanced beginner, competence, proficiency, and expert [11]. That means, we teach using example of different simple situations in the real life to explain what can we do to interact and which solutions we provide to solve daily problems.

For difference age we use different methodology, as we mentioned above, there is a different way of understanding and retention of information in all of these ages.

A. Learning Computer Science Concepts

To introduce computer science concepts, we starts with the main question "what is an algorithm ?". To answer that question, we use first technique called script language used in kids [3] in all ages.

At first time adults and Teens understood very fast, kids and tweens take a time to understand the aim of that examples. We use different asks in different ages, simple questions to explain that an algorithm is inside in any situation and in any action.

- *Script language for kids*
  (Q): If you want to go to the bathroom, what should you need to say ? How do clean your desk ?"
- *Script language for tweens*
  (Q): Have you ever play a game in your computer ?, do you hear about PSP or XBOX ?, Do you know what Mario Kart is it?".
- *Script language for teens*
  (Q): Do you know how to solve a linear equation ?, Which rules do you need to follow when you play foot ball ?, Have your ever play guitar ?".
- *Script language for adults*

(T): Have you ever developed something ?, do you hear about Programming languages ?, Do you know what Software means?".

These questions look different but in the background they are very similar. In all categories, they answered "Yes!" and then explained how it works with a sequence of steps. You can see how many time it takes for them in Table I.

TABLE I
TABLE OF AVERAGE TIME TO THINK ABOUT THE SITUATION AND SOLVE.

Category	Situations	time thinking range
Kids	How to go to the bathroom	18-34 min
Tween	How turn on a computer	17-25 min
Teen	How to use pythagoras theorem	15-20 min
Adults	how to develop a software project	10-15 min

B. Improving programming skills

To introduce programming skills, we explain programing language with CodeCombat [13] with python language and BlocklyGames [14] with blocks and javascript language.

Blockly Games and Codecombat provide some tests to improve and teach about concepts like algorithms, loops, conditionals, functions and logic operations. To measure the understanding of the concepts we starts with simple samples like labyrinth in Blockly Games or programming actions in CodeCombat.

CodeCombat as Blockly games, uses methods to give direction to simulate movement of the character inside the maze level (See Figure 1).

Until the exercise of the game finish, we note that programming code was more easy than blocks for the students. That is because they use mind map technique [7] and others use script language to idenfity situations, more experienced students can implement directly.

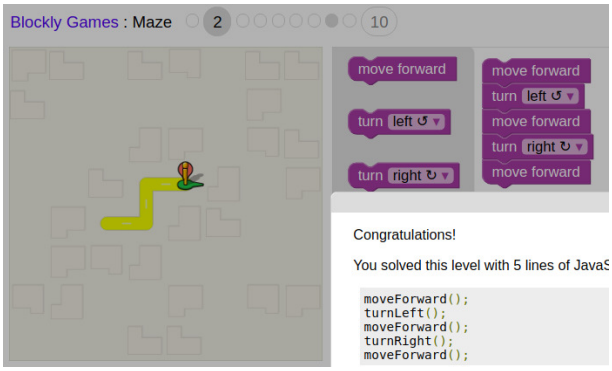
TABLE II
TABLE OF AVERAGE TIME TO DESCRIBE AND SOLVE THE LOOP PROBLEM.

Category	time thinking the problem	time to solve the problem
Kids	20-30 min	20-30 min
Tween	14-21 min	15-20 min
Teen	11-16 min	10-15 min
Adults	6-12 min	9-17 min

After loop example we can identify the best method to understand it for students was a combination of mind maps (See Table II), script language and mepathors, in other words, a combination of all methodologies is necesasary for a complete learn. In previous works, they just using the methodology of script language [12] but they detect that in some cases it was necessary more theory or more examples to exaplin better the idea but it takes more learning time.

C. Implementing a mini Robot

In this section we introduce as the goal of the course the implementation of two mini robots called Otto and Kyo (See



(a) Blockly Games

(b) Code Combat

Fig. 1. Programing kyo with TinkerCad.

Fig. 3). In this part, we use all methodologies to explain about electronic components, the mathematical fundamentals and how they should to program the sensors.

To implement and programming these robots, their movements and their sensors we use Block programming and C language (with the concepts explained above) for this part we use the online platform to simulate the component programming (See Fig. 2). To programming Otto we use the platform BitBloq [15] and to programming Kyo we use the platform Tinkercad [16].

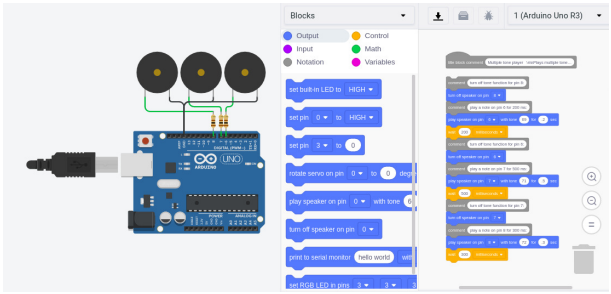
At this time, we measure how fast they understand basic electronic concepts and components description and how fast they can imagine a solution to programming using blocks. We starts with script language then with metaphors and finally with mind maps to get a solution for the problem of how should we programming these mini robots.

But in thi part we focus on programming skills and how they can translate their mind maps into a program (blocks or programming language). We note that there is a improving in programming speed after they understand the logic following in the mind map previously drawn as you can see in Table III.

TABLE III
TABLE OF AVERAGE TIME TO DESCRIBE THE MINI ROBOT IMPLEMENTATION.

Category	Time to describe the problem	Average Time to Solution
Kids	14-20 min	15 min
Tween	8-15 min	11 min
Teen	5-9 min	8 min
Adults	9-12 min	13.5 min

We need the average time to measure the learning curve and interpret that as how much the adaptive behavior is increasing.



(a) Blocks

```
Text                                   1 (Arduino Uno R3)
*/

int pos = 0;

void setup()
{
  pinMode(8, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
}

void loop()
{
  // turn off tone function for pin 8:
  noTone(8);
  // play a note on pin 6 for 200 ms:
  tone(6, 880, 200); // play tone 69 (A5 = 880 Hz)
  delay(200); // Wait for 200 millisecond(s)
  // turn off tone function for pin 6:
  noTone(6);
  // play a note on pin 7 for 500 ms:
  tone(7, 988, 500); // play tone 71 (B5 = 988 Hz)
  delay(500); // Wait for 500 millisecond(s)
  // turn off tone function for pin 7:
  noTone(7);
  // play a note on pin 8 for 300 ms:
  tone(8, 1047, 300); // play tone 72 (C6 = 1047 Hz)
  delay(300); // Wait for 300 millisecond(s)
}
```

(b) Code

Fig. 2. Blocks Programming vs Programming Code.

IV. RESULTS

a) Measuring the learning curve: As definition of learning curve, we need to verify if the average time of learning in previous works in kids [3] and teens [4] is correct in all new concepts and for computer science we need to add mathematic, physics and electronic components concepts. So we teach and refor this topics with different methodologies, using the formula below we approximate the average time to learning and we compare with the real average time get from the students.

We note that our average time in some cases is a bit less than the approximate time and also is less than the average time mentioned in previous works.

$$Y_x = Kx^{log_2(b)}$$

Where:

K: Number of hours used to understand the first unit (or task).

Y_x: Number of hours to understand the x^{th} unit.

x: Number of the unit.

b: Percent of learning.

So, we have this tables with respective approximations. We take the K value from the first part in script languages in Table I we take b from [17] and [18] and x is equal to 4, because is the 4th unit. See Table IV

TABLE IV
TABLE OF AVERAGE TIME TO DESCRIBE AND SOLVE.

How many time takes to learn a new concept with examples?			
Category	Real Time	Calculate time	Previous works Time
Kids	55-65 min	66.486 min	68.64 min
Tween	52-56 min	57.365 min	56.49 min
Teen	42-52 min	50.934	51.32 min
Adults	25-41 min	42.163	40.89 min

b) Following the adaptive behavior: As we show above, the time to solve problems is decreasing and that is because they understand more fast and they adapt himself to hear and think in computer terms with computer concepts. The best way to show this is with questionnaire before and after the course. See Table V.

TABLE V
TABLE OF PREFERENCES FOR COMPUTER SCIENCE

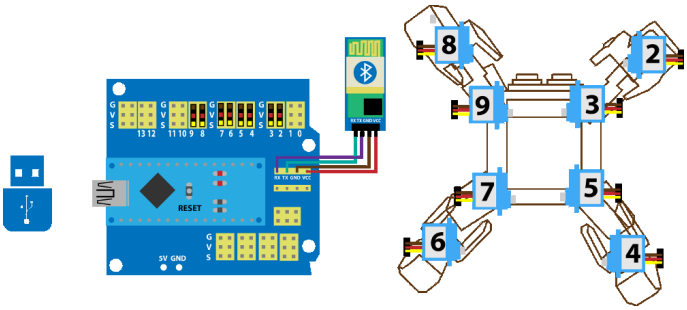
What do you thing about computer science?		
Category	Before	After
Kids	Don't know what it is	Want play with robots in schools
Tween	Just movies	Want to design and learn programming in schools
Teen	Just for game develop	Want to study informatic or game develop
Adults	Just Excel and Word	Want to develop informatic projects Using new software tools

c) Determining the best methodology per ages: To determinate which is the best methodology per ages, we analyses results above and take the questionaire as a personal preference from students, we can say that for kids the best way and their favourite method was mind maps, for tweens and teens their favourite was script languages and for adults was real life examples (metaphors). See Table VI.

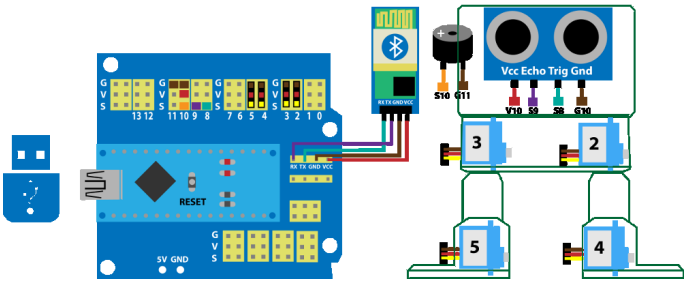
TABLE VI
TABLE OF BEST METHODOLOGY PER AGE.

Which is the best methodology per age?			
Category	Age range	methodology	Occupation
Kids	6-9	mind maps	primary school
Tween	10-12	script language	primary-secondary school
Teen	13-17	metaphors	secondary school
Adults	18 and so on	real life examples	academies, institutes universities, workers

d) Objective of the course: We implement two different mini robots following the preferences of our students to learn about mathematics and physics concepts and mechanical parts.



(a) Kyo



(b) Otto

Fig. 3. Kyo and Otto design and components.

We note that as age increases, preference for programming code increases and the same with which robot they select, kids select the most tender robot between Otto and Kyo and teens and adults select the more harder to implement See Table VII.

TABLE VII
TABLE OF PREFERENCES BETWEEN CODE VS BLOCKS.

Which do you prefer between code/blocks and Otto/Kyo robot?				
Category	Preference %	Type	Preference %	robot
Kids	92 %	Blocks	87%	Otto
Tween	50 %	Blocks/Code	60%	Otto
Teen	76 %	Code	55%	Kyo
Adults	80 %	Code	100%	Kyo

V. Conclusions

This work introduces the different adaptive behavior with different methodologies in the learning speed of ur group divided in 100 kids, 100 tweens, 100 teens and 100 adults.

We note that tweens and teens have more ability to understand new concepts using games as metaphors. We note that Adults have a strong learning speed to understand new concepts based on past experience. From kids to juniors, they present a fast learning speed, but they forget concepts in a little period of time.

Acknowledgment

This work was supported by the project PCTI-2-P-039-17 from INNOVATE-PERU, Ministerio de la Producción (PRODUCE-PERU) and Universidad Nacional de Ingeniera (UNI-PERU) for all the help and facilities given during all this program.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

References

[1] Felipe Moreno, Leonardo León, Juan Guizado, Michael Vera, A comparison of the adaptive behavior from kids to adults to learn Block Programming, CEUR Workshop Proceedings, Vol. 2193.

[2] INEI, Population index, Lima, Peru, https://www.inei.gob.pe/estadisticas/indice-tematico/population/, Last accessed 2 Feb 2018.

[3] Sajana Sigde, Technology and Learning Capacity of Children: A Positive Impact of Technology in Early Childhood, Johnson & Wales University - Providence.

[4] Amanda Lenhart, Paul Hitlin and Mary Madden, Teens and Technology, PEW INTERNET & AMERICAN LIFE PROJECT.

[5] Jeanette M. Wing, Computational Thinking Viewpoint, Communications of the ACM, volume 49,

[6] Smitha Sunil Kumaran Nair, Khadija Al Farei, A brain friendly tool to facilitate research-teaching nexus: Mind maps, 8th International Conference on Information and Communication Systems (ICICS), 2017.

[7] Shahla Gul, Muhammad Asif, Waqar Ahmad and Uzair Ahmad: Teaching Programming: A Mind Map based Methodology to Improve Learning Outcomes. In: International Conference on Information and Communication Technologies (2017). number 3, 2006.

[8] Daniel Wendel, Paul Medlock-Walton, Thinking in blocks: Implications of using abstract syntax trees as the underlying program model, IEEE Blocks and Beyond Workshop (Blocks and Beyond), 2015.

[9] David Werntrop, Uri Wilensky, The challenges of studying blocks-based programming environments, IEEE Blocks and Beyond Workshop (Blocks and Beyond), 2015.

[10] D. Prez-Marn, R. Hijn-Neira, M. Martn-Lope : A Methodology Proposal based on Metaphors to teach Programming to children, In: IEEE Revista Iberoamericana de Tecnologias del Aprendizaje (2018).

[11] A. Robins ,J. Rountree, and N.Rountre: Learning and teaching programming: A review and discussion, In: Computer Science Education (2003).

[12] D. Ginat: On novice loop boundaries and range conceptions, In: Computer Science Education (2004).

[13] CodeCombat, https://codecombat.com/play. Last accessed 2 Jan 2018.

[14] Bockly Games, https://blockly-games.appspot.com/. Last accessed 18 Jan 2018.

[15] BitBloq https://bitbloq.bq.com/. Last accessed 20 Jan 2018.

[16] Tinkercad, https://www.tinkercad.com. Last accessed 28 Jan 2018.

[17] Roman H. T., Teaching Your Kids to Think and Solve Problems. IEEE USA Books & eBooks, 2017.

[18] Josh Shipp, Raising Teens to Live with Technology Responsibly, IEEE Technology and Society Magazine, Volume 36, Issue 4, pages 42-43, 2017.