Loo/Think @JC
賽馬會運算思維教育
Inspiring digital creativity 啟發數碼創意

CTE 2020

Proceedings of International Conference on Computational Thinking Education 2020

19 - 21 August 2020

CTE 2020

Computational Thinking Education

Created and Funded by

香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
同心同步同進 RIDING HIGH TOGETHER

Co-created by

香港教育大學
The Education University of Hong Kong

Massachusetts Institute of Technology

CityU 香港城市大學 City University of Hong Kong

CoolThink@JC

Proceedings of International Conference on

Computational Thinking Education 2020

19-21 August 2020

Hong Kong

Funded and Created by

The Hong Kong Jockey Club Charities Trust

Co-created by

The Education University of Hong Kong

Massachusetts Institute of Technology

City University of Hong Kong

Publication of The Education University of Hong Kong

10 Lo Ping Road, Tai Po, New Territories, Hong Kong

ISSN 2664-035X (CD-ROM)

ISSN 2664-5661 (online)


CTE 2020

Editors

Preface

International Conference on Computational Thinking Education 2020 (CTE2020) is the fourth international conference organized by CoolThink@JC, which is created and funded by The Hong Kong Jockey Club Charities Trust, and co-created by The Education University of Hong Kong, Massachusetts Institute of Technology, and City University of Hong Kong.

CoolThink@JC strives to inspire the digital creativity among students and nurture their proactive use of technologies for social good from a young age. In collaboration with the world's leading experts and local educators, CoolThink@JC empowers teachers with high-quality teaching materials, learning platform, and professional development programmes. Since 2016, CoolThink@JC has trained more than 110 teachers from 32 pilot schools and benefited over 20,000 primary students with CoolThink classes. The CoolThink@JC approach prepares students for a fast-changing digital future through a hands-on, minds-on, and joyful learning experience. An independent evaluation has found that students participated in CoolThink@JC grew twice as much in problem-solving skills when compared with non-participating students. Following the successful implementation of the four-year pilot, the second phase of the CoolThink@JC is launched in 2020, with the aim of mainstreaming computational thinking education.

CTE2020 is held online on 19-21 August, 2020. Last year, the conference attracted over 600 worldwide scholars, educational practitioners and policymakers from 17 countries/ regions. The International Teacher Forum is first introduced this year to reach out to K-12 CT teachers. Under the pandemic, CTE2020 experienced reschedule and has switched from face-to-face to online mode. With the support from speakers, panelists, IPC Co-chairs, IPC members and paper authors, we have gone through challenges and are excited to welcome partcipants to join us at the conference to share their research and ideas.

"**Computational Thinking Education**" is the main theme of CTE2020 which aims to keep abreast of the latest development of how to facilitate students' CT abilities, and disseminate findings and outcomes on the implementation of CT development in school education. There are 16 sub-themes under CTE2020, namely:

Computational Thinking

Computational Thinking and Coding Education in K-12

Computational Thinking and Unplugged Activities in K-12

Computational Thinking and Subject Learning and Teaching in K-12

Computational Thinking and Teacher Development

Computational Thinking and IoT

Computational Thinking and STEM/STEAM Education

Computational Thinking and Data Science

Computational Thinking and Artificial Intelligence Education

Computational Thinking Development in Higher Education

Computational Thinking and Special Education Needs

Computational Thinking and Evaluation

Computational Thinking and Non-formal Learning

Computational Thinking and Psychological Studies

Computational Thinking in Educational Policy

General Submission to Computational Thinking Education

The conference received a total of 46 submissions (32 full papers, 11 short papers and 3 poster papers) by 107 authors from 19 countries/regions (see Table 1).

Table 1: Distribution of Paper Submissions for CTE2020

Country / Region	No. of Authors	Country / Region	**No. of Authors**
Australia	2	**Israel**	3
Brazil	5	**Malaysia**	4
Canada	2	**Singapore**	10
China	17	**South Korea**	8
Cyprus	2	**Spain**	3
Finland	5	**Sweden**	1
Germany	8	**Taiwan**	14
Greece	1	**The Netherlands**	4
Hong Kong	4	**United States**	11
India	3	**Total**	**107**

The International Programme Committee (IPC) is formed by 98 Members and 13 Co-chairs worldwide. Each paper with author identification anonymous was reviewed by at least three IPC Members. Related sub-theme Chairs then conducted meta-reviews and made recommendation on the acceptance of papers based on IPC Members' reviews. With the comprehensive review process, 37 accepted papers are presented (12 full papers, 17 short papers and 8 poster papers) (see Table 2) at the conference.

Table 2: Paper Presented at CTE2020

Sub-themes	Full Paper	Short Paper	Poster Paper	Total
CT	1	0	0	1
CT and Coding Education in K-12	3	2	2	7
CT and Unplugged Activities in K-12	1	1	0	2
CT and Subject Learning and Teaching in K-12	0	2	0	2
CT and Teacher Development	1	2	0	3
CT and IoT	0	1	0	1
CT and STEM/STEAM Education	1	2	1	4
CT and Artificial Intelligence Education	1	0	3	4
CT Development in Higher Education	3	2	0	5
CT and Evaluation	0	2	0	2
CT and Non-formal Learning	0	0	1	1
General Submission to CT Education	1	3	1	5
Total	**12**	**17**	**8**	**37**

On behalf of CoolThink@JC and the Conference Organizing Committee, we would like to express our gratitude towards all partners and participants for their contribution to the success and smooth operation of CTE2020.

We sincerely hope everyone enjoy and get inspired from CTE2020.

With Best Wishes,

Prof. KONG, Siu-cheung

The Education University of Hong Kong, Hong Kong

Conference Chair of CTE2020

Principal CHU, Tsz-wing

St. Hilary's Kindergarten and Primary Schools, Hong Kong

Conference Chair of CTE2020

Table of Contents

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Computational Thinking Competences in Countries from Three Different Continents in the Mirror of Students' Characteristics and School Learning

Amelie LABUSCH[1*], Birgit EICKELMANN[2]
[1,2] Paderborn University, Germany
amelie.labusch@upb.de, birgit.eickelmann@upb.de

ABSTRACT

Computational thinking (CT) aspires to be learned by everyone for active participation in society. However, differences in students' learning of computational thinking within and between educational systems and differences in competence among students differentiated by social background and gender emerge. The IEA study ICILS 2018 (*International Computer and Information Literacy Study*) addresses this issue by measuring competences in computational thinking and examining the conditions for the acquisition of competences in an international comparison. This allows in-depth analyses to answer the question to what extent differences in students' average competences in computational thinking can be explained by students' social background, their learning of computational thinking tasks at school, and their gender. For this purpose, regression analyses are carried out using data from three countries from three different continents (Republic of Korea, USA and Germany). The dependent variables are students' competences in computational thinking, their variance is to be explained by the independent variables social background, learning of computational thinking tasks at school and gender. The results show that performance differences in favor of students with socially privileged background exist in all three countries. Controlled by social background and gender, students' learning of computational thinking tasks at school shows significant negative relationships to their competences in computational thinking in the Republic of Korea and Germany. In addition, significant performance differences between girls and boys in favor of boys under control of social background and students' learning of computational thinking tasks at school in the USA and Germany show up.

KEYWORDS

computational thinking, ICILS 2018, school learning, students' characteristics

1. INTRODUCTION

Computational thinking is growing in relevance as a key competence of the 21st century (Voogt, Fisser, Good, Mishra, & Yadav, 2015). From the perspective of Aho (2012), it is seen as a set of thought processes that are used to model problems and their solutions in a way that algorithmic processing becomes possible. The competences in computational thinking thus concern cognitive processes that go far beyond the mere application of hardware and software. In this understanding, computational thinking focuses on problem-solving processes that can be made accessible through the development and application of algorithms, associated processes of modeling and formalization of implementation on a computer or digital system. Students develop problem-solving skills in computational thinking that are independent of a programming language or development environment and can include both subject-specific and general aspects of problem-solving skills (Labusch, Eickelmann & Vennemann, 2019).

However, computational thinking is differently or even not at all anchored in school curricula worldwide. According to analyses for the European Commission, computational thinking was already anchored in eleven European education systems (Bocconi, Chioccariello, Dettori, Ferrari & Engelhardt, 2016) as early as 2016, with further countries being added since then. In the overview of the different approaches in different countries and educational systems, also on an international level, three different approaches to the curricular anchoring of computational thinking can be identified (Eickelmann, 2019): (1) computational thinking as a cross-curricular competence, (2) computational thinking as part of computer science, and (3) computational thinking as an individual subject or learning area.

However, since the first works by Papert (1980) and Wing (2006), a de facto consensus has emerged in theoretical or concrete curricular approaches on what is termed computational thinking regardless of the form in which the curriculum is anchored and how the concept of these competences has been developed. For the design of school support there is therefore still a need to advance the development of generally accepted strategies for describing and assessing competences in computational thinking (Barr & Stephenson, 2011). In addition, the dynamics of the competence area impeded the development of a theoretically sufficiently elaborated concept of computational thinking over the years, rendering this competence area difficult to measure (Grover & Pea, 2013).

The empirical investigation of computational thinking has so far been complicated not least by the diversity of theoretical and empirical approaches and the diversity of the definitions underlying the often rather smaller studies or even by the complete lack of a working definition, and thus the lack of an explanation of the theoretical approach (e.g. Curzon, Bell, Waite & Dorling, 2019). Although not all curricula explicitly mention the field of computational thinking, often there are elements that can be assigned to this field. This shows that the constructs computational thinking is based on are in some places anchored in principle in the curriculum, but in many cases have not always been bundled to achieve their goals. Only in recent years, several studies emerged in an international context which explicitly focus on computational thinking. The results of these studies include the fact that existing test instruments are partly complementary. While the evaluation of items of the Bebras

competition (Dagiene & Futschek, 2008) refers to the analytic and apply levels of the taxonomy – i.e. general analytical thinking – and the evaluation mechanisms of the Dr. Scratch environment (Moreno-León & Robles, 2015), the Computational Thinking Test (CTt; Román-González, 2015) with the levels *Understand and Remember* focuses on conceptual knowledge in computational thinking (Curzon et al., 2019).

The IEA (*International Association for the Evaluation of Educational Achievement*) study ICILS 2018 (*International Computer and Information Literacy Study*) closes this gap (Fraillon et al., 2019). For the first time, an international additional module to investigate competences in computational thinking has been introduced. In an international comparison, the competences of eighth graders have been examined based on the representative student sample of ICILS 2018 by means of computer-based student tests developed in particular for this area, and the conditions for acquiring these competences assessed by background questionnaires. As this is an international option to the study, only a part of the in ICILS 2018 participating countries, including the Republic of Korea, the USA and Germany, participate in the additional module (Eickelmann, Bos, Gerick, Goldhammer, Schaumburg, Schwippert, Senkbeil, & Vahrenhold, 2019; Fraillon, Schulz, Friedman, & Duckworth, 2019).

Within the scope of the additional module of the ICILS 2018 study, an international group of experts developed a theoretical measurement construct for the field of computational thinking, which incorporates and evaluates existing approaches and concepts in the field of computational thinking, and thus combines them. The theoretical construct also formed the basis for the development of the computer-based test modules used in ICILS 2018 (Fraillon et al., 2019). In this construct, a distinction is made in the area of computational thinking between conceptualizing a problem (strand I) and operationalizing a solution (strand II). In ICILS 2018, computational thinking is defined as "an individual's ability to recognize aspects of real-world problems which are appropriate for computational formulation and to evaluate and develop algorithmic solutions to those problems so that the solutions could be operationalized with a computer" (Fraillon et al., 2019, p. 91).

A closer look at computational thinking in school reveals, for instance, that slightly less than two fifths (39.9%) of eighth graders in Germany have, according to their own statements, learned to break down a complex process into smaller parts at school at least to a medium extent. On international average (49.4%), the proportion is significantly higher than in Germany, as is the case for Luxembourg (47.5%), the Republic of Korea (57.5%), Finland (58.6%), Denmark (62.9%) and the USA (70.8%) (Eickelmann et al., 2019). Differences between the countries can already be stated at this point. A systematic investigation of the relationship between students' competences in computational thinking and their school learning of computational thinking in an international comparison is lacking. However, this would be important for the further development of teaching computational thinking.

Various studies have also focused on different groups of students according to individual student characteristics, in particular gender. Román-González, Pérez-González, and Jiménez-Fernandez (2017) found a statistically significant difference in competences in favor of the male members of the test group ($t = 5{,}374$; $p < 0.01$; effect size Cohens $d = 0.31$). Atmatzidou and Demetriadis (2016) report that the computational thinking skills of girls improved significantly after an intervention and that girls and boys ultimately achieved the same level of qualification through the intervention. In other studies, e.g. by Werner, Denner, Campe and Kawamoto (2012) and Yadav et al. (2014), no gender differences were found. In ICILS 2018 there are, for instance, no differences in average competences in computational thinking between girls and boys in the Republic of Korea and in Germany, but in the USA of 7 points in favor of boys. The dependence of student competence on their socio-economic background is known for other domains, e.g. mathematics and science (OECD, 2019). In all ICILS-2018-participating countries there are striking differences in performance, differentiated by students' social background (Eickelmann et al., 2019; Fraillon et al., 2019). Other studies do not tend to focus on the relationship between students' competences in computational thinking and their social background.

Bringing all these insights together, it emerges that there is a lack of information on how school learning of computational thinking relates to competences in computational thinking under control of background characteristics that have previously been described as pervasive.

In order to investigate this in more depth on an international comparative basis, three countries participating in ICILS 2018 from three different continents were selected: The Republic of Korea (Asia), the USA (North America) and Germany (Europe). Thereby different educational systems are selected, which differ in teaching and culture.

The current contribution thus deals with the following research question:

To what extent can differences in students' average competences in computational thinking be explained by students' social background, by learning computational thinking tasks at school and by students' gender in three countries from three different continents?

2. STUDY AND METHODS

The research question will be answered with data from the internationally comparative large-scale assessment ICILS 2018 (*International Computer and Information Literacy Study 2018*; 2015–2019), which is coordinated by the IEA for the second time after ICILS 2013 (Fraillon et al., 2019). In an international add-on module to the ICILS 2018 study, the competences of eighth graders in the area of computational thinking were also measured for the first time in an international comparison (Eickelmann et al., 2019). In addition to the students' competences, the theoretical framework model of the study also covered the conditions for acquiring competences. Information on schools and individual prerequisites and processes was collected via

background questionnaires for the tested students, teachers, school principals and ICT coordinators.

In each country that participated in ICILS 2018, the representative data basis realized via the tests and questionnaires was supplemented by information on context conditions collected from a national context survey. Nine of the ICILS 2018 participants, namely Denmark, Finland, France, Germany, Luxembourg, Portugal, the Republic of Korea, the USA and the benchmark participant North Rhine-Westphalia (federal state of Germany), participated in the international option computational thinking (Eickelmann et al., 2019; Fraillon et al., 2019). For three out of nine participants from three different continents – the Republic of Korea (N=2.875 students), the USA (N=6.790 students) and Germany (N=3.655 students) – in-depth analyses were carried out and the results are reported in the following, to answer the research question by means of a regression analysis.

Thus, the initial task involved identifying and measuring competences in computational thinking, the students' social background, their school learning in computational thinking, and their gender. They were measured with an internationally developed and elaborated set of instruments along a theoretical framework model in nine educational systems worldwide. The regression analysis comprises four models, whereby competences in computational thinking represent the dependent variable in regression modeling.

Computer-based competence tests with a live software environment were developed and used to assess the competences in computational thinking of students in the eighth grade. Each student worked on two 25-minute test modules in computational thinking, including, for instance, visual coding tasks, nonlinear systems transfer tasks and simulation tasks (Eickelmann et al., 2019; Fraillon et al., 2019).

In the first two of the four models, the students' social background represents independent variables. In model I, cultural capital is taken as an indicator of social background, operationalized by the number of books the students' family own at home. In educational research, the number of books at home has proven to be a particularly effective indicator of the students' cultural capital (Hatlevik et al., 2018). The regression analyses refer to the distinction between students whose families have a maximum of 100 books (low cultural capital) and those who have more than 100 books (high cultural capital) at home (Eickelmann et al., 2019).

Model II incorporates the medium and high HISEI values, which consider the economic resources in the parental home as a further indicator of social background. The so-called *International Socio-Economic Index of Occupational Status* (ISEI; Ganzeboom, de Graaf, Treiman, & de Leeuw, 1992) is an internationally standardized set of instruments to classify occupations and translate this into income estimates. The regression analysis refers to the highest occupational status of parents (HISEI). A low HISEI value (below 40 points) applies to postmen and women, train attendants and hairdressers, for example. Police officers, nurses, social workers and administrative specialists have a medium HISEI value (40 to 59 points). A high HISEI value (60 and more points) is allocated, for example, to teachers, journalists and lawyers.

In model III, the internationally developed index (Fraillon et al., 2019) for students' learning of computational thinking tasks at school (Cronbach's α = .90) is used as an independent variable. This index was formed based on a scale in the student background questionnaire. Students were asked to what extent they have been taught how to do different computational thinking related tasks (e.g. *to break down a complex process into smaller parts*) in the current school year. *To a large extent*, *To a moderate extent*, *To a small extent*, and *Not at all* were at their disposal as reply options.

In model IV the students' gender - differentiated into male and female - was introduced.

In the following, the unstandardized regression coefficients for each of the three countries Republic of Korea, USA and Germany are reported in four-step regression models, so that it is possible to interpret the content of these coefficients as point values by which the average student achievement (constant) changes when controlled by social background, students' learning of computational thinking tasks at school and gender. The coefficient of determination R^2 as a quality measure of linear regression indicates how well the independent variables are suited to explain the variance of the dependent variable or to predict its values.

The sampling procedure in ICILS 2018 corresponded to the design of a two-stage cluster sample in which standard errors of a relevant statistic were estimated using the *Jackknife Repeated Replication Technique* (Rust, 2014). The analyses were performed using the *IEA IDB Analyzer* (Rutkowski et al., 2010), which was used as an add-on program to the *IBM SPSS Statistics 25* software and estimates with corresponding student-level sample weights.

3. RESULTS

The following three tables show the resulting regression models for the Republic of Korea (table 1), the USA (table 2) and Germany (table 3).

Table 1. Regression Model Explaining Differences in Students' CT by their Social Background, School Learning of CT and Gender in the Republic of Korea.

	Model I		Model II		Model III		Model IV	
	b	(SE)	b	(SE)	b	(SE)	b	(SE)
cultural capital[A]	31.7*	(6.5)	23.5*	(6.2)	23.6*	(5.8)	23.6*	(5.8)
medium HISEI value	-	-	16.0*	(6.1)	15.7*	(6.0)	16.1*	(6.1)
high HISEI value	-	-	26.8*	(7.5)	27.6*	(7.2)	27.8*	(7.2)
students' learning of computational thinking tasks at school[B]	-	-	-	-	-0.4*	(0.2)	-0.5*	(0.2)
gender[C]	-	-	-	-	-	-	-7.8	(4.8)
constant	515.3		510.3		532.0		536.4	
R^2	.02		.02		.03		.03	

b - regression weight (unstandardized).
dependent variable: students' computational thinking.
* significant coefficient (p < .05).
[A] 0 - maximum of 100 books; 1 - more than 100 books.
[B] international index (M = 50, SD = 10).
[C] 0 - male; 1 - female.

IEA: International Computer and Information Literacy Study 2018 © ICILS 2018

Table 2. Regression Model Explaining Differences in Students' CT by their Social Background, School Learning of CT and Gender in the USA.

	Model I		Model II		Model III		Model IV	
	b	(SE)	b	(SE)	b	(SE)	b	(SE)
cultural capital[A]	62.1*	(3.5)	44.4*	(3.2)	43.5*	(3.3)	44.2*	(3.3)
medium HISEI value	-	-	22.5*	(3.2)	21.8*	(3.4)	21.9*	(3.3)
high HISEI value	-	-	48.0*	(3.9)	46.9*	(4.2)	46.4*	(4.3)
students' learning of computational thinking tasks at school[B]	-	-	-	-	0.0	(0.2)	0.1	(0.2)
gender[C]	-	-	-	-	-	-	-14.6*	(3.4)
constant	477.4		466.8		468.5		474.3	
R^2	.07		.09		.09		.09	

b - regression weight (unstandardized).
dependent variable: students' computational thinking.
* significant coefficient (p < .05).
[A] 0 - maximum of 100 books; 1 - more than 100 books.
[B] international index (M = 50, SD = 10).
[C] 0 - male; 1 - female.

IEA: International Computer and Information Literacy Study 2018 © ICILS 2018

Table 3. Regression Model Explaining Differences in Students' CT by their Social Background, School Learning of CT and Gender in Germany.

	Model I		Model II		Model III		Model IV	
	b	(SE)	b	(SE)	b	(SE)	b	(SE)
cultural capital[A]	64.1*	(5.8)	48.6*	(5.5)	54.0*	(5.6)	54.9*	(5.6)
medium HISEI value	-	-	30.2*	(5.9)	24.5*	(5.4)	24.8*	(5.4)
high HISEI value	-	-	51.2*	(8.0)	45.1*	(7.2)	44.3*	(7.2)
students' learning of computational thinking tasks at school[B]	-	-	-	-	-0.7*	(0.3)	-0.8*	(0.3)
gender[C]	-	-	-	-	-	-	-13.9*	(5.0)
constant	459.2		443.5		480.6		489.8	
R^2	.10		.13		.14		.15	

b - regression weight (unstandardized).
dependent variable: students' computational thinking.
* significant coefficient (p < .05).
[A] 0 - maximum of 100 books; 1 - more than 100 books.
[B] international index (M = 50, SD = 10).
[C] 0 - male; 1 - female.

IEA: International Computer and Information Literacy Study 2018 © ICILS 2018

The first regression model (model I) shows that eighth-graders with high cultural capital (more than 100 books in the home) in the Republic of Korea achieve on average 31.7 points more in competences of computational thinking than those from families with low cultural capital (a maximum of 100 books in the home). This difference is significant. With Model I, 2 percent of the variance in competences in computational thinking can be explained for the Republic of Korea. In the USA, a significant difference of 62.1 points can be observed regarding cultural capital, which is substantially higher than in the Republic of Korea. The variance explanation is 7 percent. In Germany, there is even a significant difference of 64.1 points in cultural capital with a variance explanation of 10 percent.

Moreover, considering the economic resources in the parental homes, operationalized via the HISEI (model II), it is evident in all three countries that students from economically privileged parental homes achieve significantly higher scores in computational thinking than those living under economically less privileged conditions. In the Republic of Korea, the difference in cultural capital is reduced to 23.5 points. The difference between students with medium HISEI and those with other values is 16.0 points, while the difference between students with high HISEI and others is 26.8 points. In the Republic of Korea, as in the previous model, 2 percent of the variance can be explained with model II. In the USA the difference in cultural capital is reduced as well, in this case to 44.4 points. The difference by the medium HISEI value is 22.5 points and the difference by the high HISEI value is 48.0 points. The variance explanation of model II in the USA is 9 percent. In Germany, the difference according to cultural capital under controlling for HISEI is at 48.6 points. The difference between students with medium HISEI value and others is 30.2 points and the difference between students with high HISEI value and others is 51.2 points. The variance explanation for model II in Germany is 13 percent.

Furthermore, taking the index students' learning of computational thinking tasks at school into account (model III), in the Republic of Korea there is a significant difference of 0.4 points. The relation between students' competences in computational thinking and their learning of computational thinking tasks at school under control of their social background is negative. The involvement of the selected index increases the variance explanation of the competence to 3 percent. In the USA, there is no performance difference regarding students' learning of computational thinking tasks at school. The variance explanation does not change compared to the previous model and still amounts to 9 percent. In Germany, under control of the students' social background, a significant negative relationship between the students' competences in computational thinking and their learning of computational thinking tasks at school emerges (-0.7 points). The variance explanation increases to 14 percent.

In the final model IV, the gender of the students is also taken as a predictor of competences in computational thinking. Under consideration of students' social background and their learning of computational thinking tasks at school, there is no significant performance difference between girls and boys in the Republic of Korea. The overall model thus explains 3 percent of the performance differences. The performance difference according to cultural capital in the Republic of Korea is 23.6 points in model IV as in model III, 16.1 points in the medium HISEI value and 27.8 points in the high HISEI value. With regard to students' learning of computational thinking tasks, a significant negative relationship to competences in computational thinking of 0.5 points results under control of social background and gender. In the USA, boys under control of social background and the learning of computational thinking tasks achieve significantly higher competences in computational thinking on average by 14.6 points than girls (model IV). The overall model thus explains 9 percent of the performance differences. The performance difference by cultural capital in the USA is 44.2 points in Model IV, 21.9 points in medium HISEI value and 46.4 points in high HISEI value. Regarding students' learning of computational thinking tasks under control of social background and gender there is no relationship to competences in computational thinking. In Germany, boys under control of social background and the learning of computational thinking tasks achieve 13.9 points more on average and therefore significantly higher competences in computational thinking than girls. The overall model explains 15 percent of the differences in

performance. The performance difference according to cultural capital in Germany is 54.9 points in Model IV, 24.8 points at medium HISEI value and 44.3 points at high HISEI value. Regarding students' learning of computational thinking tasks, there is still a significant negative correlation to competences in computational thinking under control of social background and gender. This is -0.8 points.

In summary, there are large differences between and within the three countries. For example, there are substantial differences in competences in computational thinking by high and low cultural capital. The performance is thus closely linked to social background. However, this difference is not as substantial in the Republic of Korea as in the USA and Germany. This is also reflected in the fact that only 2 percent of the variance can be explained in Model I in the Republic of Korea, but 7 percent in the USA and even 10 percent in Germany. Despite the addition of a further indicator of social background and under the control of cultural capital, the explanation of variance remains the same in the Republic of Korea, in the USA it rises to 9 percent and in Germany to 13 percent. At this point after model II, the explanation of variance in the USA's regression model no longer alters. Also, there is no relationship between students' school learning of computational thinking and their competences in computational thinking, not even under control of the gender of the students. Although there is a performance difference between girls and boys in favor of boys, this does not explain any further variance. In the Republic of Korea and in Germany, there is a slight but remarkable significant negative relationship between students' competences in computational thinking and their learning of computational thinking tasks at school. This results in a slightly higher variance explanation in model III according to model II in both countries. In Germany, this is further increased by the addition of gender in model IV, where under control of social background and students' learning of computational thinking tasks at school there is a significant difference in performance in computational thinking in favor of boys and more variance is explained. In the Republic of Korea, no more variance is explained in model IV and there is also no difference in performance between girls and boys. It can therefore be stated that in Germany, it is primarily the student characteristics, but also to a certain extent school learning, that play a role in the competences in computational thinking. In the Republic of Korea, social background and school learning play a role, but rather a subordinate one: hardly any variance is explained. In the USA, background characteristics play a role, but school learning does not.

4. CONCLUSION

It is certainly not unexpected that there are different results between countries. Common to all three countries is the close relationship between competences and social background, also remaining under control of other variables. This is worrying because a large proportion of the students worldwide have less chance of educational success due to their social background. It would be advisable to reduce differences so that all students – no matter how privileged the families are – can successfully participate in society. However, the ratio between students' competences in computational thinking and their learning of computational thinking at school varies across the three countries. While no correlation can be found in the USA under control of individual student characteristics, it is even slightly negative in the Republic of Korea and in Germany. Since an index was used for the present analyses, in further in-depth analyses it would be necessary to take another look at which aspects of computational thinking are particularly beneficial in teaching, but nevertheless the teaching of computational thinking should in any case be organized in such a way that it promotes students' competences. To this end, in some countries it is initially necessary to embed computational thinking in school curricula. Another approach might be to teach computational thinking in a gender-sensitive way to reduce differences in competence between girls and boys. Generally speaking, the results show that there is a major need for development in all countries, and here it could be an objective to work on a number of adjustments in order to improve the results over the next few years, particularly with a view to ICILS 2023, and to give every student the opportunity to have sufficient competences in computational thinking in order to participate in society and later acquire a good profession.

5. REFERENCES

Aho, A.V. (2012). Computation and Computational Thinking. *Computer Journal, 55*(7), 833–835.

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.

Barr, V., & Stephenson, C. (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), 48–54.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice.* Luxembourg: Publications Office of the European Union.

Curzon, P., Bell, T., Waite, J., & Dorling, M. (2019). Computational Thinking. *The Cambridge Handbook of Computing Education Research.* Cambridge: Cambridge University Press, 513–546.

Dagiene, V., & Futschek, G. (2008). Bebras International Contest on Informatics and Computer Literacy: Criteria for Good Tasks. *Informatics education – Supporting computational thinking. ISSEP 2008. Lecture notes in Computer Science.* Berlin, Germany: Springer, 19-30.

Eickelmann, B. (2019). Measuring Secondary School Students' Competence in Computational Thinking in ICILS 2018 – Challenges, Concepts and Potential Implications for School Systems around the World. *Computational Thinking Education*. Singapore: Springer, 53-64.

Eickelmann, B., Bos, W., Gerick, J., Goldhammer, F., Schaumburg, H., Schwippert, K., Senkbeil, M., & Vahrenhold, J. (Eds.) (2019). *ICILS 2018 #Deutschland.*

Computer- und informationsbezogene Kompetenzen von Schülerinnen und Schülern im zweiten internationalen Vergleich und Kompetenzen im Bereich Computational Thinking [ICILS 2018 #Germany. Students' computer and information literacy in second international comparison and competences in computational thinking]. Münster, Germany: Waxmann.

Fraillon, J., Schulz, W., Friedman, T., & Duckworth, D. (2019). *Assessment Framework of ICILS 2018*. Amsterdam: IEA.

Ganzeboom, H.B.G., de Graaf, P.M., Treiman, D.J., & de Leeuw, J. (1992). A Standard International Socio-economic Index of Occupational Status. *Social Science Research, 21*(1), 1–56.

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

Hatlevik, O.E., Throndsen, I., Loi, M., & Gudmundsdottir, G.B. (2018). Students' ICT Self-efficacy and Computer and Information Literacy: Determinants and Relationships. *Computers & Education, 118*, 107–119.

Labusch, A., Eickelmann, B., & Vennemann, M. (2019). Computational Thinking Processes and their Congruence with Problem-solving and Information-processing. *Computational Thinking Education*. Singapore: Springer, 65-78.

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*. New York, USA: ACM, 132-133.

OECD (2019). *PISA 2018 Results (Volume II): Where All Students Can Succeed*. Paris, France: PISA, OECD Publishing.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas.* New York: Basic Books, Inc. Publishers.

Román-González, M. (2015). Computational Thinking Test: Design Guidelines and Content Validation. *Proceedings of the 7th Annual International Conference on Education and New Learning Technologies (EDULEARN 2015)*. Valencia, Spain: IATED Academy, 2436–2444.

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernandez, C. (2017). Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678–691.

Rust, K.F. (2014). Sampling, weighting and variance estimation in international large-scale assessment. In L. Rutkowski, M. von Davier & D. Rutkowski (Eds.), *Handbook of international large-scale assessment. Background, technical issues and methods of data analysis* (pp. 117–153). London: Chapman & Hall/CRC Press.

Rutkowski, L., Gonzalez, E., Joncas, M., & von Davier, M. (2010). International Large-Scale Assessment Data: Issues in Secondary Analysis and Reporting. *Educational Researcher, 39*(2), 142–151.

Werner, L., Denner, J., Campe, S., & Kawamoto, D.C. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 215–220.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–35.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J.T. (2014). Computational Thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational Thinking in Compulsory Education: Towards an Agenda for Research and Practice. *Education and Information Technologies, 20*(4), 715–728.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Coding Education in K-12

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

An Item Response Theory Analysis of the Sequencing of Algorithms & Programming Concepts

Nathalia da Cruz ALVES[1*], Christiane GRESSE VON WANGENHEIM[2*], Jean Carlo Rossa HAUCK[3*], Adriano Ferreti BORGATTO[4], Dalton Francisco de ANDRADE[5]
[1,2,3,4,5]Department of Informatics and Statistics, Federal University of Santa Catarina, Florianópolis/Brazil
nathalia.alves@posgrad.ufsc.br, c.wangenheim@ufsc.br, jean.hauck@ufsc.br, adriano.borgatto@ufsc.br, dalton.andrade@ufsc.br

ABSTRACT

In order to guide computing education in K-12, several curricula and standards have been proposed, including the prominent K-12 Computer Science Framework. However, noting significant differences in content sequencing between curriculum guidelines, a question arises as to whether the proposed sequence is appropriate for learning. Furthermore, an analysis of the difficulty of these concepts is necessary to assist scaffold content sequencing or to assign different weights to the concepts in student assessment. Therefore, this paper presents the results of a large-scale analysis of computing concepts difficulty and compares it with the standards sequencing proposed by the K-12 Computer Science Framework. Our focus is on programming concepts, as in practice computing education in K-12 is typically approached by teaching algorithms and programming concepts. We perform an analysis using Item Response Theory based on the automatic assessment of over 88,000 App Inventor projects with the CodeMaster rubric. The results demonstrate that the easiness of some concepts can be explained by their inherent characteristics, but also due to the characteristics of App Inventor as a programming tool. And, although the analysis demonstrates the alignment of the content sequencing of the K-12 Computer Science Standards with the difficulty, we also observed that some concepts related to algorithms and programming are not explicitly covered by the framework, such as strings and Boolean operators. Thus, the results of this research can be used by researchers as well as teachers to improve computing education in K-12.

KEYWORDS

computational thinking, App Inventor, K-12 computer science standards, Item Response Theory

1. INTRODUCTION

Computational thinking (CT) is making its way into K-12 worldwide (Lye & Koh, 2014). Regardless of the area of expertise, it is important to know the fundamentals and basic principles of computing so that one can perform his activity fully. CT refers to the thought processes involved in creating algorithmic, or step-by-step, solutions that can be executed by a computer (Wing, 2006). In this context, several efforts have been made to develop guidelines and curricula for K-12 computing education. One of the most prominent models is the K-12 Computer Science Framework (CSTA, 2016) defining standards, the sequencing of CT concepts and practices for different educational levels in K-12. The K-12 Computer Science Framework contains five educational levels: 1A (for grades K-2 and ages 5-7), 1B (for grades 3-5 and ages 8-11), 2 (for grades 6-8 and ages 11-14), 3A (for grades 9-10 and ages 14-16), and 3B (for grades 11-12 and ages 16-18).

In practice, computational thinking is commonly taught focusing on algorithms and programming concepts and related CT practices (Grover & Pea, 2013) being one of the main knowledge areas of computing. This comprises the competency to develop algorithms to solve problems in a language computers can understand including several sub-concepts in accordance with the K-12 Computer Science Framework (Fig. 1).

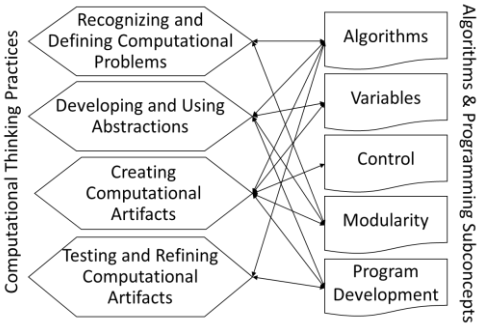


Figure 1. CT practices and algorithms & programming concepts (CSTA 2016).

Variables refer to storing and manipulating data from computer programs. Control concepts specify the order in which instructions are executed within an algorithm or program (e.g. using loops and/or conditionals). Modularity involves dividing complex tasks into simpler tasks and combining them to create something complex. Program development represents the software engineering process that is repeated until acceptance criteria are met (CSTA, 2016). In addition, several CT practices are related to algorithms & programming as presented in Figure 1 (CSTA, 2016). Other guidelines and curricula, such as Computing at School (CAS, 2015) or the Australian Curriculum, Assessment and Reporting Authority (ACARA, 2015), cover similar basic concepts and practices.

In order to introduce programming in K-12, typically visual block-based programming environments such as Scratch or App Inventor are used (Papadakis et al., 2017). Diverse instructional strategies are adopted, including well-structured exercises such as Code.org, yet, often in a constructivist context, a problem-based learning approach with open-ended ill-structured programming activities is adopted (Law, 2016; Shute et al., 2017). These instructional units typically aim at teaching students to create their own games or mobile applications to solve real-world issues (Fee

& Holland-Minkley, 2010). In order to assess open-ended ill-structured problems, often performance-based assessments are performed based on the created software artifacts (Alves et al., 2019). These assessments aligned with curricula are typically based on rubrics scoring the ability to develop a software artifact, and, thus, indirectly inferring the achievement of CT practices and concepts (Sherman & Martin, 2015). Some CT rubrics have been automatized, such as Dr. Scratch (Moreno-León & Robles, 2015) and CodeMaster (Alves et al., 2020), allowing to assess students' CT competences in an automated way.

Yet, although research and practical applications of computing education in K-12 is strongly increasing worldwide, it seems often to be based on experience rather than systematic evidence. Thus, a question that remains is related to content sequencing indicating in what order should students learn concepts. The relevance of this question is also demonstrated by research in this area. In order to analyze CT progression, Seiter & Foreman (2013) used Scratch projects of students from grades 1-6 to identify how CT concepts varied by grade. Franklin et al. (2017) analyzed student projects from grades 4-6 in sequence, events, and initialization using LaPlaya (a Scratch-like programming language). Grover & Basu (2017) analyzed students' misconceptions of loops, variables, and boolean logic from grades 6-8 also using Scratch. Lytle et al. (2019) analyzed CT progression on the "use-modify-create" lesson using Cellular environment (an extension of the block-based programming environment Snap!). Rich et al. (2017;2018;2019) analyzed K-8 learning trajectories for sequence, repetition, conditionals (Rich et al., 2017), decomposition (Rich et al., 2018) and debugging (Rich et al., 2019) in Scratch studies integrating CT into Mathematics. But, although there are several studies analyzing some aspects of CT using visual programming environments, no research focusing directly on content sequencing in relation to the K-12 Computer Science Framework and specifically with respect to App Inventor has been found.

Thus, the objective of this study is to analyze the proposed sequencing of the K-12 Computer Science Framework (CSTA, 2016) based on the observed difficulty of programming concepts in App Inventor projects. This analysis is enabled by using CodeMaster (an automated rubric). Specifically assessing CT in accordance with the K-12 Computer Science Framework, the CodeMaster automated rubric assesses several items related to algorithms and programming concepts that can be extracted from the source code.

2. CODEMASTER RUBRIC

CodeMaster is an automated performance-based assessment rubric and grader. It enables an analysis of the code of App Inventor programs supported by a free web-based tool providing feedback to students and teachers in the form of a CT score on programming projects. The model has been developed based on a systematic mapping study (Alves, 2019) following an instructional design process (Branch, 2010) and the procedure for the rubric definition proposed by Goodrich (1996). Evaluation of reliability and construct validity indicated that the CodeMaster rubric can be regarded as reliable (Cronbach's alpha α=0.84). With respect to construct validity, there also exists an indication of convergent validity based on the results of a correlation and factor analysis indicating that the rubric can be used for a valid assessment of algorithm and programming concepts of App Inventor programs as part of a comprehensive assessment completed by other assessment methods (Alves et al., 2020).

The CodeMaster rubric is composed of 16 items related to algorithm and programming concepts, however, in this work, we are considering only items related to the K-12 Computer Science Standards (Table 1) from levels 1B to 3A, and that can be found in apps of the App Inventor Gallery. We, thus excluded apps with extensions (as the App Inventor Gallery does not allow apps with extensions). Other standards, which cannot be automatically assessed, are also excluded from our analysis, as they are not present in the CodeMaster rubric.

Table 1. K-12 Computer Science Standards (CSTA, 2017) present in CodeMaster rubric.

Identifier	K-12 Computer Science Standard	A&P Subconcept	Practice
1B-AP-09	Create programs that use **variables** to store and modify data.	Variables	Creating Computational Artifacts
1B-AP-10	Create programs that include sequences, **events**, **loops**, and **conditionals**.	Control	Creating Computational Artifacts
1B-AP-11	**Decompose** (break down) problems into smaller, manageable subproblems to facilitate the program development process.	Modularity	Recognizing and Defining Computational Problems
2-AP-11	Create **clearly named variables** that represent different data types and perform operations on their values.	Variables	Creating Computational Artifacts
2-AP-13	**Decompose** problems and subproblems into parts to facilitate the design, implementation, and **review** of programs.	Modularity	Recognizing and Defining Computational Problems
3A-AP-14	Use **lists** to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.	Variables	Developing and Using Abstractions

Based on these standards, the CodeMaster rubric defines items and performance levels for each item. The performance levels descriptors of the CodeMaster rubric are derived directly from the learning objectives of the K-12 Computer Science Standards (standards identifiers are underlined in Table 2). The performance levels are described on ordinal scales, ranging from "criterion is not (or minimally) present" to advanced usage of the criterion.

Table 2. Excerpt from the CodeMaster rubric items adopted for this research (Alves et al., 2020).

Item	Performance Level			
	0	1	2	3
Variables	No use of variables.	Modification or use of predefined variables. 1B-AP-09	Creation and operation with variables. 2-AP-11	-
Naming	Few or no names are changed from their defaults.	10 to 25% of the names are changed from their defaults. 2-AP-11	26 to 75% of the names are changed from their defaults. 2-AP-11	More than 75% of the names are changed from their defaults. 2-AP-11
Lists	No lists are used.	At least one list is used. 3A-AP-14	More than one list is used. 3A-AP-14	Lists of tuples are used. 3A-AP-14

Events	No type of event handler is used.	One type of event handler is used. 1B-AP-10	Two or three types of event handlers are used. 1B-AP-10	More than three types of event handlers are used. 1B-AP-10
Loops	No use of loops.	Simple loops are used. 1B-AP-10	'For each' loops with simple variables are used. 1B-AP-10	'For each' loops with list items are used. 1B-AP-10
Conditional	No use of conditionals.	Uses 'if' structure. 1B-AP-10	Uses one 'if then else' structure. 1B-AP-10	Uses more than one 'if then else' structure. 1B-AP-10
Procedural Abstraction	No use of procedures.	One procedure is defined and called. 1B-AP-11	More than one procedure defined. 1B-AP-11	There are procedures for code organization and re-use. 2-AP-13

The assessment using the CodeMaster rubric is automated by performing a static code analysis. The analysis is done by counting the kind and the number of command blocks used with respect to algorithms and programming concepts, such as variables, conditionals, loops, etc. The automated assessment is supported by the CodeMaster tool available on-line (http://apps.computacaonaescola.ufsc.br:8080/).

3. RESEARCH METHOD

Following the Goal Question Metric approach (Basili, Caldiera & Rombach, 1994), the objective of this study is defined as to analyze the difficulty and sequencing of the standards related to CT practices and Algorithms & Programming subconcepts from the K-12 Computer Science Standards (CSTA, 2017). To achieve this goal, a case study is conducted following Yin (2017).

3.1. Data Collection

Initially, we collected data in the form of App Inventor projects from the AppInventor Gallery. In order to optimize the sample size, we downloaded the publicly available and accessible apps from the App Inventor Gallery in June 2018. As a result, we obtained the source-code from 88,864 App Inventor apps. We assessed these projects using the CodeMaster tool. Out of the 88,864 downloaded projects, 88,812 were successfully assessed with the CodeMaster rubric. 52 projects failed to be analyzed due to technical difficulties. The collected data were pooled in a single sample in order to analyze the concepts sequencing (rather than a specific app).

3.2. Data Analysis

In order to analyze the difficulty and sequencing, we use the Item Response Theory (IRT) Gradual Response Model proposed by Samejima (1969). IRT allows analyzing item properties, such as difficulty and discrimination, using falsifiable models. This is done by estimating the correspondence between an unobserved latent trait, in this case, CT, and observable evidence, in this case, the assessed App Inventor apps. The Gradual Response Model assumes that items are polytomous and its response categories are ordered (such as in CodeMaster rubric). Samejima's model proposes a probabilistic model for parameter estimation that is not dependent on a specific set of items and is used to determine the probability for someone to receive a specific score (or higher), given the level of the underlying latent trait, which in this context is CT.

Adopting IRT, for each item is estimated: the parameter *a* (common to all item categories) and the parameters *b's*, indicating the distance from adjacent difficulty performance levels (see Fig. 2). The dataset was analyzed using the mirt package from the R programming language (Chalmers, 2012).

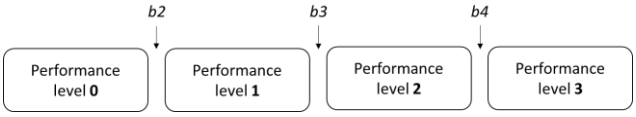


Figure 2. Difficulty parameters (*b*'s) for items with 4 adjacent difficulty performance levels (as in CodeMaster rubric).

Due to the focus on the individual properties of each item, IRT allows the placement of items on a scale that distinguishes what is easier and harder from the learner's point of view. Using the scale, the items order relations are compared to the K-12 Computer Science Standards sequencing.

4. ANALYSIS

4.1. Parameter estimation and scale creation

Using the Gradual Response Model (Samejima, 1969) the parameters of the items are estimated. The metric is established by setting population parameters to average = 0 and standard deviation = 1. Since the CodeMaster rubric contains ordinal polytomous items, several b parameters are estimated to differentiate the passage from one score to another:

- b_2 = represents the difficulty of getting score 1 on any item,
- b_3 = represents the difficulty of getting score 2 on any item,
- b_4 = represents the difficulty getting score 3 on any item.

Consequently, items on a 2-point ordinal scale (no description for score 3) also do not present a parameter b_4 (example: item variables). In IRT, parameters *a* and *b's* can theoretically assume any real value between $-\infty$ and $+\infty$. However, a negative value for *a* parameter is not expected. Typically values above 1.0 are considered good, as they indicate that the item discriminates well learners with different abilities. In this study, parameters *b* are the main indicators to be analyzed, as they indicate the difficulty of the item. For parameters *b*, values close to or within the range [-5, 5] are expected, with negative values indicating that an item has below average difficulty and positive values indicating above average difficulty.

In general, most items were well estimated, with *a* parameter value above 1 (Table 3). In addition, the values of the difficulty parameters (b_2, b_3, and b_4) are within the range [-5, 5]. Only the item Lists presented parameter b_4 slightly above 5. Standard errors (SE) of each parameter b presented similar results and are in low magnitude, therefore, presenting no estimation problem.

Table 3. Parameters *a* and *b's* estimated with standard errors (SE).

Item	a (SE)	b_2 (SE)	b_3 (SE)	b_4 (SE)
Variables	2.97 (0.02)	-0.83 (0.01)	-0.01 (0.01)	NA

Naming	1.68 (0.01)	-0.31 (0.01)	0.07 (0.01)	1.89 (0.01)
Lists	1.24 (0.01)	1.49 (0.01)	2.00 (0.02)	5.20 (0.07)
Events	2.88 (0.02)	-1.65 (0.01)	-0.90 (0.01)	-0.47 (0.01)
Loops	1.77 (0.03)	2.14 (0.02)	2.29 (0.02)	2.57 (0.03)
Conditional	2.32 (0.02)	0.34 (0.01)	0.80 (0.01)	1.57 (0.01)
Procedural Abstraction	3.18 (0.03)	0.99 (0.01)	1.08 (0.01)	1.19 (0.01)

Analyzing the results, it can be inferred that obtaining 1 point for the item Events is easier than in any other item since this item has the smallest *b* parameter (b_2 = -1,65). On the other hand, obtaining 3 points for item Lists is more difficult than any other item, as it presents the highest value for a *b* parameter (b_4 = 5.20).

Based on the estimated difficulty parameters, the items are placed on a scale (0.1), i.e. with average = 0 and standard deviation = 1 (Figure 3). The scale is an "arbitrary" scale where the relations of order between its points are most important and not necessarily its magnitude. The items are arranged at the scale points according to the estimated difficulty parameters (b_2, b_3, and b_4), as presented in Table 3. For example, the b_2 parameter of the item Events is equal to -1,65, so it is placed at point -1.5 of the scale.

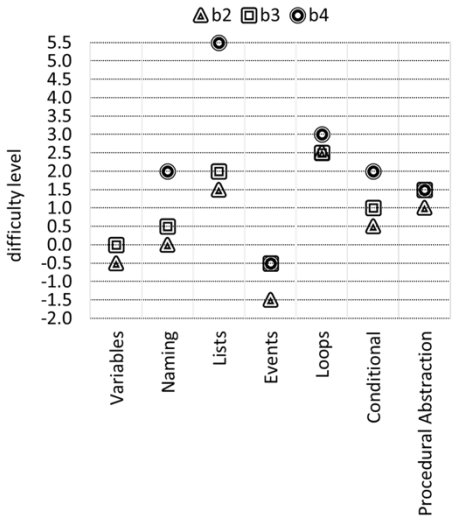


Figure 3. Placement of the items on the scale.

From the placement of items on the scale, we can infer that an item with a parameter *b* estimated at 1.5 is 1.5 standard deviations above the average ability. Thus, this item is more difficult than all items that are placed below point 1.5 at the scale. In the context of App Inventor programming, the easiest items include events and variables (Figure 3), as these items have negative (below average) parameters *b*. These parameters are semantically consistent, as App Inventor encourages variable creation and unlimited use of events (Turbak et al., 2014).

The most difficult items include lists and loops. Score 3 for the list item has the highest difficulty parameter (Lists b_4), being the most difficult to achieve among all items. Although the loops item is also considered difficult, it is noteworthy that loop blocks in App Inventor programs are rarely used because many iterative processes that would be expressed with loops in other programming languages are expressed as an event that performs a single step of the iteration every time it is triggered (Turbak et al., 2014). Thus, the difficulty parameters of loops may be poorly represented through the App Inventor dataset, as more than 94% of apps are assessed with 0 points in loops (see fig. 4). In other visual programming environments, such as Scratch, the usage of this concept and consequently the observed difficulty may be different.

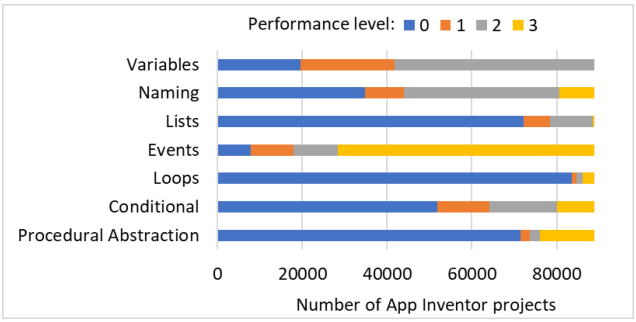


Figure 4. Frequency of the performance level score for each item.

4.2. Comparison of K-12 Computer Science Standards sequencing with estimated IRT parameters

Based on the results of the scale placement (Fig. 4), we analyze the content sequencing proposed by the K-12 Computer Science Standards (Fig. 5). Here we expect that CodeMaster items that are easier should be sequenced on early levels (1B or 2) and those that are difficult are sequenced on final levels (2 or 3A).

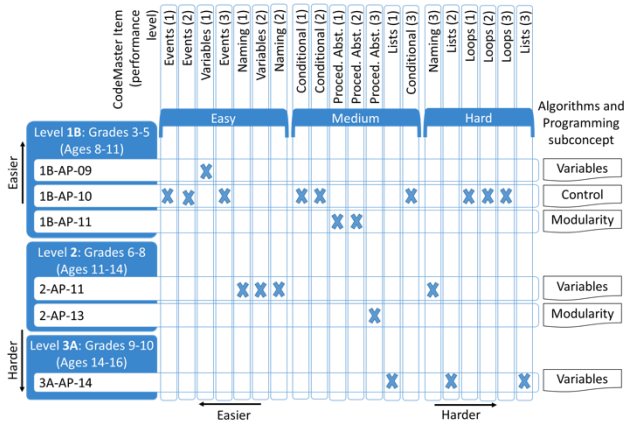


Figure 5. Positioning items on the scale.

Most of the items are sequenced in accordance with their difficulty. For example, easy items are placed on levels 1 and 2, e.g., events, variables, and naming. Events are widely used in App Inventor programs; even simple apps need events in order to function properly. Variables are also widely used, both as predefined and blocks to create/modify user variables. However, there are also some discrepancies between the degree of difficulty based on the IRT analysis and the placement of standards throughout K-12 (Figure 5). Naming (components and procedures) is essential to make the program understandable. However, it seems to be much harder than obtaining a low-medium performance level (1 or 2 points) than to obtain a high-performance level (3 points) for Naming. In other words, the gap between Naming(2) and Naming(3) is bigger than the gap between Naming(1) and Naming(2).

Items with medium difficulty are placed throughout all K-12 Computer Science Standards levels (1B to 3A), including Conditional, Procedural Abstraction and the first

performance level for Lists (since Lists(2) and Lists(3) were placed together with difficult items). Conditionals include "if-then" or "if-then-else". Although it seems there is no difference in using "if-then" (Conditionals(1)), or "if-then-else" (Conditionals(2)), using more than one "if-then-else" (Conditionals(3)) is much harder as it was almost placed with difficult items (Fig. 5). Item Lists(1) was the only level 3A item derived from the K-12 Computer Science Standards placed as a medium item.

Difficult items include Lists(2), Lists(3), Loops and Naming(3). Being a level 3A concept in the K-12 Computer Science Standards (CSTA, 2017), Lists performance levels are expected to be placed together with more difficult items. However, Loops are a level 1A concept following the K-12 Computer Science Standards, yet are placed together with difficult concepts. This, on the other hand, may be explained by the underrepresentation in the dataset as well as its needless use in many cases in App Inventor as stated by Turbak et al. (2014).

5. DISCUSSION

Considering the difficulty of items, we identified that events and variables are the easiest items when programming with App Inventor. Items with medium difficulty include conditional and procedural abstraction. The most difficult items are loops and lists, while the estimated high difficulty of loops may be influenced by its infrequent use in App Inventor projects.

Analyzing the proposed content sequencing of curriculum guidelines on the example of the CSTA framework, we can observe that it is adequately aligned with the difficulty as determined via the IRT analysis (with exception of loops). For example, the proposal indicates addressing list concepts in level 3, being one of the most difficult concepts this is appropriate and allows a scaffolding approach.

However, some concepts related to algorithms and programming are not explicitly included in CSTA framework/standards, for example, strings and Boolean operators, and, thus, were excluded from this analysis. Further evolutions of curriculum guidelines could, therefore, also explicitly start to include these concepts.

5.1. Threats to validity

One risk is related to grouping data from different contexts. The App Inventor programs come from various contexts in the worldwide App Inventor community, and no additional information about the creator history in App Inventor Gallery projects is available. As the goal in this work is to identify the relationship between standards difficulty, this is not considered a problem, although the results should be perceived considering that there is no information about which context the apps were created. Another threat regarding the possibility of generalizing the results is related to the sample size and projects of only one programming language. As one of the leading visual programming environments, App Inventor contains many of the algorithmic and programming concepts covered in K-12 computing education and is similar to other environments, such as Scratch. Therefore, this risk is minimized by using a significant number of apps (over 88,000). Thus, the sample size is considered satisfactory, allowing the generation of significant results. Regarding the construct validity, measurements were systematically defined and data extraction was performed automatically, eliminating errors from manual extraction. The statistical technique used for the analysis was chosen based on the literature as one of the indicated techniques for this purpose.

6. CONCLUSIONS

Based on App Inventor projects, we analyzed the difficulty of Algorithms & Programming concepts as well as the alignment of the content sequencing as proposed by the K-12 Computer Science Standards. We noticed that the sequencing of the standards is consistent with the difficulty of the concepts. We also observed that the difficulty of achieving performance levels of certain items may depend on the specific programming language. For example, the loops concept in App Inventor may be more difficult to learn since there are other ways to program an iterative process. The results of this analysis can be used to systematically discuss and improve the pedagogical sequencing of curriculum guidelines by adopting scaffolding techniques and comparisons with other reference frameworks.

7. ACKNOWLEDGMENTS

We would like to thank all researchers from the MIT App Inventor team, who provided support for the access to the App Inventor Gallery. This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* - Brasil (CAPES) - Finance Code 001 and by the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* - Brasil (CNPq), entities of the Brazilian government focused on scientific and technological development.

8. REFERENCES

ACARA (2015). *Australian Curriculum, Assessment and Reporting Authority*. Retrieved June 9, 2019, from https://www.acara.edu.au/

Alves, N. da C., Gresse von Wangenheim, C., & Hauck, J. C. R. (2020). A large-scale evaluation of a rubric for the automatic assessment of algorithms and programming concepts. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM.

Alves, N. da C., Gresse von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education, 18*(1), 17-39.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. *Encyclopedia of Software Engineering*, 528-532. Wiley.

Branch, R. M. (2010). *Instructional Design: The ADDIE Approach*. New York: Springer.

CAS (2015). *Computing at School*. Retrieved June 9, 2019, from https://www.computingatschool.org.uk/

Chalmers R. P. (2012). Mirt: A multidimensional item response theory package for the R Environment. *Journal of Statistical Software, 48*(6), 1–29.

Computer Science Teachers Association (2016). *K-12 Computer Science Framework*. Retrieved August 9, 2019 from https://k12cs.org/

Computer Science Teachers Association (2017). *CSTA K-12 Computer Science Standards, Revised 2017*. Retrieved August 9, 2019 from http://www.csteachers.org/standards

Fee S. B., & Holland-Minkley, A. M. (2010). Teaching computer science through problems, not solutions. *Computer Science Education, 20*(2), 129-144.

Goodrich, H. (1996). Understanding Rubrics. *Educational Leadership, 54*(4), 14–18.

Grover, S., & Pea, R. (2013). Computational thinking in K–1: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.

Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean Logic. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, 267-272. ACM.

Law, B. (2016). Puzzle games: A metaphor for computational thinking. *Proceedings of the 10th European Conference on Games Based Learning*, 344-353. Academic Conferences and Publishing International Ltd.

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

Moreno-León J., & Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. *Proceedings of the 10th Workshop in Primary and Secondary Computing Education*, 132–133. ACM.

Papadakis, S., Kalogiannakis, M., Orfanakis, V., & Zaranis, N. (2017). The appropriateness of Scratch and App Inventor as educational environments for teaching introductory programming in primary and secondary education. *International Journal of Web-Based Learning and Teaching Technologies, 12*(4), 58-77.

Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 learning trajectories derived from research literature: Sequence, repetition, conditionals. *Proceedings of the 2017 ACM Conference on International Computing Education Research*, 182-190. ACM.

Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018). Decomposition: A K-8 computational thinking learning trajectory. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 124-132. ACM.

Rich, K. M., Strickland, C. T., Binkowski, T. A., & Franklin, D. (2019). A K-8 debugging learning trajectory derived from research literature. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 745-751. ACM.

Samejima, F. A. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometric Monograph, 34*(4, Pt.2), 17.

Seiter K., & Foreman, B. (2013). Modeling the learning progressions of computational thinking of primary grade students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, 59–66. ACM.

Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges, 30*(6), 53–59.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying Computational Thinking. *Educational Research Review, 22*, 142-158.

Turbak, F., Sherman, M., Martin, F., Wolber, D., & Pokress, S. C. (2014). Events First Programming in App Inventor. *Journal of Computing Sciences in Colleges, 29*(6), 81-89.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-36.

Yin, R. K. (2017). *Case Study Research: Design and Methods* (6th Ed.). Thousand Oaks: SAGE Publications.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Creativity: A Test for Interdependency

Rotem ISRAEL-FISHELSON[1*], Arnon HERSHKOVITZ[2*], Andoni EGUÍLUZ[3], Pablo GARAIZAR[4], Mariluz GUENAGA[5]
[1,2]School of Education, Tel Aviv University, Israel
[3,4,5] Faculty of Engineering, University of Deusto, Bilbao, Spain
rotemisrael@tauex.tau.ac.il, arnonhe@tauex.tau.ac.il, andoni.eguiluz@deusto.es, garaizar@deusto.es, mlguenaga@deusto.es

ABSTRACT
Computational Thinking (CT) and creativity are considered fundamental skills for future citizens. We studied the associations between these two constructs among middle school students (N=174), considering two types of creativity: Creative Thinking and Computational Creativity. We did so using log files from a game-based learning platform (Kodetu) and a standardized creativity test. We found that the more creative the students were (as measured by a traditional creativity test), the more effectively they acquired CT. We also found significant positive correlations between Computational Creativity and the acquisition of CT in some levels of the game, and a positive correlation between Creative Thinking and Computational Creativity.

KEYWORDS
computational thinking, creativity, game-based learning, learning analytics, log analysis

1. INTRODUCTION
The exponential growth in the data available from a plethora of resources and the significant development of science, make it essential for people to adopt skills that complement and provide the added value of computing capabilities to any field of expertise (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009). Both Computational Thinking and Creativity have been recognized as essential skills for the 21st century (Kalelioğlu, Gülbahar, & Kukul, 2016; Sai d-Metwaly, Noortgate, & Kyndt, 2017) and are crucially important for human development (Czerkawski, 2015).

Computational Thinking (CT) is the conceptual foundation required to define and solve real-world problems using algorithmic methods to reach solutions that are transferable and necessary to various contexts and disciplines (Shute, Sun, & Asbell-Clarke, 2017). It is a skill that helps improving thinking abilities and provides techniques to extract knowledge hidden in the data (Buitrago Flórez et al., 2017).

Creativity is a thinking ability that enables problem-solving in an innovative manner, and the production of original and valuable products (Torrance, 1974). Despite having many definitions to this construct, there is an agreement that creativity is a multi-dimensional variable comprised of four characteristics: (1) Fluency – the ability to generate a large number of ideas and directions of thought for a particular problem; (2) Flexibility – the ability to think about as many uses and classifications as possible for a particular item or subject; (3) Originality – the ability to think of ideas that are not self-evident or banal or statistically ordinary, but rather unusual and even refuted, and (4) Elaboration – the ability to expand an existing idea, develop and improve it by integrating existing schemes with new ideas (Guilford, 1950; Torrance, 1965).

Similar to CT, creativity has been identified as crucial to human inventive potential in all disciplines, and it is evident that its influence dominates various spheres of life (Navarrete, 2013). However, for many years, these two skills remained within their content areas - CT was mainly taught in the context of Science, Technology, Engineering, and Mathematics (STEM) fields, and creativity in the fields of design and art. We have come to a point where there is an understanding that both can be nurtured and should be included across the curriculum from an early age (Beghetto, 2010; Vygotsky, 2004). Indeed, creativity involves a set of thinking tools that overlap with the fundamentals of Computer Science—specifically, observation, imagination and visualization, abstraction, and creation and identification of patterns (Yadav & Cooper, 2017)—which can support the development of creativity. For this reason, various educational initiatives worldwide have begun to establish national K-12 curricula, academic standards, and instructional computerized and unplugged activities that promote these skills (ISTE, 2017; World Economic Forum, 2015).

With the recognition of its importance, CT has been integrated into school curricula around the world, and many online platforms, especially game-based learning platforms, have been developed to support and promote its acquisition (Kim & Ko, 2017). Some of these platforms—like CodeMonkey™ or Hour of Code™—take advantage of the game-based learning approach, which promotes learning through fun, interactive and rewarding game-play, in order to increase engagement and motivation for learning and to improve academic achievements in the long run (Ibanez, Di-Serio, & Delgado-Kloos, 2014; Kazimoglu, Kiernan, Bacon, & MacKinnon, 2012; Vu & Feinstein, 2017). However, while encouraging the acquisition of CT in a fun, engaging way, these platforms promote efficiency and sometimes limit creativity (for example, when not allowing free use of coding blocks). This is most evident when a learner submits a solution which may be considered as creative, but as it is not the most efficient solution anticipated by the platform, the learner would not get a full score for it.

Research on CT and creativity has been conducted from different perspectives, looking at both creativity within the scope of CT and the influence of the two constructs on each other (Miller et al., 2013; Seo & Kim, 2016). However, only limited research exists on the relationship between these two perspectives. Creativity may be dependent on the learning

context and the measuring tool (Reiter-Palmon, Illies, Kobe Cross, Buboltz, & Nimps, 2009). Therefore, we explore the associations between different measures and perspectives of creativity and look for connections between them and CT acquisition.

2. RESEARCH QUESTIONS

To avoid confusion, we use Creative Thinking to refer to a traditional measure of creativity that has no connection to the platform being used, and Computational Creativity to refer to a measure of how creativity is manifested inside the platform, as reflected by the frequency (originality) of a given solution among all other solutions (detailed in section 3.5). To meet our research goal, we formulated the following research questions:

1. What are the associations between the acquisition of CT and Creative Thinking?
2. What are the associations between the acquisition of CT and Computational Creativity?
3. What are the associations between Computational Creativity and Creative Thinking?

3. METHODOLOGY

3.1. The Learning Platform: Kodetu

Kodetu is a web app built using Google's Blockly for teaching basic programming skills (Eguíluz et al., 2017). The environment has three official games, and it is also allowing users to create their own games. Each of Kodetu's levels presents the user with a challenge in which an astronaut should get to a marked destination. The user has to define the astronaut's movements using coding blocks in the workspace. Each level of the game presents one or more CT concepts (e.g., *sequences*, *loops*, etc.). Moving to the next level is possible only upon completing the current level successfully. It should be noted that a user can reset the level and solve it again. The system is offered in three languages: English, Spanish, and Basque. While the app is being used, the system logs any action taken by its users.

For our broad study, a dedicated game was created in the Kodetu platform. The game includes ten levels with increased difficulty. In this paper, we present part of our work covering levels 1-9. The first four levels are designed with the aim of practicing the concept of *sequences*. Level 1 presents a trivial level to show how the system works. Level 2 and 3 involve turns and perspective. Level 4 presents a challenge where a long sequence of actions, including more than one rotation, is needed to reach the goal. Level 5 limits the number of blocks that can be used (i.e., code length) to prevent participants from using long *sequences* and to encourage them to take advantage of new code structures of *loops*. Level 6 presents a trivial challenge that deals with *sequences* and *loops*. Level 7 (Shown in Figure 1) also works on *sequences* and *loops* with limitation of blocks' usage. Level 8 limits the number of blocks that can be used (i.e., code length) to prevent participants from using long *sequences* and to encourage them to take advantage of new code structures of *conditionals*. Level 9 introduces *If-Else* conditionals and requires nested structures and a limited number of blocks. Solving the entire set of levels is intended to take 30 to 60 minutes. While the platform is being used, the system logs any action taken by its users.

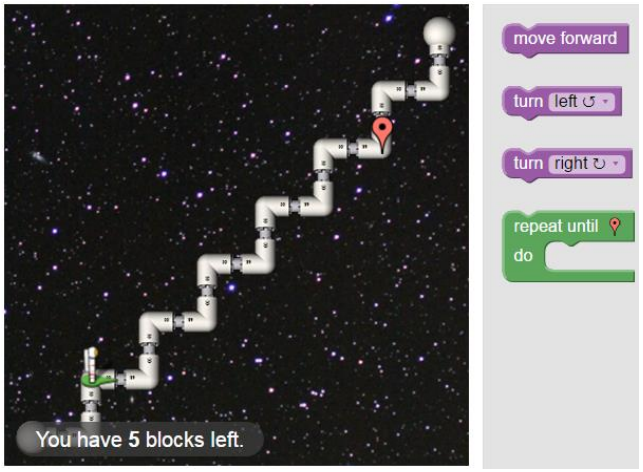


Figure 1. An Example Level of Kodetu (level 7)

3.2. Population and Research Design

For this study, we analyzed the actions of 174 middle-school Spanish students, 11-12 years old (55% boys and 45% girls) from two different schools. The students arrived to an outreach activity organized by the Faculty of Engineering of the University of Deusto and participated in a workshop about technology, programming, and robotics. During this workshop, the students played the designated Kodetu game for about 60 minutes. For the vast majority of the students, it was their first encounter with programming experience (78%, 136 of 174). In addition, 60% of students (105 of 174) reported they have a high affinity for technology.

Prior to the Kodetu session, all participants completed a pen-and-paper creativity task (Torrance's TTCT – Figural Test; see section 3.4). Data from Kodetu log files were triangulated with the data obtained via the creativity task using a unique ID for each participant. This ID was produced by Kodetu and was written down on the creativity test form by the participants. In addition, participants were asked to provide demographic data (age, gender), previous programming background (yes/no), and affinity to technology (1-low to 10-high).

3.3. Dataset and Preprocessing

The full log file included 163,137 rows, each representing an action taken by a user, including its timestamp, the level in which it was taken, its result [Success, Failure, Timeout, Error], and the code associated with this action.

3.4. Research Tool

We used the Torrance Test for Creative Thinking (TTCT) – Figural Test (Torrance, 1974) to assess Creative Thinking in four dimensions: fluency, flexibility, originality, and elaboration. In this pen-and-paper test, each student was presented with a sheet on which 12 identical, empty circles were printed. Students were asked to make as many drawings as possible using the circles as part of the drawings. An eligible drawing used the circle as part of the drawing. See examples in Figure 2.

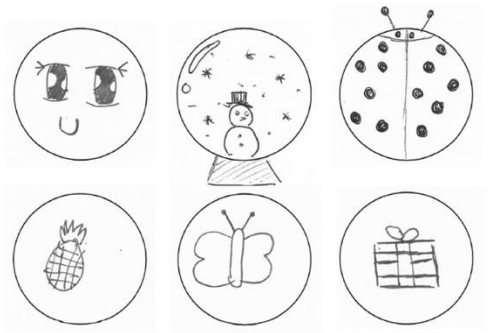


Figure 2. Example of Eligible (top row) and Non-eligible (bottom row) Drawings from TTCT – Figural Test

3.5. Variables

3.5.1. Computational Thinking

We focused on three variables to measure the acquisition of computational thinking, each computed for all levels as well as for each level separately.

- Solution Attempts.
- Correct Solution Attempts.
- Average Time [min].

3.5.2. Creative Thinking

To score the creativity task, we used eligible drawings only, that is, drawings in which the circle was considered an important part of the drawing. In order to ensure the reliability of determining eligibility, each of the first two authors coded 20 sheets for eligibility separately; then, we ran an inter-rater reliability assessment using Cohen's kappa and got a satisfying coefficient of 0.81. The authors then discussed borderline cases and agreed on guidelines for the rest of the coding, which was done by the first author.

Similarly, each of the first two authors separately coded 20 sheets for categories and then discussed their codes until full agreement achieved. The rest of the coding was done by the first author, with frequent discussions throughout this process about their very definitions and about splitting and merging categories. At the end of this iterative process, the final list consisted of 59 categories, e.g., "Emoji", "Sun", "Flower", "Signpost".

Finally, we computed the following four variables (for each student):

- Fluency – Number of eligible drawings;
- Flexibility – Number of different drawings' categories;
- Originality – Average frequency of the drawing categories, across all drawings;
- Elaboration – Number of ideas/details used in each eligible drawing;

3.5.3. Computational Creativity

Our analysis focuses on the originality of a correct solution as a proxy for creativity. This is due to the fact that the Kodetu platform, similarly to many other platforms, does not explicitly encourage multiple solutions, and once a level is solved, participants are immediately encouraged to move to the next level. Therefore, fluency, flexibility, and elaboration are not applicable in our analysis.

The originality is represented by the frequency of this solution among all the correct solutions for this level. That is, the rarer a solution is, the more creative it is considered. When there were multiple correct solutions for an individual participant, we calculated the average across her or his correct solutions. The originality was calculated for each level separately and also aggregated for all Levels.

4. FINDINGS

4.1. Descriptive Statistics of the Research Variables

In order to better understand the associations between Computational Thinking, Creative Thinking, and Computational Creativity, we first report on descriptive statistics of each of the variables. All statistical analyses used IBM SPSS version 25.

4.1.1. Computational Thinking

We found that among all participants, the average Solution Attempts was 6.16 (SD=3.08), and Correct Solution Attempt was 1.06 (SD=0.19). The Average Time it took to solve each level was 5.13 minutes (SD=11.99).

Overall, there was an increasing trend in Level Solution Attempts, with R^2=0.49 for the graph trend line (see Figure 3), indicating the increasing difficulty of the game. A similar trend was found for the Level Average Time, excluding a decrease between Level 1 to level 3, which might be related to the participants' adaptation to the interface in these initial levels. In addition, there is a decrease from level 8 to 9 that may be associated with the presentation of the concept of *conditionals* in level 8.

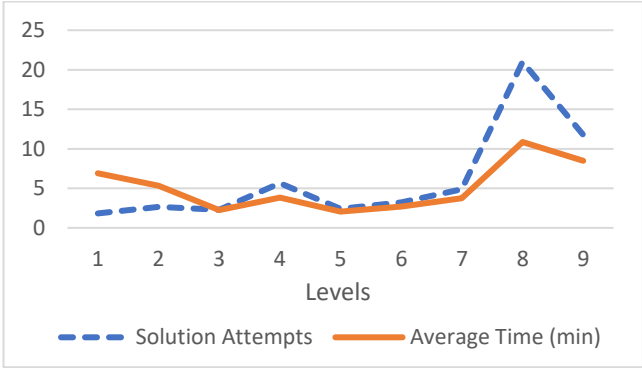


Figure 3. Solution Attempts and Average Time by Level

When comparing the performance by Gender, we found that the average Solution Attempts was greater for girls than for boys (M=6.48, SD=3.5, and M=5.93, SD=2.79, respectively). The Average Time was also greater for girls than for boys (M=3.17, SD=2.96, and M=2.87, SD=2.58, respectively).

4.1.2. Creative Thinking

As indicated above, Creative Thinking consisted of four dimensions (fluency, flexibility, originality, and elaboration). Based on normality tests (H.-Y. Kim, 2013), we assumed normality (Skewness<0.5 in absolute value) for

all dimensions of Creative Thinking except originality. A summary of the statistics is presented in Table 1.

We should comment on the relatively high mean value of originality (M=0.89, SD=0.16, N=174). Recall that we defined 59 categories of drawings for the TCTT – Figural Test. The distribution of the categories was in a "long tail" shape; that is, many categories had a very low frequency (i.e., were highly original), and only a few had relatively high frequency (i.e., were not original). The least original category ("Emoji") had a frequency of 75%.

Table 1. Descriptive Statistics for Creative Thinking

Variable	Average (SD)	Median	Skewness (SE)
Fluency	6.96 (3.65)	7	-0.23 (0.18)
Flexibility	4.25 (2.94)	4	0.48 (0.18)
Originality	0.89 (0.16)	0.94	-4.43 (0.18)
Elaboration	2.88 (0.89)	2.83	-0.12 (0.18)

4.1.3. Computational Creativity

Among all participants, the Computational Creativity score was low, as indicated by an average value of 0.24 (SD=0.24). No clear trend was observed throughout the game (See Table 2). In more than half of the cases, we could not assume normality (H.-Y. Kim, 2013), as can be seen from the high levels of the Skewness coefficients (that is, higher than 1). In most levels, one dominant solution was observed despite the existence of several others, as solved by a minority of students. Exceptions were levels 7 and 8, where only a single solution was observed for the whole population, probably because of the design of these levels and their block limit. Levels 4 and 6 showed the highest variability among participants.

Table 2. Descriptive Statistics for Computational Creativity

Level	Average (SD)	Median	Skewness (SE)
1	0.17 (0.25)	0.9	2.91 (0.18)
2	0.21 (0.27)	0.11	2.35 (0.19)
3	0.1 (0.2)	0.05	3.96 (0.18)
4	0.67 (0.19)	0.7	0.49 (0.19)
5	0.03 (0.13)	0.02	7.48 (0.18)
6	0.63 (0.17)	0.67	0.67 (0.19)
7	0.02 (0.72)	-	-
8	0.02 (0.09)	-	-
9	0.45 (0.15)	0.42	-1.78 (0.2)

4.2. Creative Thinking and the Acquisition of Computational Thinking

We tested the correlation between the Computational Thinking variables and the Creative Thinking variables. We found that Flexibility and Originality were significantly negatively correlated with Average Time, with Spearman's ρ taking values of -0.16 and -0.18, respectively, at $p<0.05$. Likewise, we found a significant negative correlation between Flexibility and Solution Attempts, with $\rho=-0.17$, at $p<0.05$. When we examined the correlation between the two variables by level, we found five cases – levels 1, 3, 5, 6, and 7 – which demonstrated significant correlations. Note that except for one case (level 1), all correlations were negative (findings are summarized in Table 3). These results indicate that **the more creative the students were** (as measured by a traditional creativity test), **the less time and effort it took them to solve the levels in the game**.

Table 3. Correlations between Computational Thinking and Creative Thinking by Levels (N=174)

	Solution Attempts	Correct Solution Attempts	Average Time
Fluency			
Level 1	ρ=-0.04 p=0.62	ρ=0.04 p=0.65	**ρ=-0.16***
Flexibility			
Level 1	ρ=-0.04 p=0.58	ρ=-0.01 p=0.94	**ρ=0.15***
Level 7	**ρ=-0.18***	ρ=0.00 p=0.96	ρ=-0.14 p=0.07
Originality			
Level 5	**ρ=-0.15***	ρ=-0.06 p=0.42	ρ=-0.04 p=0.62
Elaboration			
Level 1	ρ=0.1 p=0.19	ρ=-0.15 p=0.05	**ρ=-0.27****
Level 3	ρ=0.11 p=0.14	ρ=-0.15 p=0.05	**ρ=-0.19****
Level 6	**ρ=-0.2****	**ρ=-0.16***	**ρ=-0.21****

* $p<0.05$, ** $p<0.01$

4.3. Computational Creativity and the Acquisition of Computational Thinking

Next, we tested the associations between Computational Thinking and Computational Creativity as the latter is reflected by the originality of a correct solution in a given level compared with all other correct solutions. We did so both for the aggregated measures, as well as for each level of the game separately. We found that overall, Computational Creativity is negatively correlated with Solution Attempts, with $\rho=-0.17$, at $p<0.05$, and with Average Time, with $\rho=0.2$, at $p<0.01$. We also found four cases – levels 3, 4, 6, and 9 – which demonstrated

significant positive correlations, as reported in Table 4. These results indicate that **the more creative the students were in producing a solution**, **the more time and effort it took them to solve levels in the game**.

Table 4. Correlations between Computational Thinking and Computational Creativity by Levels (N=174)

	Solution Attempts	**Correct Solution Attempts**	**Average Time**
Level 3	ρ=0.14 p=0.08	ρ=0.05 p=0.53	**ρ=0.27****
Level 4	ρ=0.14 p=0.06	ρ=-0.02 p=0.78	**ρ=0.25****
Level 6	**ρ=0.17***	ρ-0.08 p=0.28	ρ=0.11 p=0.16
Level 9	**ρ=0.18***	ρ=0.1 p=0.27	**ρ=0.33****

* p<0.05, ** p<0.01

4.4. Computational Creativity and Creative Thinking

Finally, we examined the associations between creativity related measures: Computational Creativity and Creative Thinking. We found a **significant positive correlation between originality and the aggregated variable of Computational Creativity**, with ρ=0.2, at p<0.01. In addition, when examining these correlations between the variables for each level separately, we found that in one case – levels 6 – **originality was positively correlated with Computational Creativity, with** ρ=0.19, at p<0.05). These results indicate that **students who created more original drawings in the TTCT task were more creative in the game**.

5. DISCUSSION

Various studies have investigated the associations between computational thinking (CT) and creative thinking, however, this study is among the pioneers who examine these associations with Computational Creativity. In this study, we investigated the associations between the acquisition of CT by middle-school students who used a game-based learning platform, referring to two types of creativity – Creative Thinking and Computational Creativity. The first was defined by a traditional creativity test, not related to CT, while the second by the originality of correct solutions within the learning platform. Overall, we found interesting associations between the three research variables. Two dimensions of Creative Thinking—namely flexibility, and originality—were negatively correlated with measures of CT. As students were more creative in the TTCT task, they needed less time and effort to solve the levels in the game. This is in line with an earlier study that indicates a positive relationship between standardized creativity testing and students' performance (Anwar, Aness, Khizar, Naseer, & Muhammad, 2012). Furthermore, these findings reinforce the claim that creativity contributes to computer science and CT in particular (Kong, 2019; Miller et al., 2013).

Notably, we found that at some level of the game, there was a positive correlation between Computational Creativity and measures of the acquisition of CT. That is, students who provided more unique and original solutions needed more time and attempts to solve these levels. This is not surprising as producing a creative solution may take more time than a "standard" solution (Akinboye, 1982; M. Baer & Oldham, 2006).

We also found some intriguing associations between the two types of creativity. Computational Creativity was positively correlated with the originality dimensions of Creative Thinking. These results may imply that creativity is context-dependent (as the associations were only demonstrated in some of the game-levels) as well as transferable from one domain to another. This supports the hierarchical model of creativity, which integrates both domain-general and domain-specific types of creativity (Baer, 2010; Hong & Milgram, 2010). It also reflects earlier findings that linking TTCT score and creativity in problem-solving in programming platforms (Liu & Lu, 2002).

While the results and insights of this study contribute in offering a better understanding of the associations between CT and type of creativity, we also want to highlight its limitations. First, we analyzed data from a single learning platform (Kodetu), and it is possible that our findings were a result of some unique characteristics of this platform. Specifically, the studied platform does not encourage multiple correct solutions and, in some cases, limits the free use of coding blocks, which may affect and limit creative submission. Furthermore, the analysis is based on students from a single country (Spain). Personal and cultural characteristics may impact the way creativity is exhibited. Therefore, we plan to broaden our perspective by examining similar platforms under different conditions and with a more multi-cultural view.

6. REFERENCES

Akinboye, J. O. (1982). Correlates of Testing Time, Age and Sex in the Nigerians' Performance on the Torrance Test of Creativity. *Journal of Psychological Researches*, *26*(1), 1–5.

Anwar, M. N., Aness, M., Khizar, A., Naseer, M., & Muhammad, G. (2012). Relationship of Creative Thinking with the Academic Achievements of Secondary School Students. *International Interdisciplinary Journal of Education*, *1*(3), 1–4.

Baer, J. (2010). Is Creativity Domain Specific? In J. C. Kaufman & R. J. Sternberg (Eds.), *The Cambridge Handbook of Creativity* (pp. 321–341). New York, NY: Cambridge University Press.

Baer, M., & Oldham, G. R. (2006). The Curvilinear Relation between Experienced Creative Time Pressure and Creativity: Moderating Effects of Openness to Experience and Support for Creativity. *Journal of Applied Psychology*, *91*(4), 963–970.

Beghetto, R. A. (2010). Creativity in the Classroom. In Kaufman, J. C. & R. J. Sternberg (Eds.), *The Cambridge Handbook of Creativity* (pp. 447–463). New York, NY: Cambridge University Press.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational

Thinking through Programming. *Review of Educational Research*, *87*(4), 834–860.

Czerkawski, B. (2015). Computational Thinking in Virtual Learning Environments. *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*. Kona, Hawaii, United States: Association for the Advancement of Computing in Education (AACE), 1227-1231.

Guilford, J. P. (1950). Creativity. *The American Psychologist*, *5*(9), 444–454.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A Multidisciplinary Approach towards Computational Thinking for Science Majors. *ACM SIGCSE Bulletin*, *41*(1), 183.

Hong, E., & Milgram, R. M. (2010). Creative Thinking Ability: Domain Generality and Specificity. *Creativity Research Journal*, *22*(3), 272–287.

Ibanez, M.-B., Di-Serio, A., & Delgado-Kloos, C. (2014). Gamification for Engaging Computer Science Students in Learning Activities: A Case Study. *IEEE Transactions on Learning Technologies*, *7*(3), 291–301.

ISTE. (2017). Turn Coders into Computational Thinkers. Retrieved July 30, 2017, from https://www.iste.org/explore/articleDetail?articleid=936&category=Innovator-solutions&article=Turn+coders+into+computational+thinkers

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Modern Computing*, *4*(3), 583–596.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning Programming at the Computational Thinking Level via Digital Game-play. *Procedia Computer Science*, *9*(0), 522–531.

Kim, A. S., & Ko, A. J. (2017). A Pedagogical Analysis of Online Coding Tutorials. In M. E. Caspersen, S. H. Edwards, T. Barnes, & D. D. Garcia (Eds.), *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17*. New York, NY: ACM, 321–326.

Kim, H.Y. (2013). Statistical Notes for Clinical Researchers: Assessing Normal Distribution (2) Using Skewness and Kurtosis. *Restorative Dentistry & Endodontics*, *38*(1), 52–54.

Kong, S. (2019). Components and Methods of Evaluating Computational Thinking for Fostering Creative Problem-solvers in Senior Primary School Education. In S. Kong & H. Abelson (Eds.), *Computational Thinking Education*, 119–142. Singapore: Springer.

Liu, M.C., & Lu, H.F. (2002). *A Study on the Creative Problem-solving Process in Computer Programming*. Paper presented at the International Conference on Engineering Education, Manchester, UK.

Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., Ramsay, S., & Hazley, M. P. (2013). Improving Learning of Computational Thinking Using Creative Thinking Exercises in CS-1 Computer Science Courses. *Proceedings of Frontiers in Education Conference, FIE*, 1426–1432.

Navarrete, C. C. (2013). Creative Thinking in Digital Game Design and Development: A Case Study. *Computer Education*, *69*, 320–331.

Reiter-Palmon, R., Illies, M. Y., Kobe Cross, L., Buboltz, C., & Nimps, T. (2009). Creativity and Domain Specificity: The Effect of Task Type on Multiple Indexes of Creative Problem-solving. *Psychology of Aesthetics, Creativity, and the Arts*, *3*(2), 73–80.

Said-Metwaly, S., Noortgate, W. Van den, & Kyndt, E. (2017). Methodological Issues in Measuring Creativity: A Systematic Literature Review. *Creativity. Theories – Research - Applications*, *4*(2), 276–301.

Seo, Y.-H. & Kim, J.-H. (2016). Analyzing the Effects of Coding Education through Pair Programming for the Computational Thinking and Creativity of Elementary School Students. *Indian Journal of Science and Technology*, *9*(46), 1–5.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying Computational Thinking. *Educational Research Review*, *22*, 142-158 .

Torrance, E. P. (1965). Scientific Views of Creativity and Factors Affecting its Growth. *Daedalus*, *94*(3), 663–681.

Torrance, E. P. (1974). *Torrance Tests of Creative Thinking*. Bensenville, IL: Scholastic Testing Service.

Vu, P. & Feinstein, S. (2017). An Exploratory Multiple Case Study about Using Game-based Learning in STEM Classrooms. *International Journal of Research in Education and Science*, 582–582.

Vygotsky, L. S. (2004). Imagination and Creativity in Childhood. *Journal of Russian and East European Psychology*, *42*(1), 7–97.

World Economic Forum. (2015). *New Vision for education unlocking the potential of technology*. Geneva, Switzerland: World Economic Forum.

Yadav, A. & Cooper, S. (2017). Fostering Creativity through Computing. *Communications of the ACM*, *60*(2), 31–33.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Towards using Computational Modeling in learning of Physical Computing – An Observational Study in Singapore Schools

Peter, Sen-Kee SEOW[1], Bimlesh WADHWA[2], Zhao-Xiong LIM[3], Chee-Kit LOOI[4]
[2]National University of Singapore, Singapore
[1,3,4] National Institute of Education, Nanyang Technological University, Singapore
peter.seow@nie.edu.sg, bimlesh@nus.edu.sg, zhaoxiong.lim@nie.edu.sg, cheekit.looi@nie.edu.sg

ABSTRACT
Coding for students is no longer just constrained to software and screen-based text and graphics. Students today use programmable sensors and microprocessors to solve the problems around them. The purpose of this research is to understand how students conceptualize problems and implement solutions with physical computing. Our study is driven by the following: 1) find out what Computational Thinking (CT) competencies, specifically abstraction, decomposition and algorithmic thinking, can be developed by students and 2) to what level students develop these competencies in carrying out physical computing projects. We closely observe how 41 Grade 7 students developed solutions for problems they identify in the physical world around them. Through doing so, we explore how powerful ideas of CT play a role in a project-approach to physical computing. We believe open-ended exploration through a project-approach in physical computing should reinforce practices where CT skills can grow and flourish. Our findings show that much of students' interaction with sensors and devices is at *pre-CT* level, where students simply use pre-existing code fragments or templates. As students gain skills and confidence, they can be explicitly guided to develop CT skills with new projects of their own design justifying their choices. We strongly believe that Computational Modeling (CM) could help students develop their CT skills e.g. abstraction, decomposition, and algorithmic approach much more than the minimally guided syntax driven teaching approaches.

KEYWORDS
computational thinking, computational models, abstraction, physical computing, K-12

1. INTRODUCTION
Physical computing, an emerging approach to learning computing, teaches students about coding and computational thinking through hands-on activities with sensors using small computing boards like the micro:bit (Rogers, et al., 2017). In Singapore, primary school students are introduced to coding through the Digital Maker programme with the micro:bit (TNP, 2019).

The micro:bit is a pocket-sized physical computing device that can be input with various codes. The device is designed to be visually appealing and tactile, affordable, easy to use, interactive, and extensible. The board has a built-in display, buttons, motion detection, temperature and light sensing. It can be programmed using a desktop PC, laptop or tablet running one of several different operating system web-based programming environments: a visual block-based editor, Python or JavaScript.

Despite the ease in using the micro:bit to code, it is not certain that physical computing will actually improve students' understanding of computational thinking (CT). It is therefore important for educators to explore the question on "what and how do students develop CT competencies when they use physical computing devices to interact with the physical world?" With this as an over-arching research question, we set out to design our observational study in Singapore schools. Our aim here is to 1) find out what CT competencies, specifically abstraction, decomposition and algorithmic thinking, should be developed by students and 2) to what level, do students develop these in carrying out physical computing projects. Furthermore, we want to find out if students used any conceptual or computational modeling before or while carrying out their physical computing projects. We believe our observations would help us partially answer our over-arching research question.

The project-approach to physical computing, an often used pedagogy in schools, serves as an open-ended exploratory approach to examine the computational thinking competencies that students should learn. We observed that among many other factors that inhibit the development of CT skills, the inherent complexity of problem and solution space could overwhelm students. Additionally, the cognitive load in designing and developing their solutions could also hinder them in their learning. Therefore, we propose gentle scaffolds for developing a sound conceptual model, followed by guided Computational Modeling (CM) for overall CT skills development.

2. RELATED WORK
Our study is informed by the ideas of Computational Thinking (Papert, 1972; Wing, 2008), Computational Models (Aho, 2012; Denning, 2017; Calder, et. al., 2018), learning computing models (Sentence, Waite & Kallia, 2019) and Physical Computing (O'Sullivan & Igoe, 2004). Seymour Papert (1972) described CT as a mental skill children develop from practicing programming. In a 2006 paper, Jeannette Wing (2006) catalyzed a 'CT for all' (p. 33) movement. It has been debated since then if CT makes better problem solvers or if practice of coding can help develop CT skills, with claims that everyone can benefit by CT not yet being fully substantiated by studies (Guzdial, Kay, Norris, Soloway, 2019). Many definitions and role of CT in computing as well as in other fields, and overlap of CT and computing have followed.

Denning (2017), in his viewpoint on CT, finds the absence of computation models in the post-2006 CT definitions as a mistake. He points that key ingredients of CT e.g. abstraction, data representations and decomposition are used, in order to get a model to accomplish certain task. He encourages teachers to take note of Aho's reflection about

computation as "a process defined in terms of an underlying model of computation" (p. 832) and CT as "the thought processes involved in formulating problems so that their solutions can be represented as computational steps" (p. 832). Aho suggests the use of the term 'computation' in conjunction with a well-defined model whose semantics is clear and which matches the problem being investigated. He added that one could use CT skills to devise computation models.

There is a growing emphasis on teaching computing since the idea of CT was proposed by Wing. Countries such as the United Kingdom have mandated the integration of Computing and Computational Thinking into the National curriculum at all levels. Japan is making learning computing compulsory for elementary and higher education. Introducing computing has expanded from using screen-based tools such as Scratch to physical computing such as the micro:bit. In physical computing, students interact with the world through the use of sensors as input and controllers as output of computing devices. Computation is done on the data from the input sensors like buttons or temperature sensor to drive the controllers such as motors or LED lights.

For students learning to code, they need to deal with the complexity of knowing what data they need from the environment, how to use the sensors to collect the data, how the data is used for computation, what output needs to be created and how it should be used. This complexity could overwhelm students in designing and developing their solutions. In many approaches to introducing coding, students are taught using physical computing without introducing CT skills. The assumption is that students would learn CT skills as a result of the learning coding through physical computing. In the *pre-CT* stage, students may encounter difficulties in implementing physical computing solutions without using CT skills, such as automating machine to interact with the physical world (Fig 1). For example, students need to know how to acquire data from the environment, process to compute the data, and output to the world. Teaching CT skills explicitly to students can help students to implement their solutions better as shown in Fig 2. The teaching of CT skills can bridge some of the difficulties students face in learning coding and implementing solutions with Physical computing.

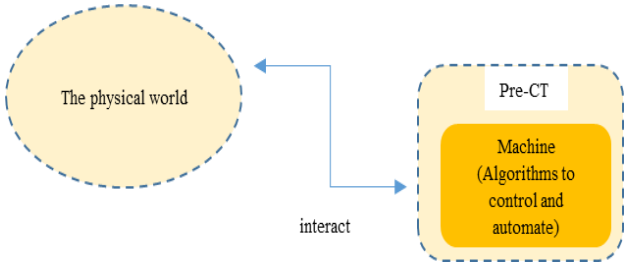


Figure 1. Illustration of the difficulties of implementing Physical Computing in the classroom without involvement of CT

PRIMM (Sentence, Waite, & Kallia, 2019), a framework for teaching programming, focuses on students talking about how and why programs work before they tackle writing their own programs. The first element of PRIMM i.e. Predict, is about students discussing and predicting what a program might do, drawing and writing out what they think will be the output in order to develop the vocabulary they need to talk about the program. We believe such vocabulary development should extend to CM, which is an important step in

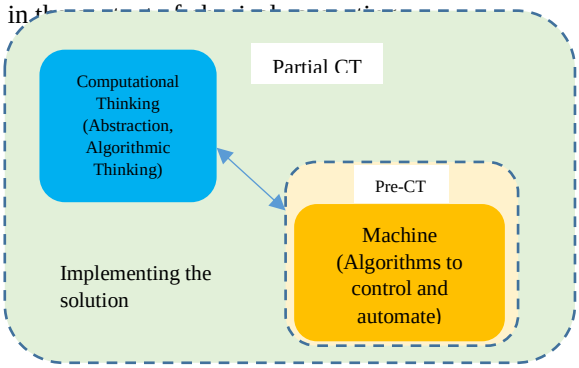


Figure 2. Illustration on how the gap between and CT and implementing Physical Computing in the classroom is being bridged

3. CONTEXT OF STUDY

To understand how CT is applied in learning of physical computing through the use of micro:bit in coding, the research team observed the micro:bit training sessions and interdisciplinary project work lessons of Grade 7 comprising of 41 students divided into 10 groups, in a local neighbourhood school, over a period of 4 ½ months. The purpose of the study was to understand how students conceptualised the problems and implemented the solutions with physical computing. Additionally, the research aimed to 1) find out what CT competencies, specifically abstraction, decomposition and algorithmic thinking, students should develop and 2) to what level, do students develop these in carrying out physical computing projects. Students find it exciting when they see their projects come to life. Physical computing is therefore very engaging that helps them understand how things work in the real world.

The students followed a project-approach to develop physical computing solutions using micro:bit. During the micro:bit training sessions, the students are first introduced to both the basic and intermediate technical aspects of the micro:bit board and makecode editor, where they carry out block coding. Thereafter, they are introduced to the development environment of the 'makecode editor', an online visual block-based coding programme, where they could develop their codes. Their solutions were expected to incorporate sensors to capture data occurring from everyday phenomena such as surrounding temperature or sound. The process that students undertake in designing their projects using the below-mentioned flowchart.

The entire training sessions and interdisciplinary project work lessons seek to complement the Applied Learning Programme (ALP) in Robotics and Programming run by the school, which aims to empower students with the technological and thinking skills that will enable them to be innovative and empathetic members of the community. (MOE, 2019)

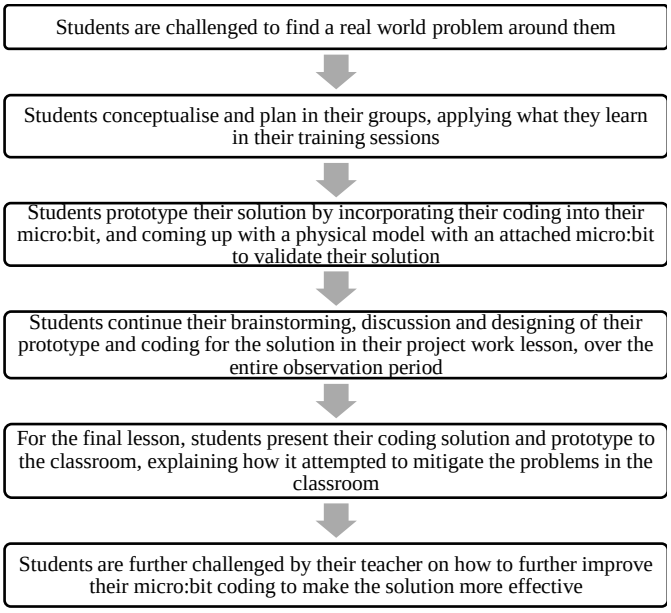


Figure 3. Flowchart showing the process students undertake

4. METHODOLOGY AND DATA COLLECTION

The students attended the two micro:bit training sessions (10 hours in total), and attempted an implementation of the design thinking approach where they developed prototypes with the micro:bit sensors, servo motors and connecting wires. Students worked within their groups for a duration of 12 weeks, typically meeting once every 2 weeks (see Table 1).

Table 1. Table of lesson observation schedule, data collected, summary of training session and lessons

Lesson	Description of Activity	Data Collected in lessons
1	Overview Project Work; Group Discussion - Identify Project Topic	Video
2	Introduction to micro:bit; Group Discussion- Project work Topic	Video
3	micro:bit Training (Day 1)	Video
4	micro:bit Training (Day 2)	Video
5	Completion of CT Questionnaire Student group discussion on Project (with students)	Video
6	micro:bit Revision by Trainer Student group discussion on Project (with Teacher and Trainer)	Video, Audio
7	Discussion on Project (with Teacher and Trainer)	Video, Audio
8	Presentation of Projects	Video, Audio
9	Interviews with Students	Video, Audio

The researchers observed the 10 hours training sessions, in order to accustom with the coding curriculum that students were being taught. They sought to understand the thought and application processes of students when they were incorporating coding knowledge and subject content knowledge into the various projects they were doing. The sessions were filmed and recorded in both audio and video format.

Additionally, we observed the students as they design, code and implement their physical computing projects in Lessons 5 to 7 (see Table 1). We looked out for CT skills application in specific milestones of problem formulation, initial design, implementation, and demonstration as they carry out their projects. The purpose of the activity was to understand how students use abstraction, decomposition and algorithmic thinking, while conceptualising the problems and implementing the solutions with physical computing.

For data collection, we selected two groups for more detailed observation (See Table 2). We followed these two groups as they developed their projects and enquired them on their actions and decisions. These two groups were selected based on the complexity of their projects and recommendations by their teachers because they were able to articulate their ideas clearly compared to their peers. We recorded the presentations they made to classmates on their ideas and solutions. After their presentation, we interviewed the group members and archived their codes for analysis.

Table 2. Projects of the two groups of students observed

Group No	No of Students	Project
A	4	Classroom Door Lock
B	4	Noise Level Detector

5. ANALYSIS AND FINDINGS

To evaluate students on their application of their CT skills, we developed a set of rubrics for CT skills. The rubrics was developed from our literature review of the CT skills and we also obtained feedback from practitioners on the levels of the rubrics and the skills. For this work, we focused in observing three CT skills, namely Abstraction, Decomposition, and Algorithmic Thinking (See Table 3).

In our analysis, we observed the two groups closely as they developed their projects with physical computing. Our observations of two groups and the projects that they worked on in the following paragraphs.

Group A worked on a problem of a sensor-operated door-lock that would open upon motion detection of a contactless card. The students were queried about the algorithmic steps and meaning of the codes in the micro:bit block. The research team hinted to students about thinking logically about the codes found in the block and figure out which blocks can be matched together to form the required codes. We observed that the students lacked the necessary knowledge on the type of data and sensor to read the contactless card. As a result, the students realized that they had to change the sensor from a card scanner to a digital keypad with numbers connected to the classroom door, as the use of a card scanner had been deemed unfeasible. Even with the change, they could not implement the use a digital keypad with the micro:bit.

Table 3. Rubrics for Classroom Observation

ABSTRACTION – to choose the right amount of detail for the problem to be modeled
Beginner: Able to identify and choose relevant data and information for the model and solving the problem Intermediate: Beginner + identify relevant data and from **multiple sources** to **integrate** for developing possible CMs Advanced: Intermediate + **physically represent through modelling and interact with** relevant data and information for the

model from multiple sources + express/articulate what is conceptualized in thinking by constructing a model using relevant details + translate abstraction into model
DECOMPOSITION – to breakdown a problem into component parts to be understood and solved
Beginner: Able to **break a problem** in smaller parts Intermediate: Beginner + articulate the **relationship** between the parts Advanced: Intermediate + **develop a model** to **understand** the complex system to **facilitate/evaluate** problem solving using computation
ALGORITHMIC THINKING – to think in terms of sequences to solving the problem
Beginner: **Create a sequence of steps** to solve a problem, with instructions to execute Intermediate: Beginner + Understand how **automation** works and use algorithmic thinking to create a sequence of steps Advanced: Intermediate + test the automated steps through a breaking down process + identifying how the information changes between the steps and refine/optimise the steps + **improving** the creation of sequence of steps in areas such as optimisation, efficiency, reusability, readability and re-factoring

Group B worked on the problem of noise detection in the classroom, which would send an alert to the teacher once the noise threshold is reached. The team had difficulty in conceptualizing their solution with the micro:bit. They recognized though that they needed a sound sensor to detect sound from the physical environment. They were able to test the sensor input and simulating an alert to the teacher by pairing two micro:bits. They however lacked the systemic knowledge. For example, they did not think deep enough on where to best place the sensor to capture the noise accurately or how sound travels could affect the coding and prototyping of their project. It showed an inadequate mental model, and therefore an inadequate conceptual model of the problem and solution i.e. how the sensors interact with the physical world, and e.g. sound travels by waves and where they place sensors matters.

We analysed the transcripts of the interview made with the students to evaluate the decisions the made with regard to the implementation of their solution. We identified how their abstraction, decomposition and algorithmic thinking are demonstrated from the interview data, as explained by the students (See Table 4). This was made in reference to the rubrics we developed.

Table 4. Interview Transcript of students in Group A demonstrating the skill of Algorithmic Thinking

R *[Researcher]* **S** *[Student]*	**Dialogue**	**Explanation**
R	How did you solve the problem?	
S [06:38]	We decided to place, instead of alerting the teacher when it hits the second level, we decided to show the teacher the level all the time	Students decide to give remote micro:bit to teacher indication of real time noise level Abstraction – choose the relevant data Decomposition – break down the problem into smaller part to show the information to the teacher
R	Oh show, show the teacher the level all the time is it? How do you send the teacher the value?	
S	We use the radio function we send the current noise level	
R	Yes	
S	And when the teacher receives it, it'll show on the screen	Abstraction - choose the relevant data and information

The codes developed for the solution and the created artefact comprising of the micro:bit board with sensors were analysed for students algorithmic thinking, decomposition and abstraction competencies (See Figure 4). For example, the group with the sound sensor used one micro:bit to read the sound level from Pin 1 and control a LED at Pin 0. The micro:bit will send the value of the sound level to another micro:bit through radio. The students designed such that the remote micro:bit will be carried by the teacher and will notify the teacher if the level exceeds the noise. The students calibrated and tested the actual noise level in their classroom that they deemed as noisy. This was the level they chose as the trigger to notify the teacher. During testing, the students noticed that there was a delay in sending the value to the teacher's micro:bit and the noise level reading was not accurate. They did not have time to solve these issues.

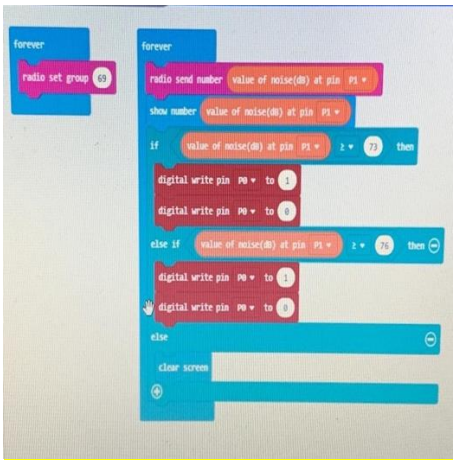


Figure 4. Example of code (Sound Sensor) done by students using makecode editor in the project design

We analysed the groups' work process in developing the solution, their completed artefact solution and codes, and the interview transcripts. The analysis from the sources were triangulated and compared to the rubrics we developed. Results showed that most were at best able to achieve only the beginner level of the CT skills (see Table 5). However, we noted that the acquiring of these skills progressed over time and towards the end, most students became more competent in them as they engaged in more programming.

From our observations, we posit that students have difficulties in designing a computing solution due to their rudimentary CT skills. At the end, the students managed to build a prototype of their solution but experienced challenges in abstracting the vital data required for the solution in the initial stages. This affected their choice of sensors to use as input to their solution and difficulties in

thinking algorithmically on the computation to obtain the automated output. Referring to Table 4, we believe both groups worked at *pre-CT* stage (See Figure 1). The above observations are specific to a few projects and students, and generalizations would require more studies with more students and diverse settings.

Table 5. Results of skills demonstrated by the two groups during the observation

CT skills	**Group A (Level achieved)**	**Group B (Level achieved)**
Abstraction	Beginner	Intermediate
Decomposition	Beginner	Beginner
Algorithmic Thinking	Beginner (students who programmed)	Beginner (students who programmed)

6. DISCUSSION AND RECOMMENDATIONS

Computational Modeling (CM) as per one of the established definitions (Calder, et. al., 2018) can help us "translate observations into an anticipation of future events, act as a testbed for ideas, extract value from data and ask questions about behaviours. The answers are then used to understand, design, manage and predict the workings of complex systems and processes, from public policy to autonomous systems." (p. 2)

Computational Thinking (CT), on the other hand is a generalized problem-solving process that can be applied to a wide variety of problems. The steps of CT includes formulating a problem in a way that enables us to use a machine to solve it. The machine here refers to computer and other devices. In the process, data and concepts are abstracted and analysed and algorithms are developed for automating a solution.

We believe, that CT definition is not explicit about modelling, i.e. representation of abstracted data and concepts before algorithms are developed. Here in this study, we first observe if indeed classrooms have modelling incorporated in the CT lessons. Our findings show that students attempt to directly code or develop algorithm once they have understood the problem. They do not develop any models or use any tools to represent data or concepts.

Today, visual block-based visual programming platforms such as Scratch, Blockly are popular vehicles for programming sensors and delivering CT. Even though students are quick to pick up the programming constructs, conceptual difficulties pertaining to problem and solution space, and developing CT skills e.g. abstraction, decomposition and algorithmic skills, are often evident.

From our observation of the work of both groups, we surmise that students face difficulties in designing a computing solution due to the missing explicit CT exposure and almost non-existent CM. They have challenges in abstracting the vital data required for the solution and thinking algorithmically on the computation to obtain the desired output.

For students, owning an idea serves as a motivation to learn. We observed that students identified a problem or innovation they wanted to pursue. We found that though students were engaged in the maker-rich environments, they did not move to thinking computationally and solving problems. Much of their interaction with sensors and devices is superficial. This is inferred through our interactions with students. When we discussed with students about for example how sensors were working or how transmission of data or signal was from one to another device, we did not find them confident of their knowledge of hardware beyond what they were using it for.

However, when we, for example, introduced input-process-output model, their understanding of the project seemed better. They could explain the project better to another researcher from our own group as well as to their teachers later. We would want them to develop higher order design skills through physical computing, not just coding. They should understand the iterative nature of finding a solution and testing. Open-ended exploration through the project-approach in physical computing should reinforce practices where CT skills grow and flourish.

Additionally, based on our observations, students have difficulty in starting the implementation due to their lack of CT and CM in the *pre-CT* stage. Reasons for such gaps are mainly due to the inherent complexity of the problem that they are trying to solve, as well as integrating different components of the solution involving use of sensors, collection of data, computation of data and automating the solution. To scaffold their learning, it is important that they are guided through developing a conceptual model, such as CT (abstraction and algorithmic skills) and CM (See Figure 4), leading them eventually towards the CT+CM stage.

We propose that a CM phase could act as a glue from understanding problem to the coding activity (see Figure 4) for students to formulate their problems in computational steps (Denning, 2017). Execution of computational models could be seen as controlling and automating the machine to solve the problem computationally. We believe focused modeling activities could help students develop their CT skills e.g. abstraction, decomposition, and algorithmic approach much more than the minimally guided syntax driven teaching approaches.

Based on our observations we suggest concrete steps that can be taken to support the development of computational thinking. We believe a project-approach through physical computing provides an excellent maker-platform, in which students are provided with the opportunity to evaluate and manipulate underlying abstractions and mechanisms. It gives ample scope of developing CT skills namely abstraction, decomposition, and algorithmic thinking.

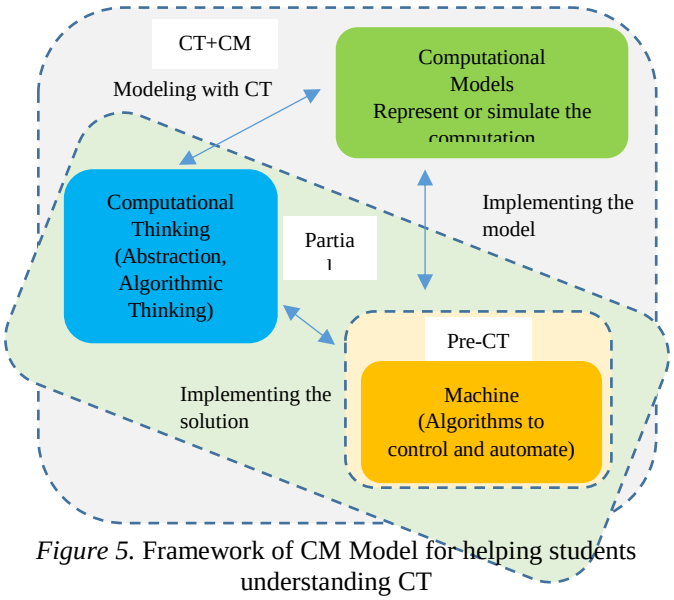


Figure 5. Framework of CM Model for helping students understanding CT

We propose a triad-model for effective and systematic development of CT skills. This model describes a pattern of engagement (see Figure 5). It is based on the premise that computational modeling promotes the acquisition and development of CT among students. At the *'pre-CT'* level, students are simply coders. For example, they code using pre-existing code fragments or templates, and acquire coding skills through a series of iterative refinements. New skills and understandings are developed over time and they begin to code with increasing levels of sophistication. This does develop an understanding of at least a subset of the abstraction contained within the problem and solution. We observed that most of the students in our study operate at the *pre-CT* level. As students gain skills and confidence, they can be explicitly guided to develop CT skills with new projects of their own design justifying their choices. At this *'CT'* level, all three key aspects of computational thinking: abstraction, decomposition and algorithmic thinking, come into play. We observed two groups of students partially acquiring CT skills when we explicitly made them think about specific issues about their problem or solution. We strongly believe that Computational Modeling (CM) could act as an effective medium to develop computational thinking skills especially in the context of physical computing. We propose another level in our framework labelled *'CT+CM'* i.e. using Computational Modeling (CM) as a medium to develop CT skills.

Here are our recommendations for effective delivery of CT skills with CM based on this study:

- Design of a thinking style workshop that could help students to develop and strengthen their mental model about the problem. It is an important and essential that students have the relevant vocabulary of the problem domain, and systemic thinking before attempting to formulate a solution.
- Guided team brainstorming sessions could help students develop conceptual models for the solution they propose. Developing a sound conceptual model at the team level could help each individual member to strengthen his/her mental model, and sync well with team before implementing the solution.
- Gentle scaffolds could be introduced, e.g. graphic organisers for the above, to ease students into developing Computation Models. CM could be the glue that connects a conceptual solution and actual code.

7. REFERENCES

Aho. A. V. (2012). Computation and Computational Thinking. *The Computer Journal, 55*, 7, 832–835.

Calder, M., Craig, C., Culley, D., de Cani, R., Donnelly, C. A., Douglas, R., ... & Hinds, D. (2018). Computational modelling for decision-making: where, why, what, who and how. *Royal Society open science*, *5*(6), 172096.

CSTA. (2013). *Bugs in the System: Computer science teacher certification in the U.S.* Retrieved December 9, 2019, from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Communications of the ACM*, *60*(6), 33–39.

Guzdial, M., Kay, A., Norris, C., & Soloway, E. (2019) Computational Thinking should just be Good Thinking. *Communications of the ACM, 62*(11), 28-30.

MOE. (2019). *Secondary School Education, Applied Learning Programme*. Retrieved December 9, 2019, from https://beta.moe.gov.sg/uploads/Secondary-School-Education-Booklet-2019.pdf

O'Sullivan, D., & Igoe, T. (2004). *Physical computing: sensing and controlling the physical world with computers*. Course Technology Press.

Papert, S. (1972). Teaching Children Thinking. *Programmed Learning and Educational Technology*, *9*(5), 245-255.

Rogers, Y. et al. (2017). From the BBC micro to micro:bit and Beyond. *interactions 24(*2), 74–77.

Sentence, S. et al. (2017). "Creating Cool Stuff – Pupils' experience of the BBC micro:bit. *In Proceedings of the 48th ACM Technical Symposium on Computer Science Education: SIGCSE 2017*. 531-536.

Sentence, S., Waite, J. and Kallia, M. (2019). Teaching Computer Programming with PRIMM: A Sociocultural Perspective. *Computer Science Education*, *29*(2-3), 136-176.

TNP (2019). *Coding enrichment classes for all upper primary school students from next year*. Retrieved December 11, 2019, from https://www.todayonline.com/singapore/coding-enrichment-classes-upper-primary-school-students-next-year

Wing, J. (2008). Computational Thinking and Thinking about Computing. *Phil. Trans. R. Soc. A*, *366*(1881), 3717–3725.

Wing, J. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33-35.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Computational Thinkers: Contemporary Approaches and Directions in Computational Thinking for K-12 Education

Steven FLOYD
Western University, Canada
spfloyd@uwo.ca

ABSTRACT
This paper explores contemporary researchers and their approaches to computational thinking (CT) for students in K-12 education. Computational Thinking (Wing, 2016) is used as a focal point of investigation as the views of Barba, Papert, Resnick, Kafai, diSessa, Denning, Aho, Wilkerson, and Grover are compared. While the varied approaches to CT may indicate disagreement on behalf of researchers in the field, it can also be a sign of the varied directions in which CT, and related concepts, can be taken. As educational jurisdictions integrate CT in K-12 curriculum, these approaches and directions should be considered by educators, policy makers, and researchers alike.

KEYWORDS
computational thinking, computer science, education, K-12, K-8

1. INTRODUCTION
In March of 2006, the Communications of the ACM published Jeanette Wing's Computational Thinking where she was seeking to expand the scope of Computer Science education beyond the post-secondary levels. Wing articulated the characteristics and importance of a "universally applicable attitude and skill set" (Wing, 2006, p. 33) called Computational Thinking (CT) which involved thinking like a computer scientist.

The article captured the imagination of educators and researchers from around the world (Grover & Pea, 2017) and according to Google Scholar, as of December 2019, had been cited over 5475 times. Important to note; however, is the fact that ideas surrounding the integration of computer science (CS) concepts and thought processes in K-12 education did not begin, and certainly did not end, with Wing's seminal work. A long history exists related to the integration of CS concepts in K-12 education and since 2006, many researchers have expanded on the definition and scope of CT, and its role in K-12 education.

What follows is a summary of the field of CT that approaches the subject by presenting the various thought leaders and their ideas. These ideas are especially pertinent at this time as educational jurisdictions around the world explore the integration of CT in the K-12 grades.

2. THINKING LIKE A COMPUTER SCIENTIST
Jeanette Wing initially defined CT as "solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Later, she refined her definition to the "thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.". While researchers have discussed Wing's initial definition at length, a primary criticism surrounds the idea of thinking like a computer scientist.

In Computational Thinking: I do not think it means what you think it means (2016), Lorena Barba explains that Wing's view fails to acknowledge CT as "a source of power to do something and figure things out, in a dance between the computer and our thoughts". Viewing the computer as a formal tool to understand, and then apply to a problem later, takes away its power: "Most people don't want to be a computer scientist, but everyone can use computers as an extension of our minds, to experience the world and create things that matter to us". Barba was attempting to move discussions away from Wing's CT, towards an idea that would allow students to use computing as a means to create new knowledge in a broad number of domains. In order to support this view, Barba made reference to several of Seymour Papert's ideas.

3. CONSTRUCTIONISM AND LOGO PROGRAMMING
Described as the father of educational computing (Stager, 2016), Papert laid the foundation for how we think about learning and teaching with computers (Kafai & Burke, 2014).

Before arriving at MIT in 1963, Papert worked closely with, and was heavily influenced by, Jean Piaget and his theory of cognitive development called constructivism. Papert built on Piaget's ideas by developing his own theory of learning that he called constructionism (Stager, 2016). While both theories focus on learning being an active process of constructing knowledge, and both include the idea that children learn new concepts by relating them to things that they already know (Ames, 2018), they differ in that constructionism acknowledges the importance of culture as the source of the materials that students will use to build their knowledge (Papert, 1993). Papert believed that in some cases the culture provides the learning materials in abundance, which facilitates Piagetian learning. In other cases; when here is a slower development of a concept, Papert saw the "critical factor as the relative poverty of the culture in those materials that would make the concepts simple and concrete" (Papert, 1993, p. 7).

It was for this reason that Papert was so enamoured with the computer as a learning tool. He felt that the relative poverty of a culture could be cured by a computer, the Proteus of machines, that can "take on a thousand forms and can serve a thousand functions" (Papert, 1993, p. xxi).

At MIT, Papert developed the Logo programming language, which he felt could alter the relationship that students had

with computers. Rather than having students be programmed by a computer (through computer-based exercises and feedback) the Logo programming environment reversed this relationship by having the student program the computer itself, which essentially meant teaching the computer how to think.

Papert uses the term "mechanical thinking" to describe the type of thinking that students are introduced to when programming in Logo (Papert, 1993, p. 27). He emphasises that by introducing students to mechanical thinking, they suddenly become aware of thinking styles, and they begin to consider other thinking styles that might exist, as well as how and why they might choose between styles. Later, Papert uses the term "computational thinking" when describing what some of his experiments were trying to integrate into everyday life. He acknowledges that the visions of these experiments were insufficiently developed, but that they will serve as "manifestations of a social movement of people interested in personal computation, interested in their own children, and interested in education" (Papert, 1993, p. 182)

Papert's work surrounding computers and education, and his development of the Logo programming language, sowed the seeds of this social movement. In 2017, Mitch Resnick, a former doctoral student of Papert's, exclaimed "I will be happy to spend the rest of my life working to nurture the seeds that Seymour sowed" (Resnick, 2017).

4. COMPUTATIONAL FLUENCY AND SCRATCH PROGRAMMING

Resnick is the director of the Lifelong Kindergarten research group at MIT that developed Scratch, the world's leading coding platform for kids. Scratch was deeply inspired by Papert's Logo but "goes beyond Logo by making programming more tinkerable, more meaningful, and more social" (Resnick, 2014, p. 2).

In New Frameworks for Studying and Assessing the Development of Computational Thinking (2012), Resnick, along with co-author Karen Brennan, propose an alternate CT framework that includes three key dimensions: concepts, practices and perspectives.

Resnick and Brennan's CT concepts include the concepts that designers engage in as they program (sequences, loops, parallelism, events, conditionals, operators, and data). CT practices differ to CT concepts in that the practices describe the processes of construction that student engage in while creating Scratch projects (being incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing). Finally, CT perspectives, describe the evolving understanding that students using Scratch exhibit about themselves, their relationship to others, and the technological world (expressing, connecting, and questioning). Together, the concepts, practices and perspectives provide a broader understanding of CT that Resnick calls Computational Fluency.

The impetus for Resnick's Computational Fluency was an attempt to "highlight the importance of children developing as computational creators as well as computational thinkers" (Resnick, 2018, p.1). Computational Fluency goes beyond the problem-solving strategies of CT by including student's creativity and expression with digital tools, and the opportunity for students to develop their own voice and identity (Resnick, 2018).

Resnick's emphasis on having students design digital artifacts is well grounded in constructionism and Resnick acknowledges the surge of interest in coding and schools "provides an opportunity for reinvigorating and revalidating the Constructionist tradition in education" (Resnick, 2014, p. 7). Resnick and Papert's views on constructionism are thoroughly discussed in Constructionism in practice: Designing, thinking, and learning in a digital world, a book edited by Resnick and another one of Papert's influential students, Yasmin Kafai.

5. COMPUTATIONAL PARTICIPATION

Kafai was a student of Papert's at the MIT Media Laboratory and also contributed to the development of Scratch. Her recent work includes Connected code: Why children need to learn programming, a book that she co-authored with Quinn Burke.

In Connected Code, Kafai and Burke describe four dimensions characteristic of Papert's constructionist thought (social, personal, cultural, and tangible) and explain how these dimensions have evolved resulting in a new form of programming whereby students can create applications as part of a larger community. This programming as a participatory process extends CT because "when code is created, it has both personal value and value for sharing with others" (Kafai & Burke, 2014, p. 17). In From computational thinking to computational participation in K-12 Education (2016), Kafai argues that CT needs to be reframed as Computational Participation moving us "beyond tools and code to community and context" (p. 27).

Kafai's Computational Participation acknowledges that CT is a social practice with a broad reach and that programming is now a way to make and be (Kafai, 2016) in the digital world (Kafai, 2016). Digital technologies are used for functional, political, and personal reasons and therefore all students should develop an understanding of interfaces, technologies, and systems that they encounter every day in order to fully participate in contemporary activities and social practices.

Kafai's Computational Participation takes a broad view of computing and acknowledges its potential impact across a wide range of fields. This broad view shares some characteristics with Computational Literacy, an idea that was developed by Andrea diSessa even before Wing's CT became popular.

6. COMPUTATIONAL LITERACY

Andrea diSessa's work focusses on the idea that computers can be the basis of a new form of literacy that is applicable to a wide variety of subjects, contexts and domains (Weintrop et al., 2016). In 2000, six years before the publication of Wing's Computational Thinking, diSessa published Changing Minds, a book in which he "invites us to imagine a world in which computational knowledge – the prime example is programming – is as widely practiced as reading newspapers and novels is today" (Papert, 2006, p. 240)

In presenting computing as a new form of literacy, diSessa advocated for the broad use of computers in schools, and for educators to see computing as means of transforming the teaching and learning of things that are hard for students to learn (Papert, 2006). diSessa uses algebra as an example of an epistemological entity that, when first developed, was not appreciated as a means of transforming complex and difficult ideas into a form that can be grasped by high school students (Papert, 2006). He argues that Computational Literacy involves computing and computer programming concepts being integrated into school subjects in much the same way that algebra has become a tool in science, mathematics and other subjects.

In Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education (2018), diSessa explains that his use of the term literacy goes beyond the idea of simply having a casual acquaintance with something. Instead, literacy means the adoption, by a broad group or even a civilization, of a "particular infrastructural representational form for supporting intellectual activities" (diSessa, 2018, p. 4). diSessa continues by criticizing Wing's computer science-centric view of CT acknowledging that because literacy is such a massive social and intellectual accomplishment, it can't belong to a single professional discipline.

diSessa concludes Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education by providing practical advice:

There is no single recipe for how computation changes a field or subfield. If your pursuits take you in different directions, then I suggest here, that will enrich the horizon for all of us. If they parallel or extend what I and others who are focused on the big picture have already done, perhaps we can converge sooner than might be expected (diSessa, 2018, p. 28).

We should consider this advice as we investigate the views and applications of CT shared by other researchers within the field.

7. CT DEFINITIONS AND MATHEMATICAL MODELS

In 2017, Peter Denning published Remaining trouble spots in computational thinking, where he explained that CT has been major component of computer science since the 1950s and so has the idea that CT can benefit people in a variety of fields. Unfortunately, Denning claims, recent attempts to make CT appealing to fields other than CS have led to "vague and confusing definitions of CT" (p. 33). Denning's two main criticisms of Wing's definition of CT include the absence of any mention of computational models as well as the suggestion that any sequence of steps constitutes an algorithm. Denning prefers, instead, to accept a definition of CT proposed by Alfred Aho, which he claims better embodies the notion of CT from computer science, computational science, as well as other fields such as the humanities, law and medicine.

In 2012, Aho defined CT quite succinctly as "the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 832). Aho explained that an important part of the CT thought processes involve finding the appropriate models of computation, and if there are none, then developing new ones. This view is exemplified in some of the mathematical modelling work by Michelle Wilkerson.

Wilkerson believes that computer science shares language with mathematics that can be used to represent models resulting in a description of patterns and processes that can make up scientific and engineered systems (Wilkerson & Fenwick, 2017). When describing CT, Wilkerson, and co-author Michele Fenwick, explain:

While mathematics focuses on quantities, computational thinking focuses on processes. Students engaged in the practice of computational thinking break a complex problem or process up into smaller steps in order to better understand, describe, or explain it (Wilkerson & Fenwick, 2017, p. 189).

Wilkerson works with having students use or build computational models and simulations in order to better understand scientific and engineered systems. This approach to CT would be considered by Shuchi Grover as a good example of integration CT in an effort to enable or enrich learning in other disciplines.

8. A TALE OF TWO (OR THREE OR FOUR OR FIVE) CTs

In A tale of two CTs (and a Revised Timeline for Computational Thinking) (2018), Grover argues that in order to make sense of CT in K-12 education we need to distinguish between main two views: computer science thinking in CS classrooms and CT in other disciplines. She explains that ideally, students will get a chance to experience CT in both settings during their K-12 schooling. Grover also presents a brief timeline of CT starting with the problem-solving practices discussed by G. E. Forsythe in 1968 and the elements of CS thinking discussed by Donald Knuth in the 1980s.

In regards to Wing, Grover credits her definition of CT for igniting K-12 computer science education and for calling attention to its role in other disciplines but also acknowledges that we should no longer be focused on "dreams of CT changing everyday behaviours of those who've learned this skill in curricular settings". Instead, we should view CT as playing a significant role in CS education and playing a role in helping students understand concepts within a variety of fields and disciplines.

9. CONCLUSION – MULTIPLE DIRECTIONS

While the varied approaches to CT may indicate disagreement on behalf of researchers in the field, it can also be a sign of the varied directions in which this powerful form of thinking can be taken. diSessa makes it clear that Computational Literacy is distinct from CT and that the field should have an analytical frame that can separate these ideas, and other CT ideas and movements (diSessa, 2018, p. 17). He goes on to explain that "it's not an issue of choosing terms; it is an issue of choosing directions" (diSessa, 2018, p. 17).

When deciding on how to frame an essay on Papert's ideas, Resnick acknowledged that it's "too simplistic to think that you can just take someone's ideas and put them into practice. Seymour was always skeptical about that type of top-down,

linear thinking" (Resnick, 2017b). Perhaps varied approaches related to computer education and CT are an inevitable outcome of the epistemological and practical underpinnings of the concept, as well as the nature of K-12 education.

As students begin to develop an understanding of "thinking like a computer", or "thinking like a computer scientist", they enter the interesting and sophisticated realm of epistemology. To claim that there is one approach to having students work within this realm, and one direction for educators and researchers to take, discredits the nature of the underlining, constructivist theory of knowledge. To claim that there is one way to implement CT concepts in the various disciplines and grades of K-12 education discredits the subjective and responsive nature of teaching and learning.

As we consider CT and K-12 education, we should understand that it's too simplistic to think that we can take Wing's general ideas of CT and put them into practice. The varied approaches and directions listed above represent an honest and authentic characteristic of a body of knowledge whose foundation lies in the constructivist theory of learning. There are several common, core principles and beliefs that lie at the heart of a number of researcher's views on CT. These should continue to be documented and shared, while the subtle differences surrounding the details of CT should continue be investigated and celebrated. The computer and the mind of a student can "take on a thousand forms and can serve a thousand functions", perhaps the varied approaches to integrating CT in K-12 education should honour this idea.

10. REFERENCES

Aho, A.V. (2012). Computation and Computational Thinking. *Computer Journal, 55*, 832–835.

Ames, M.G. (2018). Hackers, Computers, and Cooperation: A Critical History of Logo and Constructionist Learning. *Proceedings of the ACM on Human-Computer Interaction, 2*, 1-19.

Barba, L. (2016). *Computational thinking: I do not think it means what you think it means*. Retrieved December 1, 2019 from http://lorenabarba.com/blog/computational-thinking-i-do-not-think-it-means-what-you-think-it-means/

Brennan, K., Resnick, M. (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, 1*, 25.

Denning, P. J. (2017). Remaining Trouble Spots with Computational Thinking. *Communications of the ACM, 60*(6), 33–39.

DiSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge, MA: MIT Press.

DiSessa, A. A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. *Mathematical Thinking and Learning, 20*(1), 3-31.

Grover, S. (2018). *A Tale of Two CTs (and a Revised Timeline for Computational Thinking)*. Retrieved December 1, 2019 from https://cacm.acm.org/blogs/blog-cacm/232488-a-tale-of-two-cts-and-a-revised-timeline-for-computational-thinking/fulltext

Kafai, Y., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Mahwah, NJ: Lawrence Erlbaum Associates.

Kafai, Y.B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. Cambridge, MA: MIT Press.

Kafai, Y. (2016). From Computational Thinking to Computational Participation in K-12 Education. *Communications of the ACM, 59*(50), 26-27.

Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas (2nd ed.)*. New York, NY: Basic Books.

Papert, S. (2016). Minding change: Essay Review of Changing Minds: Computers, Learning, and Literacy by Andrea diSessa. *Human Development, 49*, 239-247.

Resnick, M. (2014). Give P's a Chance: Projects, Peers, Passion, Play. *Proceedings of the Third International Constructionism Conference.* Vienna: Austrian Computer Society, 13-20.

Resnick, M. (2017). The Seeds That Seymour Sowed. Retrieved December 1, 2019 from https://www.media.mit.edu/posts/the-seeds-that-seymour-sowed/

Resnick, M. (2018). *Computational Fluency*. Retrieved December 1, 2019 from https://medium.com/@mres/computational-fluency-776143c8d725

Stager, G. (2016). Seymour Papert (1928-2016) Father of Educational computing. *Nature, 537*, 308.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (2015). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wilkerson, M. H. & Fenwick, M. (2017). The practice of using mathematics and computational thinking. *Helping Students Make Sense of the World Using Next Generation Science and Engineering Practices*. Arlington, VA: National Science Teachers' Association Press, 181-204.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33–36.

Wing, J. M. (2011). *Research Notebook: Computational Thinking—What and Why?* Retrieved December 1, 2019 from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Effects of Using Mobile Phone Programs to Control Educational Robots on the Programming Self-Efficacy of the Third Grade Students

[1]Yi-ting LIN, [2]Ting-chia HSU[*]
[1,2]National Taiwan Normal University, Taiwan
i79432ytl@gmail.com, ckhsu@ntnu.edu.tw

ABSTRACT
The purpose of this study is to use mobile phone programs to control educational robots, so as to enhance the computational thinking literacy of the third grade students. This study allows learners to use mobile phone applications to drag building blocks in order to control and interact with the educational robots. This study also employed the computer program self-efficacy scale, educational robot attitude scale and cognitive load scale to measure computational thinking ability and learning performance. The research results show that learners have significantly improved their computational thinking ability after course, and further analysis found that learners' self-efficacy performance of computer programs is significantly negatively related to learning anxiety, and learning investment and learning image are significantly positively related.

KEYWORDS
computational thinking, educational robot, computer programming self-efficacy

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

三年級學生使用手機程式控制教育機器人對其程式自我效能表現之研究

[1]林羿婷，[2]許庭嘉*
[1,2]國立臺灣師範大學，科技應用與人力資源發展學系，臺灣
i79432ytl@gmail.com，ckhsu@ntnu.edu.tw

摘要
本研究旨在針對國小三年級學生於課程上使用手機程式控制教育機器人學習後，其運算思維能力提升之研究。本研究在課堂上讓學習者使用手機應用程式拖拉積木程式與教育機器人進行互動，並使用電腦程式自我效能量表、教育機器人態度量表及認知負荷量表進行運算思維能力與學習表現之測量，研究結果發現學習者在學習後其運算思維能力有顯著提升，而進一步分析發現學習者之電腦程式自我效能表現與學習焦慮有顯著負相關，學習投入及學習意象有顯著正相關。

關鍵字
運算思維；教育機器人；電腦程式自我效能

1. 前言
隨著科技發展，電腦資訊能力已成為每個人都必須學會的技能，而在面對未來越來越多的新興問題挑戰，可以使用運算思維來進行問題拆解、問題解決（Wing, 2006），且運算思維能力的應用並不侷限於電腦科技，而是可以跨領域應用在不同項目內（Barr & Stephenson, 2011）。在 2008 年 Wing 學者針對運算思維的研究有提及運算思維是兒童教育不可或缺的一部分（Wing, 2008），隨後學者的研究進行方向也朝向兒童的運算思維教育（García-Peñalvo, 2018; Grover & Pea, 2013; Rich, Binkowski, Strickland, & Franklin, 2018），不過要如何養成運算思維的能力尚待需要更多實證研究來佐證（Grover & Pea, 2013）。臺灣於 2019 年開始施行之十二年國民基本教育課程綱要將核心素養列為課程發展主軸，核心素養是指一個人為適應生活及面對未來挑戰，所應具備的知識、能力與態度（教育部，2014），在科技領域部份將運算思維列為學習重點（教育部，2018），這顯示了運算思維將成為兒童需要學習的重要能力。

隨著科技技術的提升，機器人的發展也是一個未來值得討論的重要議題。越來越多學者提出使用機器人進行教育的相關研究（Angeli & Valanides, 2019; Berry, Remy, & Rogers, 2016; Lye, Wong, & Chiou, 2013），將機器人當成提升思考邏輯、解決問題及團隊合作能力的工具是一重要的研究方向（Benitti, 2012），根據 2018 年 Cheng、Sun 與 Chen 學者對於機器人基本應用的類別統計，教育機器人屬於第二重要的應用領域，而教育機器人應用的領域大多在 STEAM 教育及語言教育，同一份研究也指出未來在學齡前及小學使用教育機器人具有最大的潛力（Cheng, Sun, & Chen, 2018）。亦有針對使用教育機器人進行運算思維能力的培養研究指出，學生的運算思維能力在學習過程中皆有增加（Atmatzidou & Demetriadis, 2016; Chen et al., 2017）。

綜上所述，運算思維已經是當前教育的發展重點，對於使用教育機器人進行運算思維能力的提升需要更多的研究佐證，因此本研究將融合教育機器人及運算思維，將使用手機應用程式控制教育機器人應用在國小三年級學童課程上，著重於運算思維能力之學習成效探討，以期透過本研究的成果，提供給未來期望使用教育機器人提升學生運算思維能力的教師，在未來課程設計上能有所啟發。

2. 文獻探討
2.1. 運算思維
運算思維是基於像電腦科學家一樣思考的想法，來解決問題、設計系統及理解人類行為，利用抽象化、拆解一個複雜困難的問題，讓它化為一個知道如何解決的問題（Wing, 2006），運算思維的能力可以用來解決學習上或生活上的問題（Tsai, Wang, & Hsu, 2018）。運算思維具有概念化的特性，而其本質在於抽象化，其中包含著能夠將元素進行交織與結合（Wing, 2008）。而當學生能夠學習運算思維並將腦海中解決問題的想法透過電腦實現，學生不僅會成為工具的使用者，更會成為工具的建造者（Barr & Stephenson, 2011）。BBC（2017）對運算思維進行了四個面向的分類，分別是問題拆解、模式識別、抽象化及演算法概念，問題拆解是指將大問題拆解成各個小問題再針對小問題進行解決，模式辨別是要找出小問題與小問題之間共同之處，抽象化是找出問題的關鍵並找出相對關係或進行模組化，而演算法概念則是最後產生解決問題的步驟，這些類別並沒有一定的次序，而是不斷循環發生的（許庭嘉，2018），ISTE 更於 2018 年提出新的運算思維標準，認為需要在學生的學習科目中加入運算思維，讓學生具備解決未來問題的能力（ISTE，2018）。

2.2. 教育機器人
教育機器人是使用機器人作為一種教育工具（Lye et al., 2013），無論是教育機器人還是使用機器人進行學習，皆同時著重於鼓勵學生的參與和探索（Berry et al., 2016），在任何領域裡面教育機器人作為學習工具是擁有發展潛力的（Benitti, 2012），Cheng 等 2 名學者在 2018 年發表的研究報告統計了機器人基本應用的類別，教育機器人屬於第二重要的應用領域。目前大部分教育機器人應用的教育領域為 STEAM 教育、語言教育、特殊教育（Pei & Nie, 2018）。學者 Pei 和 Nie（2018）將教育機器人分為四類，分別是智能助手機器人、虛擬模擬機器人、多功能套件機器人和非通用教育機器

人，智能助手機器人是指該機器人集人工智能且具有智能對話系統可以執行語義識別，虛擬模擬機器人是具模擬功能較常用於機器人教學，多功能套件機器人具有各種模組可以自由組合常用於 STEAM 教育，非通用教育機器人是針對特殊族群所設計的機器人，如自閉症兒童的教育機器人，除了四種類的機器人，這兩位學者亦統整出教育機器人的五種特徵，分別為靈活性、數據化、重複性、人性化、自然互動的。其他研究有顯示因機器人擁有重複性及互動的特性（Toh, Causo, Tzuo, Chen, & Yeo, 2016）。

2.3. 電腦程式自我效能

自我效能的概念是由 Bandura 提出，是指個體對於自己的期望會決定個體評估要付出多少努力與要經歷多久的困難（Bandura, 1977）。而電腦程式自我效能是學習者對自己電腦程式能力的看法（Korkmaz & Altun, 2014），瞭解學生的電腦程式自我效能和其他影響自我效能的因素（Hasan, 2003），將可以在未來課程設計上有更多的建議（Psycharis & Kallia, 2017）。電腦程式自我效能也會和課程表現有關，有研究發現學習程式的自我效能會受到過去學習程式的經驗影響（Benitti, 2012; Law, Lee, & Yu, 2010）。Tsai 等 2 位學者則發展了測量電腦程式自我效能的量表，量表共含有五個構面，分別是邏輯思維、合作、計算、控制和除錯，而此份電腦程式自我效能量表也可以針對學生的運算思維進行評估（Tsai et al., 2018）。

2.4. 學習焦慮

焦慮是一種情緒狀態，是一種對於情況感到緊張、害怕與憂慮的感覺。研究互動機器人的學者認為需要更關注在個體與機器人進行互動時所產生的焦慮（Nomura, Suzuki, Kanda, & Kato, 2006），亦有學者則認為焦慮是衡量學生學習經驗的重要變量，並將學生與機器人互動時產生的焦慮定義為學生在課程上產生的恐懼、焦慮和迴避和使用機器人有關，且針對此項目進行問卷測量題目的設計，用以了解學生對於機器人出現在課程上會不會產生緊張、不喜歡或受干擾的感覺（Sisman, Gunay, & Kucuk, 2018）。

2.5. 學習意向

針對教育機器人在課程上對學生的影響，學習意向指的是學生在未來課程中預期與機器人進行的互動（Sisman et al., 2018）。在一項對於機器人作為輔助學習的研究中指出，擬人化的機器人可以提高學生的學習興趣，大多數的學生都喜歡使用類人類機器人進行學習（Chin, Wu, & Hong, 2011）。而在一項使用機器人進行英文教學的研究指出，學生在學習過程中很開心，相信自己正在學習得更好、更有效，從長遠角度來看，這會增強他們的學習動機（Alemi, Meghdari, & Ghazisaedy, 2014）。

3. 研究方法

3.1. 研究流程

本研究實驗對象為 21 位臺北市某國小三年級學生，在學生課前進行電腦程式自我效能量表前測，並向學生介紹積木程式及講解機器人操作流程，接著讓學生親自操作使用手機應用程式，學生需要拉動並堆疊模組化的積木程式，當學生堆疊出相對應的積木程式後，置於桌上的機器人則會產生對於堆疊之積木正確與否的反應，學生可以觀察機器人明白自己所拖拉的積木是否正確，最後讓學生進行分組競賽活動，藉由分組競賽讓學生更加熟練積木及機器人操作，課程完成後再對學生施以電腦程式自我效能量表、教育機器人態度量表及認知負荷量表進行後測，用以探討學生使用教育機器人進行課程學習之學習成效。完整研究流程如圖 1。

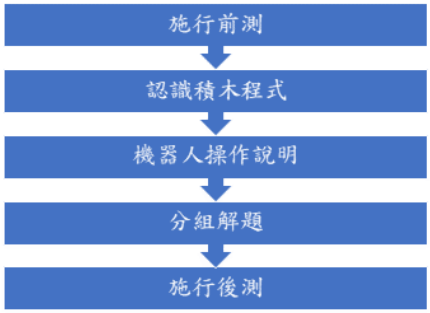


圖 1 研究流程圖

圖 2 則為學生使用手機程式，拉動積木程式方塊進行機器人操控之介面畫面。

圖 2 手機應用程式介面畫面

3.2. 測量工具

本研究使用電腦程式自我效能量表、教育機器人態度量表及認知負荷量進行學習成效測量。

電腦程式自我效能量表為 2018 年由 Tsai Meng-Jung、Wang Ching-Yeh 和 Hsu Po-Fen 發展的量表，在本研究使用的問卷量表使用三個構面，分別為「邏輯思考」、「控制」、「除錯」，邏輯思考構面的 Cronbach's alpha 值為 0.80，控制構面的 Cronbach's alpha 值為 0.82，除錯構面的 Cronbach's alpha 值為 0.83，量表各構面皆具有可接受之信度。

教育機器人態度量表為 2018 年由 Burak Sisman、Devrim Gunay 和 Sevda Kucuk 所發展的量表，在本研究的問卷量表使用三個構面，分別為「學習投入」、「學習焦慮」、「學習意向」，學習投入構面 Cronbach's alpha 值為 0.73，學習焦慮構面 Cronbach's

表 1 電腦程式自我效能表現與其他構面之相關矩陣表

	1	2	3	4
7. 電腦程式自我效能	-			
7. 學習投入	.28	-		
7. 學習焦慮	-.62**	-.17	-	
7. 學習意向	.19	.73**	-.08	-
7. 認知負荷	-.02	-.13	.12	-.25

**p<.01

alpha 值為 0.81，意向構面 Cronbach's alpha 值為 0.53，量表各構面皆具有可接受之信度。

認知負荷量表為 1998 年 John Sweller、Jeroen J. G. van Merrienboer 和 Fred G. W. C. Paas 發展的量表，在本研究的問卷使用量表 Cronbach's alpha 值為 0.83，具有可接受之信度。

3.3. 資料分析方法

本研究使用統計軟體 IBM SPSS Statistics 23，對電腦程式自我效能量表三構面進行前、後測相依樣本 t 檢定，用以瞭解學生使用手機程式控制教育機器人後的運算思維學習成效，並使用 Pearson 相關分析進一步探討學生電腦程式自我效能表現與教育機器人使用態度及認知負荷之相關性。

4. 實驗結果分析

本研究施測對象為台北市某國小三年級學生，回收之有效問卷數為 21 份，性別分布為男性 14 位（66.7%），女性 7 位（33.33%）；針對是否有聽過積木程式變項，有 3 人（14.3%）回答為是，14 人（66.7%）回答為否。

欲探究學生在電腦程式自我效能表現，將問卷三構面進行前後測相依樣本 t 檢定分析。根據分析結果發現（如表 1），三個構面的前測與後測平均值皆有顯著差異，在「邏輯思考」構面 t（20）= -3.16，p<.05，後測得分（M=3.79，SD=1.04）顯著大於前測得分（M=3.08，SD=0.95）；「控制」構面 t（20）= -4.90，p<.05，後測得分（M=2.81，SD=1.35）顯著大於前測得分（M=1.44，SD=0.93）；在「除錯」構面 t（20）= -3.62，p<.05，後測得分（M=3.29，SD=1.39）顯著大於前測得分（M=1.98，SD=1.29）。

表 2 電腦程式自我效能表現量表之相依樣本 t 檢定表

項目	平均數（標準差）		自由度	*t*
	前測	後測		
邏輯思考	3.08（0.95）	3.79（1.04）	20	-3.16**
控制	1.44（0.93）	2.81（1.35）	20	-4.90***
除錯	1.98（1.29）	3.29（1.39）	20	-3.62**

p<.01,*p<.001

欲深入探討學生電腦程式自我效能表現與教育機器人使用態度、認知負荷之相關性，進行 Pearson 相關分析，分析結果如表 2。結果發現：電腦程式自我效能表現與學習焦慮〔r（20）=.62，p<.001〕有顯著負相關；學習投入及學習意向〔r（20）=.73，p<.001〕則有顯著正相關。

5. 結論與未來展望

實驗結果顯示，學生在使用手機應用程式與教育機器人進行互動學習後，其運算思維的能力皆有顯著提升，表示學生使用教育機器人進行運算思維的學習是有效果的。研究結果符合 Atmatzidou 和 Demetriadis（2016）及 Chen 等五位學者（2017）對於運算思維及機器人主題的研究結果，意即使用教育機器人進行運算思維的能力提升是有成效的。

為了更加瞭解學生運算思維能力與使用教育機器人之間的關係，進一步針對量表各構面進行 Pearson 相關性分析後，結果發現學生的電腦程式自我效能表現與學習焦慮有顯著負相關，意即學生有較高的電腦程式自我效能則會有較低的學習焦慮，如學生有較低的電腦程式自我效能則會有較高的學習焦慮，此研究結果符合 Bandura 提出的自我效能理論（Bandura, 1977）。

本研究實驗結果亦呈現學習投入及學習意向之間呈現正相關性，吻合 Sisman 於 2018 年發展機器人態度量表時所測得之投入及意向構面間的相關性，也更加顯示以此量表穩定，可以用來測量學生的學習投入與學習意向。

綜合上述對於實驗結果與討論，本研究呈現一個使用手機應用程式及教育機器人進行教學而提升學生運算思維能力的例子，實驗成效供給未來期望使用教育機器人提升運算思維能力的教師，在課程設計上能有所啟發。由於本研究僅針對臺灣臺北市某國小三年級學生進行實驗研究，未來研究方向則可以針對不同地區的學生進行實驗研究，也可以針對不同年齡與性別的學生進行實驗研究，讓研究更加全面也增加研究的擴論程度。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 108-2511-H-003 -056 -MY3 的部分補助。

7. 參考文獻

教育部（2014）。**十二年國民基本教育課程綱要總綱**。臺北市：教育部。

教育部（2018）。**十二年國民基本教育課程綱要國民中學暨普通型高級中等學校-科技領域**。臺北市：教育部。

許庭嘉（2018）。**寓教於樂：如何從桌上遊戲學習結構化程式設計邏輯：含 Robot City v2 桌遊包**。新北市：台科大圖書。

Alemi, M., Meghdari, A., & Ghazisaedy, M. (2014). The Effect of Employing Humanoid Robots for Teaching English on Students' Anxiety and Attitude. *Preceedings of the 2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 754-759.

Angeli, C., & Valanides, N. (2019). Developing Young Children's Computational Thinking with Educational

Robotics: An Interaction Effect between Gender and Scaffolding Strategy. *Computers in Human Behavior*, 105954. doi:https://doi.org/10.1016/j.chb.2019.03.018

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. doi: https://doi.org/10.1016/j.robot.2015.10.008

Bandura, A. (1977). Self-efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological review*, *84*(2), 191. Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *Inroads*, *2*(1), 48-54.

Benitti, F. B. V. (2012). Exploring the Educational Potential of Robotics in Schools: A Systematic Review. *Computers & Education*, *58*(3), 978-988. doi: https://doi.org/10.1016/j.compedu.2011.10.006

Berry, C. A., Remy, S. L., & Rogers, T. E. (2016). Robotics for All Ages: A Standard Robotics Curriculum for K-16. *IEEE Robotics & Automation Magazine*, *23*(2), 40-46.

British Broadcasting Corporation (2017). *Introduction to Computational Thinking.* Retrieved Sep 19, 2017, from https://www.bbc.co.uk/bitesize/guides/zp92mp3/revision/1doi:10.1109/MRA.2016.2534240

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. doi: https://doi.org/10.1016/j.compedu.2017.03.001

Cheng, Y. W., Sun, P. C., & Chen, N. S. (2018). The Essential Applications of Educational Robot: Requirement Analysis from the Perspectives of Experts, Researchers and Instructors. *Computers & Education*, *126*, 399-416. doi: https://doi.org/10.1016/j.compedu.2018.07.020

Chin, K., Wu, C., & Hong, Z. (2011). A Humanoid Robot as a Teaching Assistant for Primary Education. *Proceedings of the 2011 Fifth International Conference on Genetic and Evolutionary Computing*, 21-24.

García-Peñalvo, F. J. (2018). Editorial Computational Thinking. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, *13*(1), 17-19. doi:10.1109/RITA.2018.2809939

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43. doi:10.3102/0013189x12463051

Hasan, B. (2003). The Influence of Specific Computer Experiences on Computer Self-efficacy Beliefs. *Computers in Human Behavior*, *19*(4), 443-450. doi: https://doi.org/10.1016/S0747-5632(02)00079-1

International Society for Technology in Education (2018). *ISTE Announces New Computational Thinking Standards for All Educators.* Retrieved Oct 8, 2018 from https://www.iste.org/explore/Press-Releases/ISTE-Announces-New-Computational-Thinking-Standards-for-All-Educators

Korkmaz, Ö., & Altun, H. (2014). Adapting Computer Programming Self-Efficacy Scale and Engineering Students' Self-Efficacy Perceptions. *Online Submission*, *1*(1), 20-31.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning Motivation in E-Learning Facilitated Computer Programming Courses. *Computers & Education*, *55*(1), 218-228. doi: https://doi.org/10.1016/j.compedu.2010.01.007

Lye, N. C., Wong, K. W., & Chiou, A. (2013). Framework for Educational Robotics: A Multiphase Approach to Enhance User Learning in a Competitive Arena. *Interactive Learning Environments*, *21*(2), 142-155.

Nomura, T., Suzuki, T., Kanda, T., & Kato, K. (2006). Measurement of Anxiety toward Robots. *Proceedings of the 15th IEEE International Symposium on Robot and Human Interactive Communication (Ro-Man06)*, 372 - 377.

Pei, Z., & Nie, Y. (2018). Educational Robots: Classification, Characteristics, Application Areas and Problems. *Proceedings of the 2018 Seventh International Conference of Educational Innovation through Technology (EITT)*, 57-62.

Psycharis, S., & Kallia, M. (2017). The Effects of Computer Programming on High School Students' Reasoning Skills and Mathematical Self-efficacy and Problem Solving. *Instructional Science*, *45*(5), 583-602. doi:10.1007/s11251-017-9421-5

Rich, K. M., Binkowski, T. A., Strickland, C., & Franklin, D. (2018). Decomposition: A K-8 Computational Thinking Learning Trajectory. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 124-132.

Sisman, B., Gunay, D., & Kucuk, S. (2018). Development and Validation of an Educational Robot Attitude Scale (ERAS) for Secondary School Students. *Interactive Learning Environments*, *27*(3), 377-388. doi:10.1080/10494820.2018.1474234

Toh, L. P. E., Causo, A., Tzuo, P.-W., Chen, I.-M., & Yeo, S. H. (2016). A Review on the Use of Robots in Education and Young Children. *Journal of Educational Technology & Society*, *19*(2), 148-163.

Tsai, M.-J., Wang, C.-Y., & Hsu, P.-F. (2018). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research*, *56*(8), 1345-1360. doi:10.1177/0735633117746747

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Exploring Creativity, Emotion and Collaborative Behavior in Programming for Two Contrasting Groups

Dan SUN[1], Fan OUYANG[2*], Yan LI[3], Hongyu CHEN[4]
[1,2,3,4] College of Education, Zhejiang University, China
dansun@zju.edu.cn, fanouyang@zju.edu.cn, yanli@zju.edu.cn, 21903051@zju.edu.cn

ABSTRACT
Studies have reported that acquiring programming skills remains a considerable challenge for most novice learners. In this study, Minecraft was used to explore its' effect on secondary students' creativity, emotion, as well as collaborative behaviors during programming in groups. Four out of twenty secondary students were recruited and instructed in the design and development of programming project in Minecraft, and pair programming was used to foster their collaboration. The results revealed that students' creativity was increased. Their emotion of enjoyment, hope and pride towards programming were improved and feeling of anxiety were decreased after the intervention. Two groups behavior patterns during programming were detected through click stream analysis, which revealing five typical behaviors (UTP, CIP, DIM, CIM and DBM). This study shed light on the dynamic connection between programming learning and cultivation of creative ability and positive emotion, and how to better integrate learning analytics in K-12 programming education.

KEYWORDS
programming, secondary students, creativity, emotion, behavior

1. INTRODUCTION
Based on a constructivist approach, programming has been suggested as the main strategy to improve computational thinking (CT) in schools (Wing & M., 2006). However, past experience shows that many students perceive learning computer programming as a difficult and boring task, teaching programming languages remains a big challenge for most school teachers (Barr & Guzdial, 2015). Minecraft is acknowledged as a highly popular children's digital game and show its potential in education (Pellicone & Ahn, 2018). Cipollone, Schifter, & Moffat (2014) demonstrated that Minecraft offers a unique opportunity for students to express their creativity. Minecraft also could provide the text-based language (Python) learning environment, together with a vivid debugging interface. However, less attention has been paid to apply Minecraft for programming projects. Therefore, the research questions was: What were the differences in creativity, emotion and collaborative behaviors of two contrasting groups in Minecraft programming?

2. METHOD

2.1. Research Context
In this study, Minecraft was utilized as programming learning environment. Callaghan (2016) suggested that Minecraft could contribute to the enhancement of classroom learning, the capacity to collaborate as well as the role of the teacher contributed to a learning environment. Besides, Python is a language that was designed specifically for teaching programming to non-experts. And Python has been adopted in Chinese High school Information Technology Curriculum Standards. The demonstration of teaching Python programming in Minecraft was shown in Figure 1.

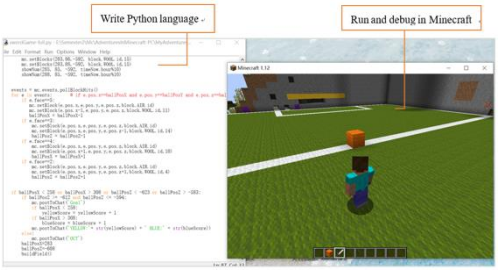


Figure 1. Write, run and debug Python in Minecraft.

2.2. Measuring Instrument
Firstly, the creativity questionnaire was revised based on the measure developed by Welch and McDowall (2010). The Cronbach's alpha value of the questionnaire was .80. Secondly, the questionnaire of emotion in programming, developed by Goetz, Frenzel, Barchfeld and Perry (2011), which is designed to assess various achievement emotions experienced by students in academic settings. The Cronbach's alpha value of the questionnaire was .93. Thirdly, click stream analysis was adopted to explore students' collaborative behavior during the programming process.

2.3. Experiment participants and procedure
Based on students' performance and collaboration during the experiment, 2 groups of 4 students of this study were selected from 20 seventh graders in a secondary located in Hangzhou, Ying and Dai in group 1 and Chen and He in group 2.

The experiment took 12 weeks of one and half hours per week. Teacher taught how to program and play with Minecraft, and students were learning basic grammar and function in Python, with implementing different project-based programming cases. During the final programming project, students were asked to conduct a pair programming activities (swordgame), including two students working on one computer, with one acted as Driver and the other as Navigator.

3. RESULTS AND DISCUSSION
Firstly, at the first beginning of the experiment, Ying and Dai in group 1 got 80 and 61, while Chen and He in group 2 got 72 and 50, with an average score of 59 for the whole class. After 12 weeks of study, two groups of students all

improved individually, on top of that, students in group 2 got a much significant enhancement during the study.

Secondly, it is apparent that all of four students' emotion were strengthened after the experiment, no matter if it is in overall performance, or in every aspects of enjoyment, hope and pride. Deci et al., (2017) also mentioned that learner's emotion could influence students' judgement, motivation and self-efficacy towards a specific task. However, He in group 2 experienced a decrease in anxiety score, which means his anxious feeling is rising along with the conducting of the class. Therefore, for those students who couldn't catch up at the beginning or those one couldn't work well with partner, the sense of anxiety would also increase with the time goes by.

Thirdly, in terms of students' behaviors during the programming in Minecraft, we incorporated the click stream analysis. As it can be seen from Figure 2, the top 2 common behaviors are CIP (coding in python) and DIM (debugging in Minecraft), the rest behaviors are DBM (distracted by Minecraft), CIM (creating in Minecraft) and UTP (Understanding the project). Students in groups 1 have spent the most of the time in CIP and more likely to write code to complete the programming task; whereas the group 2 seems to enjoy create building by mouse click (CIM) rather than code-writing. Students in group 1 were appeared to be more concentrated on programming, because of the shape of the behavior CIP, DIM for group 1 are much more dense than group 2, whereas the group 2 students' behavior are much scattered, and they have spent more time in analyzing the question and were quite easy to get distracted by the game in Minecraft.

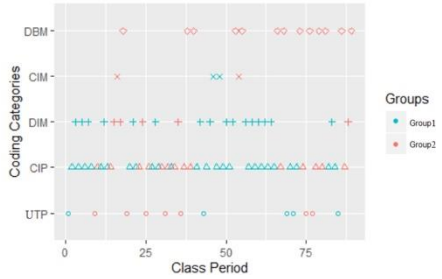


Figure 2. Time-series analysis of programming procedure

4. CONCLUSION AND IMPLICATION

In this study, Minecraft, a creative sandbox game platform, was used as learning environment to teach programming. The experimental data showed that the students' creativity and emotion toward programming were significantly improved after the intervention, revealing the benefits of the proposed approach. In addition to that, students' behaviors (UTP, CIP, DIM, CIM and DBM) were detected through click stream analysis.

Beside, this study contributes to providing new empirical evidence for the valuableness of enhancing creativity in programming education. Besides, researchers mentioned that positive emotion could influence students' intrinsic motivation, such students tend to be more creative and competitive (Deci et al., 2017), and this study also shed light on the dynamic connection between emotion and programming learning, and find a positive impact on how students' perceive programming knowledge.

On top of that, different behavior patterns were found between two contrasting groups, with one group worked peacefully and the other were having conflict. Except for partners' skill and personalities which are most emphasized in previous research (Hung & Young, 2017), this study also revealed that partners' emotion towards the task will influence each other, which will results in mutual success or cruel failure.

5. REFERENCES

Barr, V., & Guzdial, M. (2015). Advice on teaching CS, and the learnability of programming languages. *Communications of the ACM*, *58*(3), 8–9. https://doi.org/10.1145/2716345

Cipollone, M., Schifter, C. C., & Moffat, R. A. (2014). Minecraft as a Creative Tool. *International Journal of Game-Based Learning*, *4*(2), 1–14. https://doi.org/10.4018/ijgbl.2014040101

Deci, E. L., Olafsen, A. H., & Ryan, R. M. (2017). Self-Determination Theory in Work Organizations: The State of a Science. *Annu. Rev. Organ. Psychol. Organ. Behav*, *4*, 19–43. https://doi.org/10.1146/annurev-orgpsych

Goetz, T., Frenzel, A. C., Barchfeld, P., & Perry, R. P. (2011). Measuring Emotions in Students' Learning and Performance: The Achievement Emotions Questionnaire (AEQ). *Contemporary Educational Psychology*, *36*(1), 36–48. https://doi.org/10.1016/J.CEDPSYCH.2010.10.002

Hung, H.-C., & Young, S. S.-C. (2017). Applying Multi-touch Technology to Facilitate the Learning of Art Appreciation: From the View of Motivation and Annotation. *Interactive Learning Environments*, *25*(6), 733–748. https://doi.org/10.1080/10494820.2016.1172490

Pellicone, A., & Ahn, J. (2018). Building Worlds: A Connective Ethnography of Play in Minecraft. *Games and Culture*, *13*(5), 440–458. https://doi.org/10.1177/1555412015622345

Wing, J. M., & M., J. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33. https://doi.org/10.1145/1118178.1118215

Welch, D., & McDowall, J. (2010). A Comparison of Creative Strategies in Teaching Undergraduate Students in the Visual Arts and Design. *ACUADS 2010 Annual Conference*. Retrieved Nov 11, 2019 from https://research-repository.griffith.edu.au/handle/10072/3886

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Canada's CanCode Initiative and the Gender Gap in Computer Science Education

Lisa Anne FLOYD
Western University, London, ON, Canada
lapennar@uwo.ca

ABSTRACT

Initiatives are being implemented around the world to support youth with developing digital literacy skills and computational thinking. Many of these initiatives aim to close gender gaps in the area of science, technology, engineering and math (STEM). In Canada, CanCode is a federal initiative that provides funds for non-profit organizations to support K-12 teachers and their students with developing computational thinking and digital skills. Through the CanCode funding, organizations aim to increase representation of girls in high school computer science classes and post-secondary programs. There are many common approaches that are implemented by organizations including setting-up coding clubs, supporting teachers in K-8, adjusting high school STEM and computer science courses and organizing coding and robotics competitions. Literature suggests best practices and recommendations for such approaches in order to close the gender gap in computer science education. Initiatives such as CanCode are a starting point to ensure all young people, including girls, have the skills to be active contributors to the digital age.

KEYWORDS

computational thinking, computer science, K-12 education, gender gap, coding

1. INTRODUCTION

Initiatives around the world are being implemented to support students with developing computational thinking as it is "increasingly important that people have an understanding of the algorithmic, computational nature of problem-solving involving digital technology" (UNESCO, 2018, p. 26). One such initiative in Canada is CanCode, through which many non-profit organizations have received funding to "support opportunities for Canadian students (kindergarten to grade 12) to learn digital skills including coding, data analytics, and digital content development" (Government of Canada, 2019). CanCode was first launched in September, 2017, providing $50 million in funding and reached over 1.3 million students as well as 61,000 teachers across Canada (Government of Canada, 2019). It has since been extended with an additional $60 million over the next two years aiming to advance "an agenda to build Canada as a world-leading innovation economy that will create good jobs and grow the middle class" (Government of Canada, 2019).

As part of the assessment criteria to receive funding, organizations must have "demonstrated an ability to reach traditionally underrepresented groups including girls, Indigenous youth, and/or youth with disabilities" (Government of Canada, 2019). Common approaches used by the nonprofit organizations funded by CanCode to narrow the gender gap in the areas of computer science and STEM, as well as relevant research and recommendations are described in this paper.

2. THE GENDER GAP AND FUNDED INITATIVES

The Canadian government "recognizes the critical role that gender equality has in building a strong economy that works for everyone" (Government of Canada, 2018, p. 218). Diversity is important for a nation as it is known to help drive innovation, and results in more effective problem solving (Foster, 2019; Kafai & Burke, 2014; Margolis & Fisher, 2002). The Canadian government has made "targeted investments, partnerships, and innovation and advocacy efforts that have the greatest potential to close gender gaps and improve everyone's chance for success" (Government of Canada, 2018, p. 243). Although many interventions and actions are being taken by organizations with the CanCode funding to close the gender gap, "no single action can be recognized as a perfect solution" (Council of Canadian Academics, 2015, p. 122).

2.1. Coding Clubs

Many of the organizations that have received funding will be supporting teachers with starting and maintaining after school coding clubs (Government of Canada, 2019). Clubs tend to be flexible, allowing for youth to focus on their interests and also encouraging collaboration beyond the classroom walls, creating agency (Kafai & Burke, 2014). Kafai and Burke (2014) acknowledge however, that true "computational participation cannot be achieved if only a select few join the clubhouse" (p. 133). While there are equity issues associated with clubs that are held beyond the school day, those who lead such clubs can attempt to develop "more inclusive out-of-school science learning practices" (Dawson, 2017, p. 544). Ideally, teachers will begin to incorporate coding into their classroom, so that "what happens inside and outside classrooms becomes more fluid" (Kafai & Burke, 2014, p. 133).

2.2. Developing Digital Skills in High School

The CanCode initiative is meant to also support improvement of high school courses related to digital literacy and to increase the number of girls enrolled in computer science and STEM programs. Kafai and Burke (2014), Foster (2019), Margolis, Fisher and Miller (1999), and Master, Cheryan and Meltzoff (2016) recommend that new directions for designing activities as well as the tools used in K-12 educational computing efforts, are required in order to broaden not only participation, but also perceptions. Most often, curriculum is misleadingly and unnecessarily highly technical – when it should really be shown to be relevant to many aspects of the world (Margolis et al., 1999). Intentionally changing high school classrooms can create a greater sense of belonging for girls and possibly reduce the gender disparities observed in STEM courses (Master et al.,

2016). Incorporating mentor and peer support programs in high schools has also been shown to encourage girls to stay in computer science courses (Council of Canadian Academics, 2015). This is worth investing in, as there have been "positive signs that learning computer science in high school is correlated with a greater likelihood" of girls to pursue "computer science in postsecondary study" (K–12 Computer Science Framework, 2016, p. 25).

2.3. Early Exposure – Supporting Teachers in K-8

Grades K through 8 provide an opportunity to expose everyone to computer science, which is seen as "critical to reducing current gender disparities" (Master et al., 2016, p. 424), as it might prompt girls to consider computer science courses at the high school and post-secondary levels. Interventions starting as early as the primary grades engage girls early, teaching about the many applications of computer science, and providing hands-on activities which might help to reduce the gender gap (Council of Canadian Academics, 2015). While robotics kits are popular and commonly used by funded organizations, Kafai and Burke (2014) recommend that a variety of digital designs, animations and stories that incorporate different materials and contexts should also be shared with students.

In planning their activities for the youngest learners, organizations should consider not just how to spark the interest of girls in computer science, but also why they are not interested in the first place (Gaymes San Vicente, 2014). Some advocates argue that by designing computer science activities that might better fit into girls' interests, existing stereotypes are being reinforced, but as Kafai and Burke (2014) counterargue, "these tensions are productive because they open up conversations and question fairly narrow perceptions about computation" (p. 101). Master et al. (2016) share in their study that girls' lower sense of belonging "could be traced to lower feelings of fit with computer science stereotypes" (p. 424). Incorporating computer science and STEM into K through 8 classes through creative and less technical means, could help to shift the gender disparity that is currently seen in high school computer science classes and beyond.

2.4. Coding and Robotic Competitions and Hackathons

In many cases, hackathons and coding and robotic competitions have been used by the funded organizations to draw youth interest in computer science. Traditionally, such competitions have been established as part of "creative computing and engineering cultures in K-12 schools" (Kafai & Burke, 2014, p. 95), but they have not reached everyone. In fact, such competitions are expensive, tend to draw mostly boys, and do not seem to increase participation much amongst girls and minorities (Kafai & Burke, 2014). There are many other ways to broaden participation, including collaborative experiences, sharing circles and culturally responsive making opportunities (Kafai & Burke, 2014, p. 102).

3. CONCLUSION

The approaches and related research outlined in this paper indicate that initiatives such as CanCode can provide hope for narrowing the gender gap observed in the area of computer science and STEM. The non-profit organizations involved with CanCode seem to be incorporating many research-based practices outlined in literature and additional recommendations have been highlighted. The CanCode program offers a starting point to ensure that all young people, including girls, have the opportunity to contribute to the digital age by becoming authors and creators, rather than solely consumers of technology.

4. REFERENCES

Council of Canadian Academics. (2015). *Some Assembly Required: STEM Skills and Canada's Economic Productivity*. Retrieved November 30, 2019, from http://www.scienceadvice.ca/uploads/ENG/AssessmentsPublicationsNewsReleases/STEM/STEMFullReportEn.pdf

Dawson, E. (2017). Social Justice and Out-of-school Science Learning: Exploring Equity in Science Television, Science Clubs and Maker Spaces. *Science Education*, *101*(4), 539–547.

Foster, S. (2019). *Women in Stem: Critical to Innovation.* Retrieved November 27, 2019, from https://www.globalpolicyjournal.com/blog/10/01/2019/women-stem-critical-innovation

Gaymes San Vicente, A. (2014). Another Dimension to Streaming. *Our Schools, Our Selves*, *23*(2), 227–259.

Government of Canada. (2018). *Equality + Growth - A Strong Middle Class*. Retrieved November 30, 2019, from https://www.budget.gc.ca/2018/docs/plan/budget-2018-en.pdf

Government of Canada. (2019). *CanCode.* Retrieved November 26, 2019, from https://www.ic.gc.ca/eic/site/121.nsf/eng/home

K–12 Computer Science (2016). K–12 Computer Science Framework. Retrieved November 30,2019, from http://www.k12cs.org

Kafai, Y. B., & Burke, Q. (2014). *Connected Code - Why Children Need to Learn Programming*. Massachusetts: Massachusetts Institute of Technology.

Margolis, J, & Fisher, A. (2002). *Unlocking the Clubhouse: Women in Computing*. Cambridge, Massachusetts: MIT Press.

Margolis, Jane, Fisher, A., & Miller, F. (1999). Caring about Connections : Gender and Computing. *IEEE Technology and Society Magazine*, *18*, 13–20.

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether She Belongs: Stereotypes Undermine Girls' Interest and Sense of Belonging in Computer Science. *Journal of Educational Psychology*, *108*(3), 424–437.

UNESCO Institute for Statistics. (2018). *A Global Framework of Reference on Digital Literacy*. Retrieved November 30, 2019, from http://uis.unesco.org/sites/default/files/documents/ip51-global-framework-reference-digital-literacy-skills-2018-en.pdf

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Unplugged Activities in K-12

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Public-Private-Key Encryption in Virtual Reality: Predictors of Students' Learning Outcomes for Teaching the Idea of Asymmetric Encryption

Andreas DENGEL
Universität Würzburg, Germany
Andreas.dengel@uni-wuerzburg.de

ABSTRACT
With networks being an omnipresent part of children's lives, questions about safe communication in these networks emerge. While the concept of symmetric encryption can be taught in simple and gamified ways, asymmetric encryption as the key idea of secure communication in distributed networks is hard to understand for children and existing approaches to simplify the idea still have their flaws. This paper presents a virtual reality designed around a medieval love story where letters are encrypted, decrypted, and signed by using magic potions that are either public or private. A study with 78 students revealed that the key factors for learning in this virtual environment were presence, emotions, and previous knowledge while neither the effect of the used technology nor the effect of the students' motivation on their learning outcomes were significant.

KEYWORDS
virtual reality, computer science unplugged, cryptography, immersive learning

1. INTRODUCTION
Secure transmission of information is a relevant topic for modern communication: Since the rise of the internet in the 1970s, distributed networks consisting of numerous parties communicating with each other had to tackle the challenge of encryption and decryption to ensure the privacy of the participants in the network. A key idea that emerged with the rise of distributed communication networks characterized by participants that never met before is the asymmetric encryption/decryption. Public and private key algorithms (e.g. Diffie-Hellman key exchange, see Diffie and Hellman 1976 or RSA encryption/decryption, see Rivest, Shamir, and Adleman 1978) pose the main idea how distant parties can communicate securely without any prior contact even if a third party, the man in the middle, intercepts the (encrypted) messages in the network.

With questions on privacy and communication in their digital environment, teaching some of these concepts can help children to achieve a better understanding of their digital everyday surroundings. The basic idea of encryption and decryption can be explained easily, e.g. showing the Caesar encryption/decryption method as an idea of symmetric encryption. In further discussions, the children can talk about the problem that participants need a safe way of exchanging keys before starting the encrypted communication and explore possibilities to do this. In this paper, we explore the possibility of visualizing the idea of asymmetric encryption in a metaphorical way using a virtual reality game about a medieval love story and analyze what factors contribute to the students' learning outcomes.

2. METAPHORS FOR PUBLIC-PRIVATE KEY ENCRYPTION/DECRYPTION
While the mathematical concept of one-way functions that underlies the idea of asymmetric encryption can be quite abstract to explain for children, various metaphorical approaches have been developed to teach this concept. Explanatory ideas include the use of locks and keys (UC Computer Science Education 2008), the mixture of colors (Art of the Problem 2012), as well as boxes which can be locked and unlocked in two different ways (Fekete and Morr 2018) to explain the underlying concept to students.

The original Computer Science Unplugged activity (UC Computer Science Education 2008) uses a box to send a chocolate bar through a network with a man in the middle. A student is given a box that contains a bar of chocolate (as a metaphor for the message that somebody else wants to read). The box has to be sent to another participant in the network (a simple queue of students, one of them being a man in the middle). The students explore ideas of how the box can be locked so that the target person can open the lock while the man-in-the-middle cannot. The students have to deal with the challenge that the key has to remain private and cannot be sent through the network. In this scenario, a solution can be to send the locked box to the target person, the target person adds her own lock to box (so that the box is now locked twice) and sends the box back to the sender. The sender unlocks his/her own lock and sends the box back again so that the target person, once receiving the box, can unlock his/her own lock and get the chocolate. While the underlying idea of the Computer Science Unplugged activity engages students to think about the problem in a metaphorical and fun way without having to understand the underlying mathematical functions behind the key and the lock, the metaphor fails to explain both signing and encrypting. The metaphor also struggles with the physical characteristics of a key (that it can not lock something by itself as it would be needed for signing a message) and those of a lock (that, usually, one would not distribute locks).

Another idea tries to mix colors (Art of the Problem 2012) in order to simulate a secure key exchange: First, each participant has a secret color. Two students, A and B, who want to start an encrypted conversation agree publicly on a color and add their own private colors to it. They exchange the new colors (one with A's private color and one with B's private color) are exchanged publicly. After receiving the mixed color, again, A and B each add their own private color to it. The received color represents the secret key for their communication. Doing so, the parties exchanged a secret color without ever meeting each other in person. The mixture of colors is a good idea for introducing a key exchange (like the Diffie-Hellman key exchange) as it explains the idea of a one-way-function in a simple and

engaging way but its applications for really encrypting and decrypting messages are limited. Further, the realization often fails in reality as the amount of color has to be measured exactly for the activity to work.

IDEA provides IKEA-like manuals for concepts related to Computational Thinking. In their manual for public-private key encryption/decryption (Fekete and Morr 2018), they present a box that can be locked in two directions as a metaphor for the key pair used in a public-private-communication. If the box is locked in one direction with the public key, it can only be unlocked by using the private key. If it is locked using the private key, it can only be unlocked in the other direction with the public key, as both keys only turn the lock in one direction. The metaphor is very close to the actual principle of a secure communication involving public-private-key encryption/decryption and is also capable of explaining the idea of signing a message. But its practical application is limited as it is difficult to actually build a box like this for activities where the students can explore the possibilities of encrypting and decrypting messages.

While there are some approaches of visualizing networks and communication in networks in non-immersive virtual environments (Voss et al. 2013; Sturgeon, Allison, and Miller 2009) and basic concepts of IT security in immersive virtual realities (Puttawong, Visoottiviseth, and Haga 2017), none of them focus on the idea of asymmetric encryption. As shown before, metaphors can contribute to students' understanding but have often some constraints or flaws for carrying out the metaphors in real activities/tasks for students. Virtual reality technology can provide a useful tool to get rid of the constraints of the actual reality (Bricken 1990) in order to create engaging learning environments.

3. THE DESIGN OF FLUXI'S CRYPTIC POTIONS

The approach for our immersive EVE Fluxi's Cryptic Potions, which was developed using Unity, combines the original Computer Science Unplugged activity, where the students communicate in an unknown network and send each other messages (or chocolate bars) with the mixing colors idea. Our medieval setting takes the player into a castle chamber where he/she encounters Fluxi, a carrier dragon, who delivers messages to and from the student. The player receives a letter from a friend, Nikolay, who asks the player if he/she will be at Sir Dance-A-Lot's (the metaphorical man-in-the-middle) party this evening. Fluxi asks the player to reply to Nikolay by telling him that he/she wants to attend but has not received an invitation yet. Fluxi brings the letter to the post office (simulating the network structure) and returns with an encrypted response. Fluxi explains that the post office provides each participant of the network with two cryptic potions: a private and a public potion. While all potions cipher messages, the encryption can only be reversed by using the corresponding other potion. The public potion of each participant is stored publicly in the post office and everyone can get a copy of it. In contrast, the recipe of the private potion is secret and only known by the user. After explaining the benefits of this asymmetric encryption process, he instructs the user to decrypt Nikolay's message by using his/her private potion.

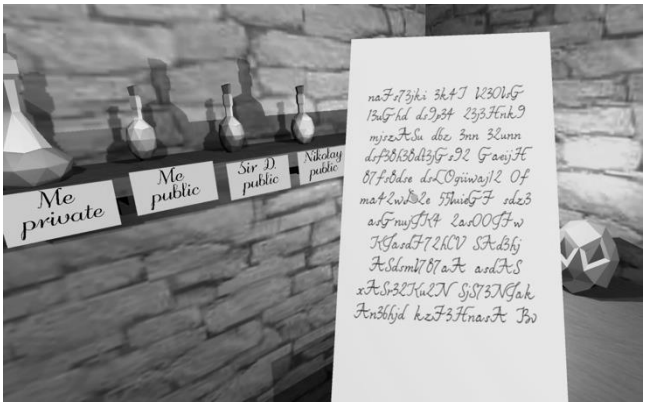


Figure 4. Fluxi's Cryptic Potions

Nikolay informs the player that Princess Isolde, Prince Charming, and Fluxi's aunt Gertrude will be at the party as well. He notes that the player always wanted to dance with one of them and that he/she should write a letter to the person of interest. But Nikolay also warns the player that Sir Dance-A-Lot wants to dance with all of them which is why the communication should be kept secret and, therefore, all messages should be encrypted. The player writes a new, encrypted letter to Nikolay (with Nikolay's public potion) and agrees that it would be a good idea to encrypt the messages. After delivering this letter to the post office, Fluxi returns with the invitation of Sir Dance-A-Lot. The invitation seems to be ciphered as well, and Fluxi explains that Sir Dance-A-Lot signed the invitation so that everyone knows that the message must be from him. After reasoning why this process is secure (in terms of authentification), the player adds Sir Dance-A-Lot's public potion to the letter in order to decipher it. The player writes a message saying that he/she will attend the party and signs the letter with his own private potion. After delivering the letter, Fluxi asks the player who it is he/she wants to dance with. As the dialogues and letters are quite similar and the tasks stay the same, we will present the scenario for a player who chose Prince Charming. The player writes a message to Prince Charming asking him for a dance. Fluxi gets the prince's public potion from the post office and explains that this potion can encrypt letters for Prince Charming and decipher signed letters from him as well. The player encrypts the message with Prince Charming's public potion (to avoid Sir Dance-A-Lot reading it) and gives the letter to Fluxi. After returning from the post office, Fluxi gives the player a signed response from Prince Charming: He does not believe the player's authenticity as the player encrypted the message for Prince Charming, but did not sign it. Hence, the player has to rewrite his/her letter, encrypt it with Prince Charming's public potion and sign it with his/her own private potion. After resending the letter, Fluxi returns with a signed and encrypted reply from Prince Charming, telling the player that the prince waited an eternity for this question and would be glad to dance with him/her.

The controls in Fluxi's Cryptic Potions were gaze-based via point-and-click. The player could pick up potions and letters, write new messages, and talk to the dragon. The player could not move or teleport, resulting in him/her staying in the same room all the time. In all technological settings, the player sat on a chair, simulating the same position as in the EVE.

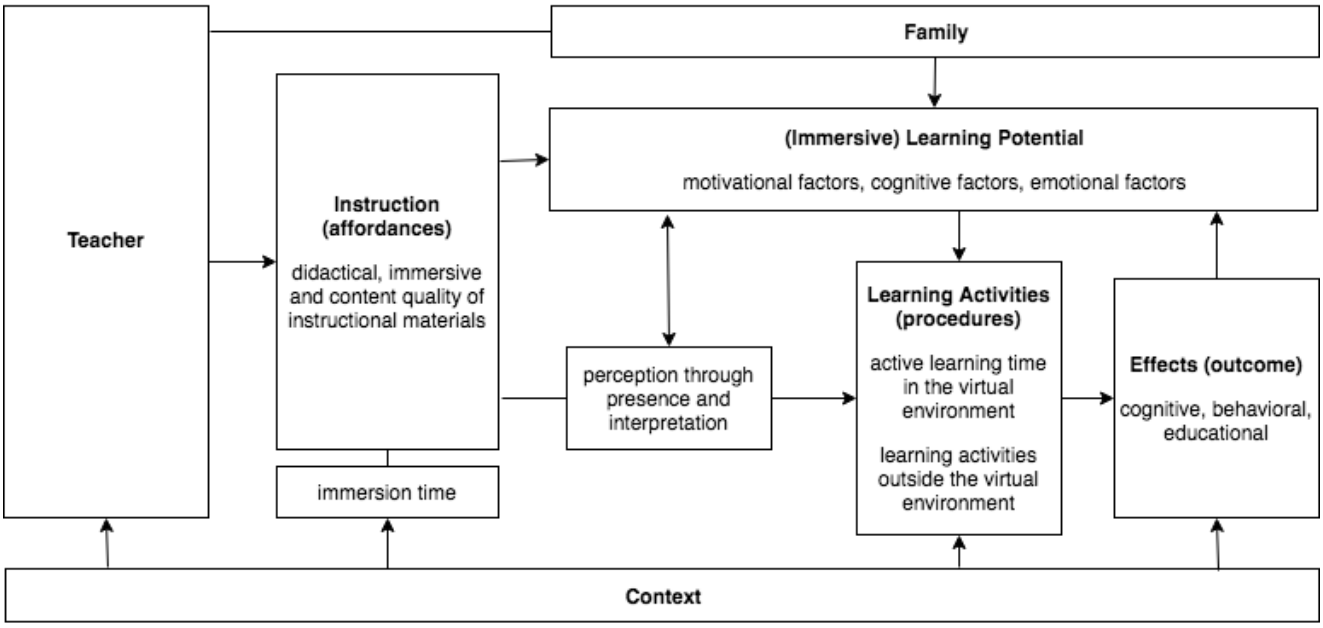


Figure 2. The Educational Framework for immersive Learning (EFiL) by Dengel and Mägdefrau, 2018

4. LEARNING IN IMMERSIVE ENVIRONMENTS

Necessary to consider the factors influencing learning outcomes, especially when teaching and learning with immersive technology like virtual reality, e.g. immersion as a quantifiable description of technology (Slater et al. 1999) and presence as the feeling of 'being there' (e.g. Biocca 1997). Dalgarno and Lee (2010) identify representational fidelity (the display of the environment, the display of view changes and object motion, the object behavior, the representation of the user, the provided spatial audio, and the kinesthetic and force feedback) and the learner interaction (embodied actions, embodied verbal and non-verbal communication, control of environment attributes and behavior, and construction/scripting of objects and behaviors) as affordances of 3D learning environments. These characteristics of EVEs can induce the construction of identity, a sense of presence, and co-presence inside of the user. These user characteristics, in turn, affect the learning benefits (spatial knowledge representation, experiential learning, engagement, contextual learning, and collaborative learning) through the afforded learning tasks provided by 3D EVEs. By combining Dalgarno and Lee's framework with the idea of presence being a person-specific, unique characteristic of EVEs (for a discussion about this, see Mikropoulos 2006), Dengel and Mägdefrau (2018) introduce the Educational Framework for Immersive Learning (EFiL, Fig. 2). The EFiL localizes the factors immersion and presence in the educational supply-use-framework for the explanation of scholastic learning presented by Helmke (2014) and provides a solid basis for explaining learning outcomes in immersive and non-immersive EVEs: According to the EFiL, learning activities in EVEs "are determined through the (immersive) learning potential [including motivation, cognitive factors, and the emotional state of the learner], the context of the learner, the perception of the didactical, immersive and content quality of the instructional materials at a certain level of presence and the interpretation of these materials. The factors influencing immersive learning are related among each other and (especially in scholastic environments) affected by the family and the teacher of the learner" (Dengel and Mägdefrau 2018, p. 614). Dengel (2020) notes that the EFiL can be used as a framework for explaining learning in EVEs in general, but, in order to understand the relations between the factors, one has to consider already established research from the educational sciences and psychological research.

The potential of immersive media has been acknowledged for the use in Computer Science Education (Dengel, 2019): By taking on the idea of Computer Science Unplugged (introduced by Bell and Fellows, see e.g. Bell, Rosamond, and Casey 2012), the concept of Computer Science Replugged thinks of ways to integrate immersive technology to enhance existing Computer Science Education activities and to generate new activities in virtual environments while preserving most of the key characteristics of an Unplugged activity (kinaesthetic, fun and engaging with a sense of story to the activities, see Bell et al. 2009): "By using immersive technology, the induced feeling of presence can provide a perception of non-mediation and, therefore, a first-hand experience" (Dengel, 2019, p. 2).

5. METHOD

By following the assumptions of the EFiL (cognitive abilities are modeled through the previous scholastic performance in German, which is the students first language, and Maths) and the constraints of the factors' relations formulated by Dengel (2018), it is hypothesized that

(1) The level of immersion predicts the user's level of physical presence.

(2) The user's emotional state predicts his/her sense of physical presence (a: Stronger positive emotions

increase presence. b: Stronger negative emotions decrease presence.).

(3) The user's motivation predicts his/her pre-test performance (a: Intrinsic motivation increases pre-test performance b: Extrinsic motivation decreases pre-test performance.)

(4) A higher sense of presence predicts a higher post-test performance.

(5) A better result in the student's pre-test predicts a better post-test performance.

(6) The user's cognitive abilities predict his/her post-test performance (a: A higher previous scholastic performance in German increases post-test performance. b: A higher previous scholastic performance in Maths increases post-test performance.).

As noted in section 4, the factors marked as independent variables here are related to each other. For the path analysis approach presented in this study, the relations suggested by popular theories like the Expectancy-Value-Theory (Ryan and Deci 2000), the Control-Value-Theory (Pekrun 2000), and meta-studies like Hattie (2008) were considered for formulating the research model.

5.1. Sample and Procedure

78 students (36 female, 4 missing values) between the age of 13 and 16 from different classes of an Austrian school took part in the study. Asymmetric encryption was not part of their computer science classes before. Their performances in the subjects Maths (M = 2.51, SD = .91) and German (M = 2.42, SD = .92), which could be reported anonymously by the students on their parents' notice, were average (with 1 being the highest and 6 the lowest grade).

A week after completing the pre-test and the motivation questionnaire, the students took part in the experiment in groups of four to six which were assigned to different technology settings. This study was part of a bigger study, providing three EVEs in total. First, they filled out an emotional state questionnaire and then waited until the next VR experience was available. Each student was provided with another technology for each EVE. After completing each VR experience, they filled out the corresponding presence questionnaire and post-test.

5.2. Instruments

An adapted version of the Slater-Usoh-Steed presence questionnaire (Slater, Usoh, and Steed 1994) was used where the mean score was calculated out of six questions on a seven-point Likert scale (M = 4.14, SD = 1.56, α = .91). Further, an emotional state questionnaire of Titz (2001) was used, assessing academic emotions on a 6-point Likert scale (positive emotions: M = 2.91, SD = .98, α = .73; negative emotions excluding fear: M = .69, SD = .68, α = .68). The context motivation questionnaire assessed intrinsic motivation (M = 3.10, SD = 1.02, α = .85), identified motivation (M = 3.34, SD = .97, α = .79), introjected motivation (M = 2.39, SD = 1.07, α = .76), and extrinsic motivation (M = 2.70, SD = 1.02, α = .65) for learning in the subject Computer Science (original version by Hanfstingl (Hanfstingl et al. 2010), adapted and evaluated for the subject Computer Science by Dengel, 2020) on a 5-point Likert scale. For the path analysis, only intrinsic motivation and extrinsic motivation were analyzed as they tended to show the greatest difference in motivation for learning computer science between the students. The pre- and post-tests were the same and assessed the students' understanding skills: The first task and the second task asked the students to explain why a specific key was used in order to sign/encrypt a message, resulting in a performance test of four points total. The students scored better in the post-test (M = 1.83, SD = 1.20, α = .68) than in the pre-test (M = 1.32, SD = 1.16; the pre-tests scale reliability was not calculated as the tasks were supposed to be new to the students). A third task where the student had to insert the correct keys into blanks was removed due to a reduction of the overall scale reliability of the post-test. The Fluxi's Cryptic Potions EVE was presented with three different technologies: a laptop, a mobile VR (using a Moto Z smartphone and a Daydream View headset), and an HTC Vive.

5.3. Results

While there were no outliers in the sample's results, the results of the post-test, the pre-test, the extrinsic motivation scale, the negative emotions scale, and the scholastic performances in German and Maths were not equally distributed (Shipiro-Wilk method, $p < .05$). Non-parametric analysis was used where it was applicable.

An ANOVA could show significant differences ($F = 22.68$, $p < .0005$) between the induced levels of presence for the different technologies (laptop: M = 3.11, SD = 1.23; Mobile VR: M = 4.07, SD = 1.54; HTC Vive: M = 5.40, SD = .93). A Gabriel (used because of slightly varying group sizes) post-hoc test could show that presence in the laptop setting at a significance level of $p < .05$ from the Mobile VR setting and at a significance level of $p < .0005$ from the HTC Vive level. The level of presence was significantly different from the Mobile VR level as well ($p < .01$). A multiple linear regression model including immersion, positive emotions, and negative emotions as predictors of presence was calculated, but only immersion was included as the only predictor of presence with $\beta = .63$, $p < .0005$) A predictive effect of positive emotions and negative emotions on presence was not significant (this relation will be explored further in the path analysis).

To predict the pre-test performance, another multiple linear regression model [corrected $R^2 = .06$, $F(2, 70) = 3.27$, $p < .05$] was calculated, including intrinsic motivation ($\beta = .12$, $p > .05$) and extrinsic motivation ($\beta = -.23$, $p > .05$).

The students' post-test performance could be predicted [corrected $R^2 = .26$, $F(4, 59) = 6.59$, $p < .0005$] by the factors presence ($\beta = .24$, $p < .05$), pre-test performance ($\beta = .50$, $p < .0005$), the previous scholastic performance in German ($\beta = .29$, $p < .05$) and the previous scholastic performance in German ($\beta = -.27$, $p < .05$).

As noted in section 4, the factors that predict learning achievement are related to each other. Therefore, a path analysis was calculated, integrating suggested relations within and between the different theoretical constructs.

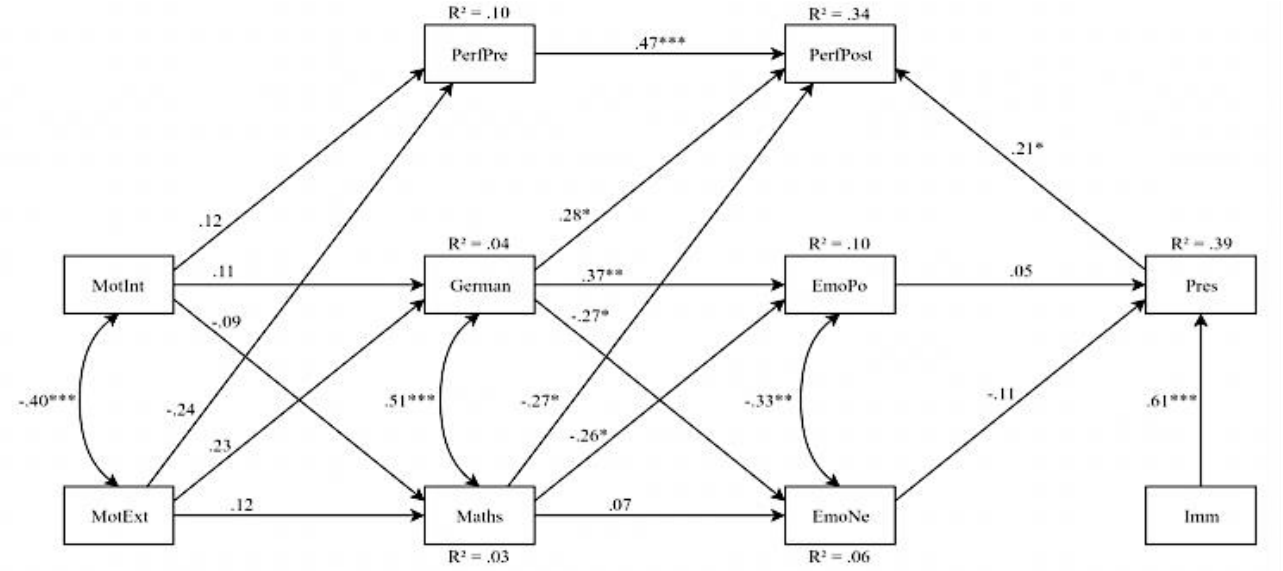


Figure 3. Path Analysis Showing Effects between Intrinsic Motivation (MotInt), Extrinsic Motivation (MotExt), Scholastic Performances in German and Maths, Positive Emotions (EmoPo), Negative Emotions (EmoNe), Presence (Pres), Immersion (Imm), Pre-Test Performance (PerfPre), and Post-Test Performance (PerfPost)

Levels of Significance: *p<.05; **p<.01; ***p<.001

The path analysis showed good fit indices (Chi^2 p = .30, RMSEA = .041, CFI = .968, TLI = .946, SRMR = .073). Figure 3 shows the correlations between and predictive effects of the different factors. Intrinsic motivation and extrinsic motivation are correlated negatively; scholastic performances in German and in Maths are correlated positively; positive and negative emotions are correlated negatively. A higher scholastic performance in German predicts lower positive emotions; a higher scholastic performance in Maths predicts higher positive emotions; a higher scholastic performance in German predicts higher negative emotions (these effects are inverse in the path analysis due to the fact that the best scholastic performance is grade 1, the worst is grade 6). A higher level of immersion predicts a higher level of presence. Presence, pre-test performance, as well as scholastic performances in German and Maths predict the post-test performance.

5.4. Discussion

Regarding the hypotheses, H1, The level of immersion predicts the user's level of physical presence, can be maintained as the ANOVA and the post-hoc tests show significant differences. H2, The user's emotional state predicts his/her sense of physical presence (a: Stronger positive emotions increase presence. b: Stronger negative emotions decrease presence.) could not be verified as emotions were not identified as predictors of presence. Regarding H3, The user's motivation predicts his/her pre-test performance (a: Intrinsic motivation increases pre-test performance b: Extrinsic motivation decreases pre-test performance.), the sample was too small to find significant effects, this hypothesis has to be investigated further. H4 assumed that A higher sense of presence predicts a higher post-test performance. The effect of presence on post-test performance was found to be significant. Therefore, presence poses an important predictor of learning outcomes for the presented EVE. H5, A better result in the student's pre-test predicts a better post-test performance, could be verified as well in this study and can be maintained. For H6, the user's cognitive abilities predict his/her post-test performance (a: A higher previous scholastic performance in German increases post-test performance.

b: A higher previous scholastic performance in Maths increases post-test performance.), both subhypotheses can be maintained as the study showed significant effects of the previous scholastic performance on the post-test learning outcomes.

Even though some of the results are not significant (as assumed, due to the small sample size), the general idea of the EFiL, which was used for the selection of the hypotheses, was found to be true: Presence is an important predictor of learning outcomes and is influence by immersion. Even though the learning outcomes are influenced by many factors, the level of immersion is not one of them.

6. CONCLUSIONS

Teaching the basic idea of asymmetric encryption using VR technology has the opportunity that it is possible to realize metaphors and analogies that are impossible to carry out in the physical reality. Still, using metaphors for explaining general ideas is tricky: It is the role of the teacher to explain the metaphor/analogy before or after the activity. Furthermore, if the teacher uses the EVE as an introduction to asymmetric cryptography in advanced classes, dealing with computational complexity and mathematical background becomes crucial as well. Doing so, VR experiences should be seen as an addition to existing teaching methods, not as substitutes. They have to be included at the right point in the learning process in order to show their potential. While the VR activity was effective in terms of learning outcomes, it is, by now, not possible to conclude that using the VR environment has benefits over

real activities. In addition, the EVEs can be enhanced in multiple ways: Providing possibilities to interact with other students, for example to send each other secret messages or to intercept other students' messages and try to decrypt them would make the EVE more fun and motivating while adding more interaction possibilities. That said, using VR in cryptography education has its merits, but also poses challenges for the teacher. Future studies could focus on exploring the benefits and challenges of using this metaphorical VR representation in comparison to real activities or traditional learning approaches. Having this in mind, it would also be interesting to explore, what other topics in CS education can benefit from the use of immersive technology in the classroom and how immersive technology, in general, can enhance learning.

7. REFERENCES

Art of the Problem. (2012). *Public key cryptography - diffiehellman key ex-change (full version)*. Retrieved May 27, 2019, from https://www.youtube.com/watch?v=YEBfamvdo

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer Science Unplugged: School Students Doing Real Computing Without Computers. *The New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Bell, T., Rosamond, F., & Casey, N. (2012). Computer Science Unplugged and related Projects in Math and Computer Science Popularization. In M. R. Fellows H. L. Bodlaender (Eds.), *The Multivariate Algorithmic Revolution and Beyond*, 398–456.

Biocca, F. (1997). The Cyborg's Dilemma: Progressive Embodiment in Virtual Environments. *Journal of Computer-Mediated Communication*, *3*(2), JCM324.

Bricken, W. (1990). *Learning in Virtual Reality*. Seattle.

Dalgarno, B., & Lee, M. J. W. (2010). What are the Learning Aordances of 3-D Virtual Environments? *British Journal of Educational Technology*, *41*(1), 10–32.

Dengel, A. (2019). Computer Science Replugged: What Is the Use of Virtual Reality in Computer Science Education? *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE'19).* Association for Computing Machinery, New York, NY, USA, Article 21, 1–3.

Dengel, A., & Mägdefrau, J. (2018). Immersive Learning Explored: Subjective and Objective Factors Influencing Learning Outcomes in Immersive Educational Virtual Environments. *Proceedings of 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*. IEEE, 608-615.

Diffie, W., & Hellman, M. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654.

Fekete, S. P., Morr, S. (2018). *Public key krypto*. Retrieved May 27 2019, from https://idea-instructions.com/public-key

Hanfstingl, B., Almut, T., Andreitz, I., & Müller, F.H. (2010). Evaluationsbericht Schüler-und Lehrerbefragung 2008/09. *Interner Arbeitsbericht. Klagenfurt, Institut für Unterrichts-und Schulentwicklung.*

Hattie, J. A. (2008). *Visible Learning: A Synthesis of over 800 Meta-analyses Relating to Achievement.* Routledge.

Helmke, A (2014). *Unterrichtsqualität und Lehrerprofessionalität: Diagnose, Evaluation und Verbesserung des Unterrichts*. Seelze-Velber: Klett Kallmeyer.

Mikropoulos, T. A. (2006). Presence: A Unique Characteristic in Educational Virtual Environments. *Virtual Reality*, *10*(3-4), 197–206.

Pekrun, R. (2000). A Social-cognitive, Control-value theory of Achievement Emotions. Motivational Psychology of Human Development: Developing Motivation and Motivating Development. *Advances in Psychology*, *131*, 143–163.

Puttawong, N., Visoottiviseth, V., & Haga, J. (2017). Vrfiwall Virtual Reality Edutainment for Firewall Security Concepts. *Proceedings of 2017 2nd International Conference on Information Technology (INCIT)*. IEEE, 1-6.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, *21*(2), 120-126.

Ryan, R., & Deci, E. (2000). Self-determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-being. *American Psychologist*, *55*(1), 68.

Slater, M., Linakis, V. Usoh, M., & Kooper, R. (1999). Immersion, Presence, and Performance in Virtual Environments: An Experiment with Tri-Dimensional Chess. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. ACM, 163-172.

Slater, M., Usoh, M., & Steed, A. (1994). Depth of Presence in Virtual Environments. *Presence: Teleoperators and Virtual Environments*, *3*(2), 130–144.

Sturgeon, T., Allison, C., & Miller, A. (2009). Exploring 802.11: Real Learning in a Virtual World. *Proceedings of Frontiers in Education Conference.* IEEE, 1–6.

Titz, W. (2001). Emotionen von Studierenden in Lernsituationen: Explorative Analysen und Entwicklung von Selbstberichtskalen-Anhang.

UC Computer Science Education. (2008). *Computer science unplugged - the show*. Retrieved May 27, 2019, from https://www.youtube.com/watch?v=VpDDPWVn5Q&t=10s

Voss, G.B., Nunes, F.B., Muhlbeier, A.R., & Medina, R.D. (2013). Context-aware Virtual Laboratory for Teaching Computer Networks: a proposal in the 3d opensim environment. *XV Symposium on Virtual and Augmented Reality*, 252–255.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Comparison of the Learning Behaviors of the Third Grader Students Integrating Robots and the Computational Thinking Board Game in Singapore and Taiwan

Yi-Sian LIANG[1], Ting-chia HSU[2*]
National Taiwan Normal University, Taiwan
mianmian0202@gmail.com, ckhsu@ntnu.edu.tw

ABSTRACT
The purpose of this study is to explore the use of educational robots and computing thinking board games by primary and middle school students in different countries, and to explore whether there are differences in learning behaviors during the learning process. It was found that the primary school students in Singapore had the highest number of behaviors in irrelevant courses, and the same textbook content was applied to the primary three in Taiwan. It can be seen that Taiwanese students tend to spend time talking with competitors. This phenomenon can increase students' oral communication and enhance their learning fun during the discussion. Singaporean students rank first in behaviors that are not related to the course. It is speculated that the content of the textbooks may be too difficult, which may lead to restrictions on communication. This study suggests that textbooks can be moved to other grades in Singapore in the future to help Singaporean students improve the same learning effectiveness as Taiwanese students.

KEYWORDS
learning analysis, educational robots, computational thinking, board game

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

比較新加坡和台灣小學三年級學生整合機器人與運算思維桌遊之學習行為

梁儀嫻[1]，許庭嘉[2*]
國立臺灣師範大學 科技應用與人力資源發展學系，臺灣
mianmian0202@gmail.com, ckhsu@ntnu.edu.tw

摘要
本研究旨在探討不同國家小三學生使用教育機器人結合運算思維桌遊，探討學習過程中的學習行為是否有差異。結果發現新加坡小學的學生在無關課程的行為次數居所有行為之冠，而相同教材內容應用在台灣小三上，發現與組外對談的行為次數居所有行為之冠。可見台灣學生傾向花時間和競爭同儕對話，此現象可增加學生進行口語交流，在討論過程中提升他們對學習的有趣性。新加坡學生在與課程無關學習的行為位居之冠，推測可能教材內容太難，導致溝通形成限制。本研究建議未來可將教材移至新加坡其他年級，幫助新加坡學生提升與台灣學生相同學習成效。

關鍵字
行為分析；教育機器人；運算思維；桌上遊戲

1. 前言
在數位時代中，每個人都應該具備運算思維能力（Korkmaz, Çakir, & Özden, 2017）。運算思維（Computational thinking, CT）是學習者的基本技能，也是評估教育的關鍵因素（Zhong, Wang, Chen, & Li, 2016）。運算思維是新一代學習者必須掌握的一套解決問題的技能，才能在充滿由軟體驅動物體的數位時代中蓬勃發展（Román-González, Pérez-González, & Jiménez-Fernández, 2017）。

有學者發現機器人活動是一種有效的教學策略，可以提高人們對機器人的興趣，提高自我效能及與機器人一起教學，發展出對科學概念的理解並促進運算思維能力的發展（Jaipal-Jamani & Angeli, 2017）。機器人技術可以被當作是創造許多科學教育方法的〝工具〞，例如探究式學習和解決問題（Altin & Pedaste, 2013）。越來越多的使用社交技能的人型機器人在教育領域中的科學教育、特殊教育和外語教育等（Sisman, Gunay, & Kucuk, 2019）。

有學者發現，遊戲式學習可以幫助學習者避免無聊，因而獲得新的學習體驗環境（Rawendy, Ying, Arifin, & Rosalin, 2017）。過去二十年中，遊戲式學習的環境已經發展成為功能強大的學習工具，也引起了各種教育利益相關者期大的興趣（Groff, 2018）。透過遊戲式學習、問題導向學習、視覺化程式設計是可以潛在地幫助學習者在程式設計課程入門中有良好的技術表現（Topalli & Cagiltay, 2018）。

根據上述所提到的教育機器人已經應用於教育領域上，而遊戲式學習可以幫助學習者獲得一個新的學習環境，故本研究試圖探討對於在不同國家的學習者利用教育機器人進行遊戲式學習，在學習過程中的行為是否有所差異，並且針對遊戲式學習的過程進行行為序列分析。本研究將探討的研究問題為：不同國家的學習者，利用教育機器人進行遊戲式學習之間行為分析的差異為何？

2. 文獻探討
2.1. 運算思維
運算思維主要是一種思維和行動方式，可以透過使用特定的技能加以展示，然後成為一個可以做基礎評估 CT 實作本位測量的技能（Shute, Sun, & Asbell-Clarke, 2017）。可以將運算思維簡單地定義為能夠使用電腦解決生活中產生問題所必須具有的知識、技能和態度（Korkmaz et al., 2017）。

CT 主要用在程式設計和電腦科學的活動，也還依些研究與其他主題有關。同時，大多數研究在 CT 活動中採用專題式學習、問題導向學習、合作學習和遊戲式學習（Hsu, Chang, & Hung, 2018）。

2.2. 教育機器人
教育機器人的配件及一些特殊學習和教材的使用是目前科技必備的。學者也已經證明，當應用在自然數學的學科和科技教育領的學科連結時，學習效果會提高（Ospennikova et al., 2015）。

教育機器人（Educational Robotics, ER）的概念不應該只關注在分開、獨立的主題，反而應該作為一種綜合方法應用，以促進對不同領域和領域的整體理解和接受（Kandlhofer & Steinbauer, 2016）。教育機器人技術是用在學習、運算思維、程式設計和工程學的一種轉換工具，在 K-12 教育當中被視為 STEM 學習的關鍵因素（Eguchi, 2014）。

2.3. 遊戲式學習
隨著新流行的技術發展，教育界很快開始探索如何將遊戲用於教學上（Godwin-Jones, 2016）。對於教育遊戲的設計，遊戲的挑戰應該可以跟上學習者的成長能力和學習，以辨認可遊戲式學習學習環境中可以持續學習（Hamari et al., 2016）。

在學校環境中整合遊戲式學習的主要的挑戰之一是幫助學習者將遊戲中學習到的知識與學校中所學習到的知識連繫在一起（Barzilai & Blau, 2014）。有學者發現遊戲式學習已經成功的應用，也發現遊戲式學習的可以降低考試焦慮和增加參與度（Kiili & Ketamo, 2018）。

3. 研究方法

3.1. 研究對象

本次實驗共 54 名三年級學習者參加這項研究。一組為新加坡某國小三年級 26 名學習者學習第二語言（Second Language, L2），過程中使用的教材語言為華語文；另一個為台灣北部某國小三年級 28 名學習者學習外語（Foreign Language, FL），過程中使用的教材語言為英語，兩組皆沒有玩過運算思維桌遊，皆利用教育機器人進行組內合作學習。

參與者平均年齡為 9 歲。為了保護實驗對象，實驗中的實驗對象們的參與皆得到了父母的同意，並在研究過程中隱藏他們的個人資訊來保護實驗對象。此外，他們知道參加是自願的，過程中若有不適可以隨時退出研究。

3.2. 序列分析之編碼系統

為了探討學習者在遊戲式學習過程中的學習行為，針對被觀察者的行為詳實記錄下來，並參考相關研究之編碼系統。在學習過程中將學習狀況分為三類：運算思維行為、語言行為和其他行為，如*表 1* 所示。

表 1 行為編碼：組內競賽活動之行為分析編碼

類別	代碼	意義	範例
運算思維編碼	PP(People&People)	組內對談	同組的兩個人在對談
	PC(People Commnication)	組外對談	與別組在對談
	PR(People & Robot)	使用機器人	掃描卡牌使機器人移動
	ID(Individual Decision)	個人使用任務卡	使用石頭、砂土…等，
	CD(Cooperation Decision)	共同使用任務卡	放置任務卡上
	AT(Algorithm)	使用卡牌	排除卡牌(前進、左轉…等)
	PM(Physical Message)	姿體表達	行為左右轉、手勢左右轉…等
	AG(Abstraction General)	資料簡化或用其他方式表達	單程式方法便迴圈方式表達
語言編碼	LI(Learning Interaction)	被觀察者正在練習口語互動	自己口語互動
	PLI(People learning)	正有其他人在指導被觀察者口語互動	教師教觀察者學習口語互動
	NS(No Speaking)	不會口語互動	沒有講任何語言
	YS(Yes Speaking)	會正確口語互動	單字、句型接正確
	LT(Listen to teacher)	聽教師講解	教師講解遊戲規則
其他	IM(Irrelevant Message)	無關課程	發呆、離開座位…等
	SP(Separate)	組內做不同的事	各做各的事

4. 研究結果

本實驗將學習者利用教育機器人進行遊戲式學習的學習過程，進行行為分析的比較。依據影片紀錄，學習者的行為經過編碼及後續的序列分析，各獲得 494 個及 6588 個行為編碼，本研究進一步對 54 名學習者提出的 70282 種編碼進行行為頻率計算如*表 2* 所示。

表 2 兩國學習者之遊戲式學習行為出現比率

編碼	新加坡		台灣	
	次數	百分比(%)	次數	百分比(%)
PP	58	11.98%	764	14.25%
PC	67	13.84%	1123	20.94%
PR	74	15.29%	763	14.23%
ID	0	0.00%	85	1.58%
CD	6	1.24%	313	5.84%
AT	56	11.57%	795	14.82%
PM	0	0.00%	537	10.01%
AG	0	0.00%	0	0.00%
LI	18	3.72%	348	6.49%
PLI	42	8.68%	116	2.16%
NS	0	0.00%	0	0.00%
YS	0	0.00%	0	0.00%
LT	37	7.64%	408	7.61%
IM	89	18.39%	2	0.04%
SP	37	7.64%	109	2.03%

根據上表，新加坡小學的學習者中，前五名行為依序為無關課程（IM）、使用機器人（PR）、與組外對談（PC）、組內對談（PP）、使用卡牌（AT），而在台灣小學的學習者中，前五名行為依序為與組外對談（PC）、使用卡牌（AT）、組內對談（PP）、使用機器人（PR）、姿體表達（PM）；由此可知，學習者在使用教育機器人進行遊戲式學習時，皆會使用使用機器人（PR）、與組外對談（PC）、組內對談（PP）、使用卡牌（AT）。唯一的差別為新加坡小學的學習者較常做一些與課程無關（IM）的事情，而台灣小學的學習者較專注在課堂上，並會透過肢體表達（PM），完成任務。

4.1. 行為分析

為確保過程的一致性，以相同性質背景的人員，各分配50%的影片進行分析。在定量分析後，針對結果比較每個碼之間的關聯性並繪成行為編碼圖，箭頭方向為起始編碼至目標編碼，線上數字即表示該轉換行為關係的 Z 值，Z 值大於 1.96 代表著行為序列達到顯著水準（$p<0.05$）（Bakeman & Gottman, 1997），如*圖 1*、*圖 2* 所示。

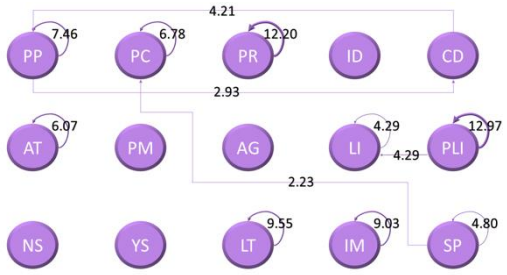


圖 1 新加坡小學學習者之行為編碼圖

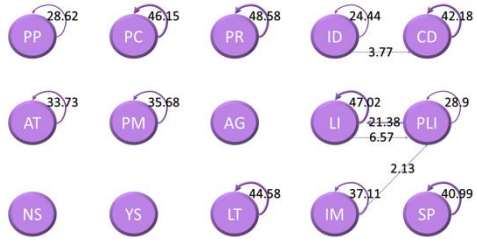


圖 2 台灣小學學習者之行為編碼圖

從*圖 1* 中可以發現，學習者的行為模式可以分為有三個獨立的關聯性。這三個關聯分別為 CD-PP-CD（共同使用任務卡、組內對談、共同使用任務卡）；SP-PC（即各做各的和與別組組員對談）；PLI-LI（即正有其他人在指導被觀察者如何說（英語）和被觀察者正在練習口語互動（自己說英語））。為當學習者在進行合作使用任務卡（CD）後會與組員討論（PP），接著在一起合作使用任務卡（CD），而當有其他人在指導學習者口語互動（PLI）後，學習者會透過指導的過程後練習口語互動（LI），另外，當學習者兩兩一組在做不一樣的事（SP）時，會意識到似乎該回到課堂中，並與組外的組員進行討論（PC）。

從*圖 2* 中可以發現，學習者的行為模式可以分為有三個獨立的關聯性。這三個關聯分別為 ID-CD（即個人使用任務卡和合作使用任務卡）；IM-PLI-LI（即無關課程、正有其他人在指導被觀察者如何說（英語）、被觀察者正在練習口語互動（自己說英語））；LI-PLI（即被觀察者正在練習口語互動（自己說英語）和正有其他人在指導被觀察者如何說（英語））。當學習者在進行合作使用任務卡（CD）後會與個人使用任務卡（ID），而當有其他人在指導學習者口語互動（PLI）後，學習者會透過指導的過程後練習口語互動（LI），若學習者講述的不正確的話，教師會進一步的指導學習者口語互動，另外在學習者若在做無關課程（IM）的事的時候，教師將會再次指導學習者進行口語互動。

從兩國的行為分析編碼圖來看，新加坡小學的學習者會共同合作思考完畢後與組員討論並一起完成任務；而台灣小學的學習者較傾向於獨立思考的部分，不過當學習者獨立思考後，會再與組員討論並共同完成任務。這部分可以說明利用教育機器人進行遊戲式學習的過程，可以增加學習者的合作力。另外，兩國小學的學習者之中，當有其他人在指導學習者口語互動時，學習者會接著練習口語互動，值得注意的事，台灣小學的學習者若講述錯誤的口語的話，指導者將會再次指導學習者進行口語互動，這幫助學習者更完整且更快速的完成任務。

5. 結論與未來展望

隨著科技應用的興起，教育機器人也跟著盛行，教育機器人已經越來越多地融入在從幼兒時期到高等教育的教育領域當中，教育機器人活動將與課程的學習目標或技能的發展中，保持一致性，例如：協作、解決問題、創造力、批判性思維和運算思維（Komis, Romero, & Misirli, 2016）。近年來，因為行為分析的的認可和對行為分析服務的需求已經大大提升，透過閱讀材料並按照課程領域分類（例如：倫理學、行為主義、單科研究方法），以便為新程式開發和語言翻譯工作提供資源（Pastrana et al., 2018）。故本研究利用教育機器人進行遊戲式學習幫助學習者學習，並透過行為分析編碼表觀察學習者的行為，發現兩國小學的學習者之中，當有其他人在指導學習者口語互動時，學習者會接著練習口語互動，值得注意的事，台灣小學的學習者若講述錯誤的口語的話，指導者將會再次指導學習者進行口語互動，這可以幫助學習者更加完整且更快速的完成任務。

在學習過程中，本研究發現新加坡小學的學習者在無關課程（IM）的部分佔據第一名，推測可能是因為新加坡小學的學習者在校內華語分班的部分較為後段班，而教材內容所使用的華語部分可能對新加坡小學的學習者來說較為困難，導致學習者遇到問題時不敢詢問不知道該如何做，才會做出與課程無關的事情，故本研究建議未來可以朝著將將教材更改為適合學習者的內容，並針對調整整體教學流程，幫助學習者達成提升學習成效的部分。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 108-2511-H-003 -056 -MY3 的部分補助。

7. 參考文獻

Altin, H., & Pedaste, M. (2013). Learning Approaches to Applying Robotics in Science Education. *Journal of Baltic Science Education, 12*(3), 365-377.

Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis*. Cambridge University Press.

Barzilai, S., & Blau, I. (2014). Scaffolding Game-based Learning: Impact on Learning Achievements, Perceived Learning, and Game Experiences. *Computers & Education, 70*, 65-79.

Godwin-Jones, R. (2016). Emerging Technologies Augmented Reality and Language Learning: From Annotated Vocabulary to Place-based Mobile Games. *Language Learning & Technology, 20*(3), 9-19.

Groff, J. S. (2018). The Potentials of Game-based Environments for Integrated, Immersive Learning Data. *European Journal of Education, 53*(2), 188-201.

Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging Games Help Students Learn: An Empirical Study on Engagement, Flow and Immersion in Game-based Learning. *Computers in Human Behavior, 54*, 170-179.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to Learn and How to Teach Computational Thinking: Suggestions Based on a Review of the Literature. *Computers & Education, 126*, 296-310.

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of Robotics on Elementary Preservice Teachers' Self-efficacy, Science Learning, and Computational Thinking. *Journal of Science Education and Technology, 26*(2), 175-192.

Kandlhofer, M., & Steinbauer, G. (2016). Evaluating the Impact of Educational Robotics on Pupils' Technical-and Social-skills and Science Related Attitudes. *Robotics and Autonomous Systems, 75*, 679-685.

Kiili, K., & Ketamo, H. (2018). Evaluating Cognitive and Affective Outcomes of a Digital Game-based Math Test. *IEEE Transactions on Learning Technologies, 11*(2), 255-263.

Komis, V., Romero, M., & Misirli, A. (2016). A Scenario-Based Approach for Designing Educational Robotics Activities for Co-creative Problem Solving. *Proceedings of International Conference EduRobotics 2016*. Cham: Springer, 158-169.

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.

Pastrana, S. J., Frewing, T. M., Grow, L. L., Nosik, M. R., Turner, M., & Carr, J. E. (2018). Frequently Assigned Readings in Behavior Analysis Graduate Training Programs. *Behavior Analysis in Practice, 11*(3), 267-273.

Rawendy, D., Ying, Y., Arifin, Y., & Rosalin, K. (2017). Design and Development Game Chinese Language Learning with Gamification and Using Mnemonic Method. *Procedia Computer Science, 116*, 61-67.

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.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying Computational Thinking. *Educational Research Review, 22*, 142-158.

Sisman, B., Gunay, D., & Kucuk, S. (2019). Development and Validation of an Educational Robot Attitude Scale (ERAS) for Secondary School Students. *Interactive Learning Environments, 27*(3), 377-388.

Topalli, D., & Cagiltay, N. E. (2018). Improving Programming Skills in Engineering Education through Problem-based Game Projects with Scratch. *Computers & Education, 120*, 64-74.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An Exploration of Three-dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research, 53*(4), 562-590.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Subject Learning and Teaching in K-12

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

On the Integration of Learning Mathematics and Programming

Dan KOHEN-VACS[1*], Chronis KYNIGOS[2*], Marcelo MILRAD[3*]
[1]Holon Institute of Technology, Israel
[1,2,3]Linnaeus University, Sweden
[2]National and Kapodistrian University of Athens, Greece
mrkohen@hit.ac.il, kynigos@ppp.uoa.gr, marcelo.milrad@lnu.se

ABSTRACT
In the field of education, there has been recent attention and call for transdisciplinary approaches related to learning mathematics and programming in schools. Despite the advent of theory and tools for such an approach, there is still a lack of a common ground and implicitness in the understanding of what exactly this would entail amongst teachers and curriculum designers. In this paper, we present a theoretical discussion in the light of our ongoing efforts to develop a more elaborated and precise language representing educational and epistemological values for integrating mathematics and programming. Accordingly, we provide an overview of our previous research efforts in this field followed by an elaborated example describing our approach. We conclude with a discussion addressing the pedagogical potential of our proposed ideas compared to the previous ones.

KEYWORDS
constructionism, subject matter integration, computational thinking, mathematics, programming

1. INTRODUCTION AND RATIONALE
The value of interdisciplinarity is a recurrent issue in educational settings and often at the core of the rationale for designing and implementing innovation in schools. The fields of Science, Technology, Engineering and Mathematics (STEM) for instance have been a subject for the application of integrative approaches to teach these different areas. Spite of these efforts, there have been many diverse understandings and views on the nature of STEM and on how to put together an educational activity where students generate joint meanings from two distinct domains while engaged in an interesting relevant activity. A number of underlying questions regarding the perception, the scope and the implementation of interdisciplinary educational activities remain implicit. For instance, is it more valuable to forge two - way connections between STEM disciplines at first? Is there a sense of thinking of one discipline as the field of application of another? Is there a sense in perceiving of one discipline serving the learning of another in an activity where both co-exist? For instance, what value does the practice of de-composing problems into simpler ones have in mathematics and in programming?

In this paper, we look closely at one example of such an interdisciplinary approach regarding mathematics, programming and computational thinking (CT). How can a mathematics teacher integrate a programming activity in their attempt to engage students in mathematical meaning making? Conversely, how can a computer science teacher can help students to write algorithms and programs employing the necessary mathematics concepts to do so? How can one discipline serve the understanding of the other and how can we design activities where students develop meanings jointly for concepts lying on both domains? To address some of these issues we review interdisciplinary approaches to learning mathematics and programming while trying to develop a more articulated view to think about the challenge of integrating them. Accordingly, we decompose the problem in three more focused ways of thinking about it, i.e. on how to design activities where one domain serves the other and vice versa and on how to think of the joint learning of these two domains. We use a special case for each of these sub-problems to analyze the different issues involved. Our proposed approach can be employed by teachers to design and think of activities integrating the two subjects in uni-disciplinary or interdisciplinary settings.

2. PROGRAMMING AT THE SERVICE OF MATHEMATICAL MEANING MAKING
It has been a long time now since a connection was made between learning to program and learning mathematics. This connection was firstly elaborated as early as in the 1960's by Seymour Papert as a theory of learning mathematics which he called *'Constructionism'*, i.e. the generation of mathematical meaning through programming a computer (Papert, 1980). Back then, programming was not yet perceived to have some value as a learning subject in general education. Indeed, Papert saw *Constructionism* as a mathematical learning activity involving the construction of and the tinkering with a digital artifact. He perceived of such an artifact as a public entity which can be shared, changed, discussed over. Such an artifact is thus never considered as 'complete' or as 'unquestionable', it is always under reform and improvement and it can be considered either as an object in itself or as a building block for higher order constructions (Kynigos, 2015). So, the initial connection between mathematics and programming in the field of education, rather than addressing a two-way connection, referred to the latter servicing the former, so to speak. Papert (1980) focused on the issue of the learning of mathematics by writing a computer program. He and others defined the Turtle as a means to create contours affording potentials to employ ideas from Euclidean and Cartesian Geometries. In addition, these affordances were proposed as absolute position and heading commands were included (Kynigos, 1992).

3. BIG IDEAS FROM MATHEMATICS
Papert coined the term 'big ideas' in mathematics to draw attention to some generic mathematical concepts which can be used as tools for solving problems in Turtle Geometry and

understanding the underlying structure of computational objects (Papert, 2000). Some examples of big mathematical ideas include generalization, fractions, ratio and proportions. Some of these ideas concepts related to angles, rate of change, periodicity. Others address class of objects defined by their properties as well as orientation in space. In this promising early work, programming was nevertheless considered in the role of servicing mathematical meaning-making. Not much attention was given to educational design aspiring on optimized intertwinement between learning mathematics and programming. An exception to this was Brian Harvey who developed his Berkeley Logo and a 3-volume book about 'programming Logo style' where Turtle Geometry only features as one chapter, the rest addressing issues of LISP-like learning to program (Sinclair & Moon, 1991). For more than a decade, substantial research was carried out with a focus on learner's mathematical meaning-making through programming. However, even though this resulted in the elaboration of a lot of potential yielded by children's expressions, explanations and exchanges, it also raised a debate as to the applicability and the effectiveness of such activity regarding the demands made by schooling and sustained educational institutions (Noss & Hoyles, 1996). This debate has hence remained unresolved. Moreover, in the 90s the interest in learning to program withered as if it had become obsolete in the wake of the spread of multimedia interfaces and the internet in its early form, drawing attention to individuals and collectives' use of digital media rather than their creations with tools affording constructionist activity.

3.1 Intertwining Applied CT with Math Challenges

Jansen et al., (2018), address the need to re-think what are the big programming ideas in connection to CT in a way parallel to the quest for the definition of mathematical big ideas which began back in the 80's. They take an epistemological point of view searching to define those big ideas in the foundational work of Turing, i.e. related to the process of learning to solve problems in the way computers do. But then again, there are few efforts re-connecting the learning of mathematics and programming. This is despite the recent elaboration of the wider value of approaches to STEM where technology and mathematics feature in a transdisciplinary setting which affords such efforts. Furthermore, in mathematics education, attention has progressed from highlighting the value of students' learning of mathematical concepts and ideas as an end in itself. There is now more emphasis on the learning of mathematics to involve the adoption of higher order mathematical processes. That is, to develop a disposition to mathematize their world by seeking for patterns, creating generalizations, looking for expression economy (Noss & Hoyles, 1996). In the same sense, with respect to programming and computational thinking, Wing (2006) has articulated the value of broadening the view of programming from the learning of concepts and techniques to the adoption of computational practices and strategies. As is well known in the Computational Thinking Education (CTE) community addressed the educational point of CT and programming to involve not only computational concepts but also practices and strategies (Jansen et al., 2018), i.e. higher-order problem solving competences such as abstraction, decomposition and pattern recognition.

3.2 Mathematical Problem Solving Applied by CT

In the past decade, the situation seems to have swung again and programming has drawn new attention but in a new guise, that of CT as a fundamental 21st century competence for all citizens, involving concepts, practices and dispositions regarding user constructions with digital media (Grover & Pea, 2018). Programming is seen in this context as a central feature of CT involving specific concepts (like e.g. conditionals, loops, variables, recursion). In addition, it involves strategies and practices as well as thinking processes such as problem solving and posing, analysis and decomposition, design, evaluation, refinement and iteration (Wing, 2006). In its current form, programming as an element of CT has been perceived with little connection to mathematical learning. So, what happened to the debate as to how programming can inspire mathematical meaning making? And furthermore, how can this debate connect to a broader debate about connections between mathematics and programming from an epistemological and educational point of view?

4. READRESSING THE PROGRAMMING - MATHEMATICS CONNECTION

As stated earlier in this paper, we reconsider the kinds of connections between mathematics and programming which we feel as worth re-visiting in the wake of attention to CT as a 21st century competence. We do this in an attempt to highlight mathematical and programming concepts in contexts where they have equivalent value and use and to consider the extent to which dispositions, practices and strategies attributed to these two domains may in fact be thought of as mutually compatible and worth integrating. Accordingly, we elaborate on a few examples that address the connectivity between mathematical concepts and thinking processes integrating with the engineering kind of mathematics required to write a computer program. We proceed and describe an overview of a few cases where we focus transdisciplinary challenges concerning mathematics, computational thinking which is later programmatically implemented on CT implementations used for coping with mathematical challenges from across domains. We use them as a starting point for later addressing our current effort and illustrate transdisciplinary approaches of applied CT in service in service of mathematics.

4.1 Overview of our previous efforts

We elicit our current research efforts and ask the question of what kind of mathematics is necessary in order to learn to program (see, for instance, Sinclair and Moon, 1991). During our previous efforts, we addressed different mathematical challenges adapted for different study levels (Jansen et al., 2018; Kynigos, 2015). Accordingly, we explored how these mathematical challenges could be coped in terms of computational thinking as well as how they could be implemented programmatically. To illustrate our efforts, we selected two mathematical cases from across domains. The 1st case corresponds to students attending primary schools coping with simple math challenges. The 2nd case

concerns high-school students coping with higher level of math consisting of geometry challenges. For each of the cases, students were required to analyze the math challenges and seek for algorithmic concepts to solve them. Next, they were presented with an applied tool to code this algorithmic concept. The coding environment was adapted according to students' level of study. Thus, young students used Visual Computer Language (VLC) as a graphical approach offering intuitive and clear view of the proceedings according along the computer program. The high-school students used Python representing a more traditional coding approach offering richer programming options which are optimized to the math challenge they coped with. As illustrated, for both cases, we used the same transdisciplinary approach consisting of postulation of math challenge followed by employment of CT to conceptualized on possible approaches to cope with challenges. Finally, these concepts were formulated as applied programs solving the math challenge. In the next subsection we present the current phase of our research while illustrating this approach in the context of solving geometry challenges while combining core CT concepts using MaLT2.

4.2 Programming to learn Mathematics

In this case we bring our current phase of our efforts to further explore new ways to use and learn mathematical ideas through programming. Consider for example the following four ways to construct a circle with the Logo based programming language in MaLT2.

Intrinsic Circle

;creates a polygon approximation of a circular curve using Instrinsic Geometry only

```
to circlea :a :n
repeat :n [fd :a rt 360/:n]
end
circlea 6 60
```

Intrinsic Circle using a Euclidean property

;creates a polygon approximation of a circular curve using a Euclidean property for radius

```
to circleb :r :n
repeat:n [fd (2*pi*:r)/:n rt 360/:n]
end
circleb 50 36
```

Euclidean Circle

;uses the Euclidean definition of points equi-distant to the centre

```
to circlec :r :n
repeat :n [pu fd :r pd point pu bk:r pd rt 360/:n]
 end
circlec 100 36
```

```
to point
 fd 2 bk 2
end
```

Cartesian Circle

;uses a Cartesian function for each quadrant

```
to circle :r
upright :r :r
upleft :r :r
downright :r :r
downleft :r :r
pu home pu
end
```

```
to upright :r :x
if :x=0 [stop]
pu
setx :x
sety sqrt ((:r*:r) - (:x*:x))
pd
fd 1
upright :r :x-1
end
```

Each of these uses different mathematical properties coming from distinct geometrical systems to construct the same figure. The first one constitutes a polygon approximation of a circle and does not employ Euclidean elements such as center or radius nor Cartesian/Algebraic ones such as circle functions. The second one employs a Euclidean property relating the circumference to the diameter in order to nevertheless construct a polygon - circle in intrinsic Geometry - style. The third uses the equidistance to the circle's center point Euclidean definition. The fourth constructs four quadrants (only one is written here for space economy) using Cartesian positioning primitives and the circle function. A pedagogical approach engaging students with the distinctions between these definitions and constructions would potentially be particularly rich for the respective mathematical meaning-making distinguishing amongst the geometrical systems employed (Kynigos, 1992). In these cases, the students would need to be able to use computational ideas such as structured programs, variables, loops, not to mention recursion. But these concepts would be just tools to focus on and consider the mathematics in a mathematics course.

4.3 Distinguishing between approaches

In this subsection, we focus on how to distinguish between the presented approach while emphasizing that even in the case where we have the same programming tool and the same problem, there can be different approaches to it, here corresponding to the ideas described in previous sections. We give an example of two very different solutions to the problem of constructing a program to create a generalized parallelogram which however can never be a square. The problem was given by a teacher in year 8 of a mathematics class. His students proposed the following program to construct a generalized parallelogram as follows:

```
to parallelogram :a :b :c
repeat 2 [ fd :a rt :c fd :b rt 180-:c]
end
```

This procedure expresses the class of objects 'parallelogram' since it contains variables for the independent linear and angular elements, expresses the property of equality by means of a loop to repeat half the figure twice and the angular dependency by means of a linear function between two consecutive avatar turns. The students were asked to solve the above problem after having constructed and discussed this procedure. They found many solutions mostly from the following kind:

```
to parallelogram :x :c
repeat 2 [forward :x right :c forward :x+20 right 180-:c]
end
```

In this solution, the students imposed an otherwise redundant functional relation between two consecutive linear elements of a parallelogram. The definition of a parallelogram implies that there must be no dependency between the length of two consecutive sides. The students solved the problem of constructing a parallelogram which can never be a square by imposing a functional relationship between those lengths which makes it impossible for a

property of the square to apply, i.e. that the lengths can never be equal since they must have a difference of 20. So here, the big aim, is represented by a generalized property of a geometrical figure combined with the idea of function and generalized number.

To parallelogram: a: b: c
If: c <90 repeat 2 [fd: a rt: c fd: b rt 180-: c]
If: c> 90 repeat 2 [fd: a rt: c fd: b rt 180-: c]
End

This response to a task was set by the authors during a programming course to learn how to program geometrical figures in MaLT2 (Kohen & Milrad, 2019). Here, the program generates a parallelogram in every case except for the value of a turn which allows the generation of a rectangle. It thus avoids the generation of a square by imposing a higher order negation of generating a rectangle. It could be argued that this solution fits better into 'the way in which a computer would resolve the problem' since the problem was worded - create a program to construct a generalized parallelogram which can never become a square. But here, the mathematical concept needed in order to construct the program looks like it's in the service of a computational idea, that of conditionals. It is necessary to know that of the turns cannot be 90 degrees then the figure cannot be a rectangle and therefore it cannot become a square. So, these are two correct solutions but one employs a mathematical idea of imposing a redundant linear relationship between two linear elements of the model and the other employs a computational idea - a conditional to simply exclude the creation of a square by means of excluding only the two values which would yield a rectangle.

5. DISCUSSION - CONCLUSION

In this paper we have presented a few examples with different approaches illustrating how mathematics and programming can be integrated in various ways. Our goal is to help curriculum designers to place joint programming and mathematics activities in either of the respective curricula or consider them in trans-disciplinary educational activities including post-normal science perspectives which focus on larger socio-scientific issues. For a computer science teacher, integrating mathematics at the service of programming concepts could be a way in to including mathematics into the teaching of programming before considering mathematics as the object of programming, i.e. starting from the approach shown in 4.1 to move to the one in 4.3. Conversely for a mathematics teacher a progression from 4.2 to 4.3 could be appropriate. In both cases being explicit about the positioning and the role of the two subjects would help designing activities which make more sense to students. This kind of discussion may help clarify educational policy and curriculum design issues related to implementation aspects in schools. What kinds of domains are rich in opportunities for them to develop CT practices and strategies in the context of using big ideas either in mathematics or in programming? What kinds of specific connections can be pedagogically engineered between such ideas from each domain, for instance between functional relations and generalized number from mathematics and variables and model animation properties from computer science? These are current and future directions in which we are focusing our research efforts on.

6. REFERENCES

Grover, S. & Pea, R. (2018). Computational Thinking: A Competency whose Time has Come. *Computer Science Education: Perspectives on teaching and learning.* London: Bloomsbury Academic, 19-37.

Jansen, M., Kohen-Vacs, D., Milrad, M. (2018). A Complementary View for better Understanding the Term Computational Thinking. *Proceedings of the International Conference on Computational Thinking Education 2018*. Hong Kong: The Education University of Hong Kong, 2-7.

Kohen, D., & Milrad, M. (2019). Computational Thinking Education for In-Service Elementary Swedish Teachers: Their Perceptions and Implications for Competence Development. *Proceedings of the International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong, 109-112.

Kynigos, C. (1992). *The Turtle Metaphor as a Tool for Children Doing Geometry in Learning Logo and Mathematics*. Cambridge, MA: MIT press.

Kynigos, C. (2015). Constructionism: Theory of Learning or Theory of Design? *Selected Regular Lectures from the 12th International Congress on Mathematical Education*. Cham: Springer, 417- 438.

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers.* Dordrecht: Kluwer. *Science and Technology, 3*(3), 249-262.

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

Papert, S. (2000). What's the Big Idea? Toward a Pedagogy of Idea Power. *IBM Systems Journal, 39*(3.4), 720-729.

Sinclair, K., & Moon, D. (1991). The Philosophy of LISP. *Communications of the ACM, 34*(9), 40–47.

Wing J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3).

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

An Empirical Study of Analyzing the Behaviors of the Sixth Grade Students in Learning English Oral Interaction with Educational Robots

[1]Chao-jui HSU, [2]Ting-chia HSU*
[1,2]National Taiwan Normal University, Taiwan
god26275001@gmail.com，ckhsu@ntnu.edu.tw

ABSTRACT
This study attempted to explore the learning behaviors of the six grade students using educational robots on the learning units of oral interaction in English. This study provided the application of smart phone for controlling the action of the robots and ask the students to orally interact with partners so as to put the objective learning sentences into practice. Then, the foreign language interactive behaviors were recorded and observed during the period of collaborative learning tasks. The participants were 18 English as Foreign Language (EFL) learners whose age were from 11 to 12. The research results showed that exercise of expressing opinions in English with objective learning sentence is the most frequent behaviors in the learning process, implying that the game of the educational robots did not preclude the students from naturally using English oral presentation to achieve the purpose of communication, so as to reduce the foreign language learning anxiety.

KEYWORDS
foreign language learning anxiety, educational robot game, English oral interaction

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

探討六年級學生使用教育機器人學習英語口語互動之行為實證分析

[1]許晁睿，[2]許庭嘉*
[1,2]國立臺灣師範大學，科技應用與人力資源發展學系，臺灣
god26275001@gmail.com，ckhsu@ntnu.edu.tw

摘要
本研究旨在探討國小六年級學生使用教育機器人在英語口語互動學習單元之學習行為分析。本研究提供手機應用程式來控制機器人行動，並要求學生和同伴用英語口語互動以實際演練學習的目標句型，然後錄影和觀察學生在合作學習的任務期間，所進行的外語互動行為。參與者由 18 名 11-12 歲以英語作為外語學習的學習者所組成。研究結果顯示，在學習過程中，學生練習使用學習目標的句型練習英語表達意見的部份是很常出現的行為，代表教育機器人遊戲並不會讓學生排斥自然使用英語口說達成溝通目的，減少外語焦慮。

關鍵字
外語學習焦慮；教育機器人遊戲；英語口語互動

1. 前言
資訊技術是世界設施的基礎。在這種社會背景下，教育像任何生產或服務部門一樣，都受到技術的影響。面對這種環境，教育系統必須使年輕人適應數位世界，因此，在學校中，我們應該訓練語言和數位素養，否則他們將成為數位文盲 García-Peñalvo（2018）。運算思維作為一種使用電腦技術解決問題的方法，已成為主流，因為許多政府正在提高兒童的程式設計能力。但是，除了所掌握的程式設計能力外，對於程式設計如何影響其他方面的研究較少（Moreno-León & Robles, 2015）。

在全球化和社會文化的趨勢背景下，移民和英語的傳遞讓英語學習者的社會和教育背景有多樣性。由於英語是一種國際語言，因此，如果不特別注意英語教學的背景，就不能做出有效的教學決策（McKay & Bokhorst-Heng, 2017）。數十年來在東亞國家中，英語教育有普及和重要性的狀況，台灣和其他東亞國家已開始在高等教育領域推廣英語中等教學（English-medium instruction, EMI）。在 EMI 教室中通過英語進行交流是英語作為外語學習者（English as a foreign language, EFL）的基本要求，但這可能是一個挑戰。因為說外語是一個複雜的過程，包含語言能力，口語技巧和策略運用（Chou, 2018）。第二語言焦慮一直是經驗和理論上不斷關注的對象。出於理論和實踐的考慮，該領域的許多研究都檢查了焦慮與第二語言成就之間的關係（Teimouri, Goetze, & Plonsky, 2019）。焦慮對於外語學習格外重要，因為焦慮可能會阻礙學習者與他人交流（Horwitz, Horwitz, & Cope, 1986）。學者們已經表明在語言學習中考慮焦慮的重要性。因此，越來越多的研究人員嘗試整合多種學習策略和技術來減輕學生在學習英語時的焦慮（Hwang, Hsu, Lai, & Hsueh, 2017）。Benitti（2012）指出，機器人教學是一個很好教學工具，並且也非常容易吸引學生的注意力。為了使學習語言過程更具刺激性，老師需要在開發活動時投入大量思想，以保持學生的興趣並實現短期目標，從而增強自信心並降低焦慮水平（Alemi, Meghdari, & Ghazisaedy, 2014）。

基於上述提到越來越多研究者想透過不同的學習策略來降低學生的學習焦慮，並且機器人教學容易吸引學生注意力。因此本研究將探討對國小六年級的學生，以英語做為學習單元，提供了手機應用程式及教育機器人，輔助學生學習運算思維的概念，同時透過課堂活動遊戲與機器人互動，來觀察學生在學習行為中的過程。

2. 文獻探討
2.1. 外語焦慮（Foreign language anxiety）
多數的外語學習者在學習外語過程中感到焦慮（Elaldi, 2016）。英語作為外語教學（EFL）在亞洲面臨許多挑戰，例如缺乏互動式語音環境、強調考試成績以及存在外語焦慮的問題（Yen, Hou, & Chang, 2015）。EFL 學習者不願在課堂上講英語，這是外語背景下常見的問題（Hamouda, 2013）。學生在學習英語時表現出許多問題和困難，如語法，詞彙和發音等，這些通常被認為會妨礙 EFL 學習者的難題（Hashemi & Abbasi, 2013）。學生必須被告知他們不是唯一一個在學習外語時遇到焦慮的人。學生認為自己英語能力低，並且缺乏信心和準備，以及害怕犯錯和得到負面評價，所以相當多的學生不願回應老師（Hamouda, 2013）。大多數學習者在外語學習中都有一定程度的焦慮，例如發音困難，被老師立即糾正問題，不理解老師所提出的問題。焦慮會對學習第二語言或外語的過程產生負面影響，老師與學生普遍認爲焦慮是學習的障礙（Horwitz et al., 1986）。焦慮的主要原因主要都是對互動、自尊心低、缺乏自信、缺乏準備和擔心失敗等因素（Marwan, 2016; Melouah, 2013）。

總和以上的敘述，語言表現不足以及被其他人評價為負面的恐懼最有可能引發年輕學習者的焦慮（Liu & Chen, 2014）。老師和課程設計者應該提出一些教學活動，以幫助學習者減少焦慮（Al-Khasawneh, 2016）。為了幫助減少學習者的焦慮，英語老師應該需要知道 EFL 學習者中存在著外語焦慮，並在課堂上表現出同情心。老師需要應對學生的外語焦慮，並防止學生迴避課堂參與（Park & French, 2013）。學校環境和情境項目，也可能會影響學生的焦慮水平，因此老師應該提供安全和有吸引力的環境（Henter, 2014）。

2.2. 運算思維與教育機器人

教育機器人技術可以做為一種工具，提供學生參與和發展運算思維的機會（I. Lee et al., 2011; Repenning, Webb, & Ioannidou, 2010）。許多學校也開始在引入一些教育機器人讓學生體驗不一樣的學習環境，提升和建立更高階的運算思維的能力，並幫助學生解決複雜的問題（Blanchard, Freiman, & Lirrete-Pitre, 2010）。

關於運算思維的教學，在過去的研究中，已經有學者接受將教育機器人作為教導學生運算思維的方法（Bers, Flannery, Kazakoff, & Sullivan, 2014; Botički, Pivalica, & Seow, 2018）。教育機器人是一種功能強大和具有高彈性的教學工具，機器人的技術通常都包含了科學、數學、資訊和科技的學科，被視為一門跨學科的活動，並為各個年級的學生帶來了重大的好處。有研究人員表示，針對 4-6 歲的兒童可以建立簡單的機器人專案項目，進而熟悉工程技術和程式語言的思考模式，同時還能建立運算思維的能力（Bers et al., 2014）。Penmetcha （2012）研究了教育機器人對大學生探索機器人技術與開發程式語言和演算法思維之間的相關影響，結果表示，無論學生的背景如何，機器人活動都能作為媒介落實整合運算思維的目的，並可以教更高層面的抽象化和程式設計概念。機器人教學活動具有改善課堂教學的巨大潛力，能讓學生從被動學習的身分，轉換為主動學習者，進而形成主動與同儕互動並建立良好關係。許多研究指出，教育機器人的課程對學生批判性思考、問題解決能力以及認知能力有正面影響（Atmatzidou & Demetriadis, 2012; Blanchard et al., 2010）。其他研究也有指出教育機器人如何提升學生學習的方式以及學生的動機、合作和創造力（Eguchi, 2010; Khanlari, 2013）。

3. 研究方法

3.1. 研究對象

本研究針對學生在英語學習中的行為進行編碼，實驗對象台灣北部某國小六年級的學生，平均年齡為 11-12 歲，有 18 位學生，過程中使用的教材語言為英語，學生需要使用手機來操控教育機器人，透過手機的應用程式拖拉積木程式，將積木程式層層堆疊，藉此來操控教育機器人，學生可以透過機器人的反應及行為，來得知自己所拉的積木程式與自身預期教育機器人的反應是否一樣，來觀看學生在英語學習過程中的行為。

3.2. 序列分析之編碼系統

為了探討學習者在學習過程中的學習行為，本研究針對學生的學習行為記錄為 log 檔並進行編碼，並參考相關研究編碼來進行編碼，使用 GSEQ 軟體進行分析分析學生在學習行為之間進行的活動結果，以探討學生的行為模式。在學習過程中將學習狀況分為三類：運算思維行為、語言行為和其他行為，如表 1 所示。

表 1 行為分析編碼表

類別	代碼	意義	範例
運算思維編碼	PP(People&People)	組內對談	同組的兩個人在對談
	PC(People Commnication)	組外對談	與別組在對談
	PR(People & Robot)	使用機器人、手機	操控手機拖拉積木程式使機器人移動
	AT(Algorithm)	使用卡牌	排除卡牌(前進、左轉等)
	PD: 1.ID(Individual Decision) 2.CD(Cooperation Decision)	組內當中: 1.個人(ID)使用任務卡 2.共同(CD)使用任務卡	使用任務卡放置石頭、砂土…等
	PM(Physical Message)	姿體表達	行為左右轉、手勢左右轉…等
	AG(Abstraction General)	資料簡化或用其他方式表達	單程式方法便迴圈方式表達
語言編碼	LI(Learning Interaction)	被觀察者正在練習口語互動	自己口語互動
	PLI(People learning)	正有其他人在指導被觀察者口語互動	教師教觀察者學習口語互動
	NS(No Speaking)	不會口語互動	沒有講任何語言
	YS(Yes Speaking)	會正確口語互動	單字、句型接正確
	LT(Listen to teacher)	聽教師講解	教師講解遊戲規則
其他	IM(Irrelevant Message)	無關課程	發呆、離開座位…等
	SP(Separate)	組內做不同的事	各做各的事

4. 研究結果

本實驗將學習者利用教育機器人進行英語的學習過程，進行行為分析的比較。依據影片紀錄，學習者的行為經過編碼及後續的序列分析後，共獲得 1542 個行為編碼，本研究進一步對 18 名學習者提出的各種編碼進行行為頻率計算，如表 2 所示。

表 2 學習者之學習行為出現比率

編碼	次數	百分比（%）
PP	118	7.65%
PC	175	11.34%
PR	594	38.52%
ID	13	0.84%
CD	98	6.35%
AT	99	6.42%
PM	103	6.67%
AG	0	0.00%
LI	110	7.13%
PLI	17	1.10%
NS	0	0.00%
YS	0	0.00%
LT	80	5.18%
IM	38	2.46%
SP	97	6.29%

根據上表，可以看出學習者中，出現前五名行為依序為使用機器人（PR）佔了 38.52%、與組外對談（PC）佔了 11.34%、與組內對談（PP）佔了 7.65%、說英語互動（LI）佔了 7.13%、姿體表達（PM）佔了 6.67%，由此可知，學生在使用教育機器人進行英語學習時，使用機器人（PR）、與組外對談（PC）、組內對談（PP）、說英語互動（LI）、姿體表達（PM）等行為是最常出現的動作。

4.1. 行為分析

進行上述行為的次數分析以後，將針對分析結果比較每個行為碼之間的關係圖，並畫製成行為編碼圖。箭頭方向為起始行為編碼至目標行為編碼，上方的數值表示從起始行為至目標行為關係的 Z 值，Z 值大於 1.96 代表著行為序列達到顯著水準（$p<0.05$）（Bakeman & Gottman, 1997），如圖 1 所示。

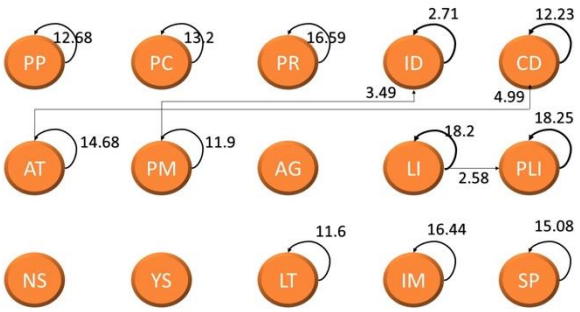


圖 1 小學六年級學生之行為編碼圖

從圖 1 中可以發現，學生的行為模式可以分為三個行為的連動性。這三個連動分別為 AT-CD（即使用手牌、和合作使用任務卡）；PM-ID（即姿體表達和與個人使用任務卡）；LI-PLI（即學生正在練習口語互動（自己說英語） 和有其他人在指導被觀察者如何說（英語））。每當學生在使用手牌時（AT），會與同組的學生一起使用任務卡（CD）;並且學生會透過模仿機器人的行為，進行姿體表達（PM），接著會個人使用任務卡（ID）來取得任務相關的道具;另外，當學習者練習口語表達後，會有其他人來指導學習者的口語互動的部分。

5. 結論與未來展望

近年來，因為對不同學習者的行為分析需求已經大幅提升，故本研究利用教育機器人輔助學習者學習，並透過行為分析編碼表來觀察學習者的行為，先前的研究表示，機器人可以成為動機的重要來源，在以人為基礎的學習方法上也能具有很大的優勢，可以減少焦慮程度並為語言學習者提供更加輕鬆的氛圍（S. Lee et al., 2011）。一些針對英語教師的教學建議：情境因素，即學校環境和情境項目，可能會影響學生的焦慮水平，因此教師應提供安全和有吸引力的環境（Henter, 2014）。多媒體環境可以減少學生的焦慮，並提供較少壓力的課堂環境。除此之外，多媒體工具使英語教師能夠幫助學生提高英語表現並降低他們的語言焦慮（Huang & Hwang, 2013）。當學生開始在外語課堂上感到安全時，他們自然會開始說話。總之，所有外語教師都需要激勵學生;鼓勵他們說話;並允許他們犯錯而不受懲罰（Atas, 2015）。

本研究透過教育機器人輔助工具，來探討學生在使用手機應用程式以及說英語的過程，並對學生的行為進行分析，實驗結果顯示，在學習過程中，透過教育機器人的輔助，學生在英語口說的部份是很常出現的動作，代表學生在英語口說的部份並不排斥，為了完成課堂上提供的任務卡，是可以在上課中與同學進行英語口說互動。接著分析了學生之間的行為關聯，發現學習者正在練習英語口說的時候，會有其他人指導被觀察者如何說，代表學生在練習英語口說時，如果學習者有錯誤的話會有指導老師指導學習者進行英語互動，這可以使學習者更知道英語口說哪裡需要改進與修正。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 108-2511-H-003 -056 -MY3 的部分補助。

7. 參考文獻

Al-Khasawneh, F. M. (2016). Investigating Foreign Language Learning Anxiety: A Case of Saudi Undergraduate EFL Learners. *Dil ve Dilbilimi Çalışmaları Dergisi, 12*(1), 137-148.

Alemi, M., Meghdari, A., & Ghazisaedy, M. (2014). The Effect of Employing Humanoid Robots for Teaching English on Students' Anxiety and Attitude. *Proceedings of 2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. IEEE, 754-759.

Atas, M. (2015). The Reduction of Speaking Anxiety in EFL Learners Through Drama Techniques. *Procedia-Social and Behavioral Sciences, 176*, 961-969.

Atmatzidou, S., & Demetriadis, S. N. (2012). Evaluating the Role of Collaboration Scripts as Group Guiding Tools in Activities of Educational Robotics: Conclusions from Three Case Studies. *Proceedings of 2012 IEEE 12th International Conference on Advanced Learning Technologies*. IEEE, 298-302.

Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis*. Cambridge University Press.

Benitti, F. B. V. (2012). Exploring the Educational Potential of Robotics in Schools: A Systematic Review. *Computers & Education, 58(*3), 978-988.

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.

Blanchard, S., Freiman, V., & Lirrete-Pitre, N. (2010). Strategies Used by Elementary Schoolchildren Solving Robotics-based Complex Tasks: Innovative Potential of technology. *Procedia-Social and Behavioral Sciences, 2*(2), 2851-2857.

Botički, I., Pivalica, D., & Seow, P. (2018). The Use of Computational Thinking Concepts in Early Primary School. *Proceedings of International Conference on Computational Thinking Education 2018.* EdUHK, 8-13.

Chou, M. H. (2018). Speaking Anxiety and Strategy Use for Learning English as a Foreign Language in Full and Partial English-Medium Instruction Contexts. *TESOL Quarterly, 52*(3), 611-633.

Eguchi, A. (2010). What is Educational Robotics? Theories Behind It and Practical Implementation. *Proceedings of Society for Information Technology & Teacher Education International Conference.* AACE, 4006-4014.

Elaldi, S. (2016). Foreign Language Anxiety of Students Studying English Language and Literature: A Sample from Turkey. *Educational Research and Reviews, 11*(6), 219-228.

García-Peñalvo, F. J. (2018). Editorial Computational thinking. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 13*(1), 17-19.

Hamouda, A. (2013). An Exploration of Causes of Saudi Students' Reluctance to Participate in the English Language Classroom. *International Journal of English Language Education, 1*(1), 17-34.

Hashemi, M., & Abbasi, M. (2013). The Role of the Teacher in Alleviating Anxiety in Language Classes. *International Research Journal of Applied and Basic Sciences, 4*(3), 640-646.

Henter, R. (2014). Affective Factors Involved in Learning a Foreign Language. *Procedia-Social and Behavioral Sciences, 127*, 373-378.

Horwitz, E. K., Horwitz, M. B., & Cope, J. (1986). Foreign Language Classroom Anxiety. *The Modern Language Journal, 70*(2), 125-132.

Huang, P., & Hwang, Y. (2013). An exploration of EFL Learners' Anxiety and E-learning Environments. *Journal of Language Teaching and Research, 4*(1), 27.

Hwang, G. J., Hsu, T. C., Lai, C. L., & Hsueh, C. J. (2017). Interaction of Problem-based Gaming and Learning Anxiety in Language Students' English Listening Performance and Progressive Behavioral Patterns. *Computers & Education, 106*, 26-42.

Khanlari, A. (2013). Effects of Robotics on 21st Century Skills. *European Scientific Journal, 9*(27).

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, Malyn-smith J., & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads, 2*(1), 32-37.

Lee, S., Noh, H., Lee, J., Lee, K., Lee, G. G., Sagong, S., & Kim, M. (2011). On the Effectiveness of Robot-assisted Language Learning. *ReCALL, 23*(1), 25-58.

Liu, H. J., & Chen, T. H. (2014). Learner Differences Among Children Learning a Foreign Language: Language Anxiety, Strategy Use, and Multiple Intelligences. *English Language Teaching, 7*(6), 1-13.

Marwan, A. (2016). Investigating Students' Foreign Language Anxiety. *Malaysian Journal of ELT Research, 3*(1), 19.

McKay, S. L., & Bokhorst-Heng, W. D. (2017). *International English in its sociolinguistic contexts: Towards a socially sensitive EIL pedagogy*. Routledge.

Melouah, A. (2013). Foreign Language Anxiety in EFL Speaking Classrooms: A Case Study of First-year LMD Students of English at Saad Dahlab University of Blida, Algeria. *Arab World English Journal, 4*(1).

Moreno-León, J., & Robles, G. (2015). Computer Programming as an Educational Tool in the English Classroom a Preliminary Study. *Proceedings of 2015 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 961-966.

Park, G. P., & French, B. F. (2013). Gender Differences in the Foreign Language Classroom Anxiety Scale. *System, 41*(2), 462-471.

Penmetcha, M. R. (2012). *Exploring the Effectiveness of Robotics as a Vehicle for Computational Thinking*. Doctoral Dissertation, Purdue University.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable Game Design and the Development of a Checklist for Getting Computational Thinking Into Public Schools. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education.* ACM, 265-269.

Teimouri, Y., Goetze, J., & Plonsky, L. (2019). Second Language Anxiety and Achievement: A Meta-analysis. *Studies in Second Language Acquisition,* 1-25.

Yen, Y.-C., Hou, H.-T., & Chang, K. E. (2015). Applying Role-playing Strategy to Enhance Learners' Writing and Speaking Skills in EFL Courses Using Facebook and Skype as Learning Tools: A Case Study in Taiwan. *Computer Assisted Language Learning, 28*(5), 383-406.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Teacher Development

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Workshops and Co-design Can Help Teachers Integrate Computational Thinking into Their K-12 STEM Classes

Sally P. W. WU[1*], Amanda PEEL[2], Connor BAIN[3], Gabriella ANTON[4], Michael HORN[5], Uri WILENSKY[6]
[1,2,3,4,5,6]Northwestern University, United States
sally.wu@northwestern.edu, amanda.peel@northwestern.edu, connorbain2015@u.northwestern.edu, gabriellaanton3.2020@u.northwestern.edu, michael-horn@northwestern.edu, uri@northwestern.edu

ABSTRACT
This work aims to help high school STEM teachers integrate computational thinking (CT) into their classrooms by engaging teachers as curriculum co-designers. K-12 teachers who are not trained in computer science may not see the value of CT in STEM classrooms and how to engage their students in computational practices that reflect the practices of STEM professionals. To this end, we developed a 4-week professional development workshop for eight science and mathematics high school teachers to co-design computationally enhanced curriculum with our team of researchers. The workshop first provided an introduction to computational practices and tools for STEM education. Then, teachers engaged in co-design to enhance their science and mathematics curricula with computational practices in STEM. Data from surveys and interviews showed that teachers learned about computational thinking, computational tools, coding, and the value of collaboration after the professional development. Further, they were able to integrate multiple computational tools that engage their students in CT-STEM practices. These findings suggest that teachers can learn to use computational practices and tools through workshops, and that teachers collaborating with researchers in co-design to develop computational enhanced STEM curriculum may be a powerful way to engage students and teachers with CT in K-12 classrooms.

KEYWORDS
computational thinking, STEM education, K-12, teacher professional development, curriculum design

1. INTRODUCTION
Initiative to incorporate *computational thinking* (CT) in K-12 education face challenges on several fronts, particularly in the United States. CT education often takes place within computer science courses, which may limit access to those who traditionally take computing courses (Heinz, Mannila, & Färnqvist, 2016). Moreover, there is a dearth of K-12 teachers trained in computer science and technologies (Advocacy Coalition, 2018; Cuny, 2012).

In order to address the systemic barriers to CT education, researchers argue for the integration of CT in K-12 STEM classes (Wilensky, Brady, & Horn, 2014). Integrating CT in STEM classes can broaden access to computational practices for all students, as STEM classes are required in middle and high school. Further, students' use of computational tools has been shown to deepen learning in mathematics and science domains (e.g., Brady et al., 2016; Wilensky, 2003). Weintrop and colleagues (2016) organize computational thinking practices in mathematics and science classrooms into four strands: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices. In this paper, we focus on *modeling and simulation* (using, modifying, and creating computational models) and *data practices* (collecting, visualizing, and analyzing data). Engaging in these CT-STEM practices can help students develop science and mathematics content understanding through authentic STEM practices used in modern science (Weintrop et al., 2016).

Integrating CT in STEM classes further addresses the shortage of teachers trained in computer science by shifting the focus to training STEM teachers in the computational tools and practices relevant to their associated fields. This shift requires both curriculum designers and teachers to reimagine classroom practices and to learn how to incorporate computational methods and tools (Ball & Forzani, 2009; Windschitl et al., 2012). We address this shift using a Design Based Implementation Research (DBIR) framework (Penuel et al., 2011) that supports teachers in professional development and integration of computationally enriched STEM units. Over multiple years of partnering with teachers and schools, our team has shifted from providing day-long professional development to ongoing teacher-driven support. Through these design iterations, we have sought to support teacher ownership, agency, and comfort in teaching with computational tools.

In the latest design iteration, we position teachers as active co-designers in modifying their existing STEM curricula to include computational tools and practices. Our approach foregrounds teachers' views on how the curriculum aligns with teaching strategies and expectations for student learning (Allen & Penuel, 2015; Coburn, 2005; Penuel et al., 2009). Researchers serve as computational experts and work alongside teachers to develop new computationally enriched STEM curricula that align with individual teacher's views and goals. The co-design process aims to (1) help teachers develop an understanding of CT and (2) empower teachers to integrate and teach CT in their STEM courses. In this paper, we present the results of a month-long professional development in which high school teachers co-design CT-STEM curricula with researchers. We investigate the research questions: (1) What did teachers learn about CT through a 4-week professional development? and (2) How did teachers integrate CT into their curriculum?

2. METHOD
To investigate our research questions, we developed the CT-STEM Summer Institute (CTSI), a 4-week professional development workshop that positioned teachers and researchers as co-designers of curriculum. Teachers and

researchers formed design teams by subject area: three biology teachers (pseudonyms: Betty, Briana, Brooke); one chemistry teacher (Carrie); three physics teachers (Penny, Peter, Philip); and one mathematics teacher (Matt). The eight participants teach high school science or mathematics in four U.S. public schools (2 urban and 2 suburban). Teachers received $1000 U.S. dollars per week of participation in CTSI and were asked to create a CT-STEM curriculum for their classroom that would be implemented in the following school year. Seven graduate students and one post-doctoral researcher were assigned to work with teachers based on their prior experience working with specific subject areas and participating teachers.

Table 1. Overview of Professional Development Activities over Four Weeks of CTSI, Organized by Day.

Week	Monday	Tuesday	Wednesday	Thursday	Friday
1	Pre-survey Introductions Demo CT Lesson	Computational Models and CT-STEM Practices	Computational Tools	Computational Tools Unit planning Reflection	Work from home
2 + 3	Work from home Review partner's work	Discuss feedback Co-design (2-3 hours)	Co-design (2-3 hours) CT-STEM Workshop	Co-design (3.5 hours) Reflection	Work from home
4	Work from home Review partner's work	Discuss feedback Co-design (2-3 hours)	Co-design (3 hours) CT-STEM Workshop	Co-design (3.5 hours) Reflection	Post-survey Post-interview Co-design (1 hour) Curriculum Showcase

Table 1 shows an overview of activities during the 4-week professional development. Teachers and researchers met in-person for 14 days from 10am-3pm, with one hour for a catered lunch.

The first week of CTSI (4 days) comprised of workshops led by the researchers. Each workshop introduced computational practices and tools by engaging teachers in lessons designed for students. Each lesson demonstrated how computational tools can engage students in CT-STEM practices while learning disciplinary content. For example, one lesson (https://tinyurl.com/IntroToCT) first asked teachers to use, modify, and debug a series of computational models that simulate how fire spreads through a forest (http://tinyurl.com/netlogofire; Wilensky, 1997) using *NetLogo*, a multi-agent programmable modeling environment (Wilensky, 1999). Next, teachers collected and analyzed 'density vs. percent burned' data using *CODAP* (https://codap.concord.org/; Common Online Data Analysis Platform), a web-based data analysis environment. Then, they posed research questions about other variables that may affect the spread of fire and discussed how scientists use such computational models. Finally, teachers reflected on the pedagogy of CT-STEM practices and how they may use computational models and/or data analysis tools with students.

In addition to NetLogo and CODAP, teachers engaged in *Unplugged* CT activities, which teach CT without computing tools (e.g., writing loops on paper), and *NetTango*, a blocks-based programming interface for exploring NetLogo Web models (Horn et al., 2014), in the context of a chemistry unit on molecular particle collisions.

The last three weeks of CTSI provided co-design time for teams of teachers and researchers to sit together as they worked on computational models and units. Teams engaged in approximately 24 hours of in-person co-design time. On Fridays and Mondays, teams worked from home and communicated via email as needed. Each team reviewed each other's work on Monday afternoons and discussed the feedback on Tuesdays. In addition, teams engaged in supplemental CT-STEM workshops that focused on CT tools or pedagogy on Wednesdays and participated in a reflection session on Thursdays. Each co-design team differed in how they collaboratively built models and curricula materials (Kelter et al., 2020).

At the end of CTSI, the teachers and researchers showcased their co-designed CT-STEM curriculum in an event open to the community: https://tinyurl.com/CTSI2019Expo. All teachers also responded to pre/post surveys and post-interviews, as described below.

2.1. Data Sources

To assess what teachers learned from CTSI (**RQ1**), the 33-item *pre/post surveys* asked teachers to rate on a 5-point Likert Scale (*1* = Strongly Disagree, *5* = Strongly Agree): their perception of CT (Adapted from Cabrera et al., 2018) and comfort with CT-STEM practices. Further, in the *post-interview*, we asked teachers what they learned from CTSI.

To assess how teachers integrated CT into their curriculum (**RQ2**), we asked teachers to describe their curriculum in the *post-interview* and examined the computational tools and practices used in their *CT-STEM curriculum*.

3. RESULTS

3.1. What Teachers Learned about CT

To address **RQ1** (what teachers learned about CT through professional development), we first analyzed teachers' ratings on the *pre-/post-survey*. Due to the small sample size, we qualitatively compare differences from pre to post. Note that Brooke did not complete the pre-survey (4.8 average across all categories on post-survey) and Philip did not complete the post-survey (4.4 average on pre-survey).

Table 2. Average Pre/Post Survey Response by Category.

	CT Value	CT in STEM	CT Integration	Modeling Practices	Data Practices	**Overall**
Pre	4.1	4.1	4.1	3.8	3.0	**3.7**
Post	4.3	4.6	4.4	4.2	4.0	**4.2**

As shown in Table 2, teachers were more likely to agree or strongly agree on all item categories on the post-survey, compared to the pre-survey. That is, after the professional development, teachers reported that they understood the role of CT in STEM education and valued CT to a greater degree. Teachers also reported higher confidence in their ability to identify and integrate computational modeling and data practices into their teaching.

Next, we analyzed the *post-interview* responses to: "What have you learned from CTSI?" We qualitatively reviewed responses of all eight teachers to identify themes mentioned by multiple teachers. Below, we present teachers' responses with the four themes underlined: computational thinking, computational tools, coding, and collaboration.

3.1.1. Computational thinking

Two teachers described learning about CT: Briana (see Section 3.1.3) and Peter. Peter described different levels of CT practices in how they affect students' thinking:

I think being able to see the different domains of computational thinking and the different levels was important. That at one level, it's just: Can you use a model? Can you change a model? Right? Can you collect data? Can you represent data? That's one level, but then can you dig in deeper? Can you change a model? Can you design a model? Can you manipulate data and represent it in different ways? Those are deeper levels that the goal is to try to push down as far as you can to get the kids' thinking, at a really deep level. So that's one thing that I've learned about computational thinking itself.

Peter learned that CT can engage students in more procedural thinking, such as using models and collecting data, as well as more deep conceptual thinking, such as changing and designing their own models. His goal now is to focus on "push[ing]" students' thinking "at a really deep level" because "the different levels [are] important."

3.1.2. Computational tools

Four teachers stated what they learned about *specific computational tools* (Peter, Matt, Philip, and Carrie). Peter and Matt listed different computational tools that they learned about and plan to use in their classroom. Additionally, Matt discussed how the computational tools can help students engage in math as professionals do:

I'd never heard of CODAP or NetLogo or NetTango or any of those. So for me, it just gave me some tools that I can use in stats and hopefully geometry to present math in a relevant way to today's learners. I think it will help me answer the question: Why are we learning this? When am I ever going to have to use this? 'Cause it'll be easy to show them, this is what actual researchers are using. 'Here's what actual statisticians are using, rather than we're using the calculator because that's what the AP exam requires you to use.'

Philip and Carrie, who had prior experience building models or implementing CT-STEM lessons, both stated that they became aware of new tools. Carrie added that she was "very excited that [she's] integrating some CODAP this year...[She] already see[s] other possible places in [her] year that [she] can use [CODAP]." Even though the workshops only aimed to help teachers integrate tools into their CT-STEM curriculum, teachers identified CT tools as resources they can use for other lessons in their classroom.

3.1.3. Coding/programming

In contrast to the four teachers above who seemed "excited" and comfortable integrating computational tools into their classrooms, three of the female teachers mentioned learning about *coding* in general because they had little or no prior experience (Betty, Penny, Briana). For example, Betty said she cannot "code anything" but learned how code works and how to explain it to her students:

I knew nothing about coding […] I cannot code anything, maybe a tiny little change I can make, but I at least see now what goes into it and I think I'll be better at explaining things to the kids.

Although Betty feels she can only make "a tiny little change" in code, another teacher Penny discussed learning "a lot" about coding by building NetLogo models for her curriculum and participating in the introductory workshops:

I never knew anything about NetLogo before and I've now learned a lot about NetLogo and modified or helped build some simulations. And that's largely my first and only exposure to coding. So that's relatively new...I thought a couple of the coolest things that we did were within the first week workshops you have for us: the forest fires simulation....that was the first thing where we really looked at the code behind it- and why aren't the trees burning? And I thought that was fun. As well as just seeing the emergent phenomena in that throwing in the same density doesn't always result in the same forest burn rates. So that was cool for me.

While Penny learned that coding was "fun" and "cool" in the first week, Briana stated that she learned to love coding in the second week as she started writing her curriculum and now wants to learn more about how to build models herself. She also mentions learning about all four themes stated across teachers (computational thinking, computational platforms/tools, coding, and collaboration):

I learned more about what computational models are, what computational thinking is. I learned how to incorporate that into my classroom and my lessons more easily. Collaboration is so important. I learned a little bit of how to do some coding and learned different modalities that can be used for different platforms that can be used for different types of analysis....the second week, my Aha moment was I think that creating models is way cooler than writing curriculum...I thought I hated the coding process. At first, I was like it's gonna be terrible, but when I actually learn the foundation/fundamentals, I was like: well this is actually really cool: how a line I write can completely change how something else works. So that was an Aha moment for me is that I would love to learn more about how to do that.

3.1.4. Collaboration

Lastly, four teachers mentioned the value of *collaboration* in their curriculum design process (Briana, Betty, Brooke, Carrie). Betty learned that "a whole team of people" contribute to constructing computational models:

I learned that the value of co-design is very important. Yeah, I'm just more comfortable with using NetLogo...I think just understanding that things have to be coded, like preferences have to be put in there. Someone put that in 'cause I'm like: how do these models know to do this? So you have to actually do some of the research ahead of time, then put it in. And you need a whole team of people. It's not- a computer programmer doesn't know the science necessarily, so you need a scientist with a computer programmer to work together. I love that. I love that idea.

Betty learned that "co-design is very important" because models involve collaborative design decisions from experts from different fields. Similarly, Brooke noted that she benefited from collaborating and brainstorming with the researcher in her team who had a different expertise:

It's just been really nice to have the time to sit down and have conversations around some of this stuff. That's giving me time to dig into the content, research more about what actually- I want it to be about think a little bit more deeply about like the alignment

of the unit itself. And that's just been really great to have [researcher] there to say: Okay, this is the idea. What might fit well? And he'll be like: 'Oh, you could do this or you could do that.' Or just that piece of brainstorming around expertise that I don't have.

In addition to brainstorming, Carrie also mentioned that "[h]aving a researcher with us the whole time was so beneficial" because she could get help on her questions right away from a collaborator sitting right next to her.

3.1.5. Summary of what teachers learned from CTSI

In sum, teachers generally learned more about CT after CTSI. Some of them learned about computational tools and practices that they can integrate into their classroom. Other teachers with limited CT experience learned coding so that they can engage in and explain CT to their students. Further, multiple teachers mentioned collaboration, which supported them in building and integrating computational tools and practices into a CT-STEM curriculum.

3.2. How Teachers Integrated CT

To assess how teachers integrated CT into their curriculum (**RQ2**), we analyzed how teachers used computational practices and tools in their CT-STEM curricula. We first describe their curriculum below and then discuss their use of CT-STEM practices (summarized in Table 3):

1. *Experimental Design and Computational Thinking*: 8-day AP Biology unit that uses a physical lab, CODAP, NetTango, and NetLogo to conduct experiments on animal behavior, further described below (Betty)
2. *Evolution Part II: Natural Selection (Darwin's Finches and The Case of the Rock Pocket Mouse)*: 20-day Freshmen Biology unit that uses CODAP and NetLogo models to collect and analyze data on the mechanisms of natural selection (Briana)
3. *Climate Change in the Great Lakes*: 10-day Environmental Science unit that uses Unplugged activities, CODAP, and NetLogo models to investigate various environmental factors and make sense of climate change models (Brooke)
4. *Energy in Chemical Reactions*: 13-day Chemistry unit that uses NetLogo and CODAP to explore changes in energy when bonds break and form during chemical reactions (Carrie)
5. *Charge Interactions*: 8-day Physics unit that uses a physical lab, CODAP, NetLogo, and PhET simulations to explore the behavior of charges in electricity and magnetism, further described below (Penny and Peter)
6. *1-D Kinematics Motion Maps*: 3-day Physics unit that uses NetLogo and NetTango to analyze and draw velocity in kinematics motion maps, building on Philip's 1-D Kinematics NetLogo model, further described below (Penny and Peter)
7. *1-D Kinematics and Newton's Laws*: six Physics lessons that use CODAP, NetTango, and NetLogo to collect and analyze data through writing formulas and generating graphs on kinematics and Newton's Laws, implemented throughout the fall semester (Philip)
8. *Descriptive Statistics*: 8-day AP Statistics unit using Python notebooks and Unplugged activities to generate formulas, data tables, and plots that describe various real-world datasets (Matt)

Table 3. CT-STEM Practices Targeted in Curriculum

	Curricular Unit (see Section 3.2)							
	1	2	3	4	5	6	7	8
Modeling and simulation practices								
Using computational models (CMs) to understand a concept	x	x	x	x	x	x	x	x
Using CMs to find and test solutions	x	x	x			x	x	
Designing CMs	x				x			
Assessing CMs	x	x	x	x	x		x	
Constructing CMs	x	x			x			
Data practices								
Collecting data	x	x	x	x	x	x	x	
Manipulating data	x	x	x	x	x		x	x
Analyzing data	x	x	x	x	x		x	x
Visualizing data	x	x		x	x	x	x	x
Creating data	x	x	x	x	x	x		

The descriptions of CT-STEM curriculum show that all teachers integrated several computational tools into their curricula to teach disciplinary content. In addition, Table 3 shows that all CT-STEM curricula targeted multiple CT-STEM modeling and data practices. To better understand how teachers integrated computational practices and tools, we present three example curricula (#1, #5, #6) below.

Biology. Betty, with her co-design partner, developed *Experimental Design and Computational Thinking* (#1) for her AP Biology course. She described it as: "really about scientific design and inquiry." In the unit, students design experiments to find the preferred habitat conditions of the *pill bug (rolypoly)*. Betty decided that students start with a physical lab experiment using two connected chambers, one damp and one dry. The students place 10 pill bugs and observe change in population of the two chambers over time. After the physical experiment, students then explore, modify and recreate the animal behavior experiment digitally using NetLogo and NetTango models.

Betty also explained that her unit engages students in multiple CT-STEM data practices: "the kids learn how to set up a controlled experiment, how to collect data, how to make graphs, and it's also where we start to teach them how to analyze some of that data." She integrated these data practices with the CT-STEM practice of using models:

[My class uses] the computational model to learn about the importance of sample size because we only get to use 10 rolypolies and then when we do Chi Square, we don't always get good answers. And then we looked it up, they're like: oh, you need at least 30, for your sample size...So with the model, they can say: oh, what happens if we have 20 rolypolies, 40 rolypolies?

Betty wanted students to not only use models but modify them based on a physical lab: "[students] are now also learning how to change the model. So the first model just has wet and dry, and then in the second activity, they actually changed the code and add their variable, like the one that they tested in class." Specifically, Betty wanted students to learn "that the model is actually coded by a human, based on things that actually happened in real life," as she herself learned at CTSI (see Section 3.1.4). Her integration of NetTango block-based programming makes this design decision particularly salient: "[students] build their chamber using NetTango. Then they put the rolypolies in and all the

rolypolies escape because they didn't tell them to stay within the chamber."

Although Betty expressed that she "cannot code anything" (see Section 3.1.3), her CT-STEM curriculum is the only unit that integrates all modeling and data practices into science content (see Table 3) and forefronts CT in its title.

Importantly, after Betty taught this unit in the fall, classroom observations and an interview suggests that this unit helped students learn science content and engage in CT-STEM practices because Betty discussed coding and CT in context of disciplinary science content, as a result of the professional development (Peel et al., 2020).

Physics. Peter and Penny, who work at the same school, developed two units together for their general Physics classes. With their co-design partners, they designed *Charge Interactions* (#5), which focused on "electrostatics: electric charge, Coulomb's law, electric fields" (Peter), and a short unit on *1-D Kinematics Motion Maps* (#6).

The electrostatics unit first asks students to engage in physical lab experiment with sticky tape and then explore a NetLogo library model on electrostatics (Sengupta & Wilensky, 2005), which was modified with researchers to fit the curriculum. Penny described the unit as primarily focused on the model and how the code works:

Most of it is around the simulation and specific questions asking them to observe particular behaviors or how things happen using their prior knowledge to try to explain why those are things that are happening. And then a few questions asking them to look at the code and, fine, where did we program in that the electron should repel from each other? Like where did we program in that the conductor's color is gray. Could you change that?

Then, students use CODAP to understand Coulomb's Law, as Peter explained: "If we really want them to come up with Coulomb's law, which is our goal, then you have to keep one thing constant and vary another. And CODAP lets you do that really quickly. So that's why we chose that." Finally, students examine a PhET simulation of charges.

Penny and Peter finished their first unit in Week 3, and then modified Philip's 1-D Kinematics NetLogo model for the motion maps unit (#6). Peter saw this short unit as a way to help students dynamically see changes in velocity: "[students] don't often see the map being drawn, as something moves. I think that the simulation that we put together does that and sort of bridge that gap between what we want them to see and what they actually see." The unit also asks students to build their own motion map using NetTango, as Peter explained: "The NetTango thing is a way to help kids gain more control over making a motion map…they have that ownership of the whole process and I think they'll be able to internalize what's going on better."

As of this writing, Penny and Peter have not yet implemented their Charge Interactions unit, but classroom observations of students engaging with the 1-D Kinematics Motion Maps unit showed that both teachers encouraged students to not only understand the science content, but to "explore the code" and "try to break the model."

4. DISCUSSION

Results from our qualitative study suggests that engaging high school STEM teachers in workshops and co-design of CT-STEM curricula in a 4-week professional development can help them develop an understanding of CT and integrate CT into their classroom. We are particularly encouraged by the fact that although these eight teachers already valued CT at the beginning of the workshop because they chose to participate in the professional development, all teachers reported even more favorable perceptions of CT and greater confidence in integrating it into their classroom at the end of the professional development. Teachers shared in post-interviews that they learned not only about CT and computational tools for their classroom, but also about coding in general and the value of collaboration in the co-design process. Due to the relatively recent emergence of CT in STEM for K-12 teachers, particularly in the United States, this work takes one step towards understanding where teachers may need particular support when learning about CT and how to help teachers integrate CT into their classroom practices.

Our analysis of co-designed curriculum showed all teachers were able to integrate multiple computational tools that engage their students in CT-STEM practices. Teacher interviews and classroom observations show that teachers designed and implemented activities that reflect what they personally learned about coding, computational tools, and CT during the professional development. For example, Betty learned that computational models involve design decisions made by people and thus engaged her students in designing computational models where they write code for the behaviors that they expect to see. Further, because Penny found it "fun" and "cool" to see the code behind a model to understand how it works, she encouraged her students to similarly explore and break the code.

Taken together, these findings suggest that teachers benefited from both parts of our professional development: workshops in Week 1 and co-design in Weeks 2-4. Particularly, learning about specific computational tools and how to use them in the context of disciplinary content was important for four of the eight teachers, who reported being "excited" about integrating the tools into their classrooms. However, three of our teachers had little experience with coding and may not have the ability to integrate new computational tools into their classroom without the additional support provided in Weeks 2-4. At the end of the professional development, these three teachers reported learning to be comfortable with code and one teacher, Briana, even learned to love coding in the second week when she began working side-by-side with researchers to co-design curriculum. Moreover, multiple teachers viewed researchers as valuable thinking partners with expertise in CT. Hence, co-design may be an effective way to help teachers in integrate CT into their curriculum, particularly those with little or no CT experience. This finding aligns with prior work which showed that teachers' confidence in CT and ability to reach their curricular goals grew over a multi-week process of working with researchers as co-designers (Wu et al., 2020). We propose that additional research support integration of CT in K-12 by positioning teachers not only as learners of CT in workshops or

trainings, but as co-designers and collaborators who can augment existing STEM disciplinary content with CT in their classroom.

This work has the potential to engage more K-12 teachers and students in computational practices and tools by integrating CT into existing K-12 STEM classrooms. Through one summer professional development, teachers were empowered to develop and implement eight computationally enhanced STEM curricula for up to three weeks in mathematics and science classrooms. Our observations of these classrooms showed that the teachers talked about their experience during the 4-week professional development and leveraged what they learned about CT to help students become more comfortable with CT and engage in CT-STEM practices. Additional professional developments will help us identify what factors contribute to our success, beyond those specific to our eight teachers. This will help us scale this work to a larger population using in-person and online support on CT integration. By helping more teachers understand CT and computational tools, we can empower K-12 STEM teachers to engage their students in authentic scientific practice while also broadening participation in computing.

5. REFERENCES

Advocacy Coalition. (2018). *2018 State of Computer Science Education*. Retrieved October 8, 2019, from https://advocacy.code.org/

Allen, C. D., & Penuel, W. R. (2015). Studying Teachers' Sensemaking to Investigate Teachers' Responses to Professional Development Focused on New Standards. *Journal of Teacher Education, 66*(2), 136-149.

Ball, D., & Forzani, F. (2009). The Work of Teaching and the Challenge for Teacher Education. *Journal of Teacher Education, 60*(5), 497-511.

Brady, C., Orton, K., Weintrop, D., Anton, G., Rodriguez, S., & Wilensky, U. (2016). All Roads Lead to Computing: Making, Participatory Simulations, and Social Computing as Pathways to Computer Science. *IEEE Transactions on Education, 60*(1), 59-66.

Cabrera, K., Morreale, P., & Li, J. J. (2018). Computer Science+ Education: An Assessment of CS Professional Development. *Journal of Computing Sciences in Colleges, 33*(3), 141-147.

Coburn, C. E. (2005). Shaping Teacher Sensemaking: School Leaders and the Enactment of Reading Policy. *Educational Policy*, *19*(3), 476-509.

Cuny, J. (2012). Transforming High School Computing: A Call to Action. *ACM Inroads, 3*(2), 32-36.

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

Horn, M. S., Brady, C., Hjorth, A., Wagh, A., & Wilensky, U. (2014, June). Frog Pond: A Codefirst Learning Environment on Evolution and Natural Selection. *Proceedings of the 2014 conference on Interaction Design and Children*. ACM, 357-360.

Kelter, J., Peel, A. M., Bain, C., Anton, G., Dabholkar, S., Aslan, U., Horn, M. & Wilensky, U. (in press). Seeds of (r)Evolution: Constructionist Co-Design with High School Science Teachers. *Proceedings of Constructionism 2020.*

Peel, A. M., Dabholkar, S., Anton, G., Wu, S. P. W., Horn, M. S., & Wilensky, U. (in press). A case study of teacher professional growth through co-design and implementation of computationally enriched biology units. *Proceedings of 14th International Conference of the Learning Sciences (ICLS) 2020*. Nashville, TN.

Penuel, W. R., Fishman, B. J., Cheng, B. H., & Sabelli, N. (2011). Organizing Research and Development at the Intersection of Learning, Implementation, and Design. *Educational Researcher, 40*(7), 331-337.

Sengupta, P. and Wilensky, U. (2005). *NetLogo Electrostatics model.* Retrieved December 1, 2019, from http://ccl.northwestern.edu/netlogo/models/Electrostatics

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147.

Wilensky, U. (1997). *NetLogo Fire model*. Retrieved December 1, 2019, from http://ccl.northwestern.edu/netlogo/models/Fire

Wilensky, U. (1999). *NetLogo*. Retrieved December 1, 2019, from http://ccl.northwestern.edu/netlogo/

Wilensky, U. (2003). Statistical Mechanics for Secondary School: The GasLab Multi-agent Modeling Toolkit. *International Journal of Computers for Mathematical Learning*, *8*(1), 1-41.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering Computational Literacy in Science Classrooms. *Communications of ACM*, *57*(8), 24-28.

Windschitl, M., Thompson, J., Braaten, M., & Stroupe, D. (2012). Proposing a Core Set of Instructional Practices and Tools for Teachers of Science. *Science Education*, *96*(5), 878-903.

Wu, S.P.W., Anton, G, Bain, C, Peel, A.M., Horn, M.S. & Wilensky, U. (2020). Engage teachers as active co-designers to integrate computational thinking in STEM classes. Presented at NARST Annual International Conference (NARST 2020). Portland, Oregon.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

The Effect of Teacher Interventions and SRA Robot Programming on the Development of Computational Thinking

Nardie FANCHAMPS[1], Marcus SPECHT[2*], Paul HENNISSEN[3*], Lou SLANGEN[4*]
[1, 4]Fontys University of Applied Science, Netherlands
[2]Delft University of Technology, Netherlands
[3]Zuyd University of Applied Science, Netherlands
nardie.fanchamps@fontys.nl, M.M.Specht@tudelft.nl, paul.hennissen@zuyd.nl, l.slangen@fontys.nl

ABSTRACT
The implementation of programming in primary education is currently receiving a considerable amount of attention in the context of developing 21st century skills and digital literacy. The application of programmable robots is a playful integration of developing programming skills and computational thinking. Once pupils understand the basics of robot programming, they can solve challenging new programming tasks themselves without the teacher taking over too much of the learning process. It is therefore worthwhile to investigate the extent to which teachers' instructional approach and guiding interventions influence the development of pupils' computational thinking. Furthermore, programming robots have some special affordances for educational purposes as robots typically have to be programmed to interact with their environment. Little evidence is known to which extent programmable robots using the SRA-approach contribute to the development of computational thinking skills among primary school pupils. The use of Sense, Reason and Act (SRA) programming includes the application of loops, routines and conditionals when controlling actuators on the basis of sensory information with which the robot can anticipate changes in the environment. Our findings indicate that teachers versus students experience the way of teaching (perceived monitoring and scaffolding) significantly different when programming robots. We make recommendations as to which competences the guiding teacher needs. It is also shown that programming robots using an SRA-approach contribute to the development of specific characteristics (*reformulating problems, problem decomposition, abstraction, algorithms and procedures & parallelisation)* of computational thinking.

KEYWORDS
programming, sense-reason-act, robotics, computational thinking, teacher interventions

1. INTRODUCTION
In all educational sectors from primary school to higher education one key question is how to integrate the development of computational thinking in the curriculum. Little is known about the possibilities and impact and there are many questions such as: what contribution does learning to program make to cognitive development, what is the relationship between programming and computational thinking, and what is the influence of the teacher's actions in this regard (McCombs, Lauer, & Peralez, 1997; Morehead & LeBeau, 2005).

Computational thinking is the ability to solve complex problems using the basic concepts of computer science (Wing, 2006). The integration of programming in education is a way to stimulate the development of computational thinking among pupils and requires specific teacher competences (Lye & Koh, 2014). In our explorative research project we therefore investigate to what extent the way of providing instruction and teacher interventions (e.g. asking questions, giving a hint, showing/following, taking over) during teaching programming influences the development of computational thinking (e.g. algorithmic thinking, problem decomposition, debugging, parallel thinking).

Sense, Reason and Act (SRA) programming is a special approach of programming. It requires the creation of programs that implement control structures using sensors and actuators with the application of variables, loops and conditionals, for example when using programmable robots (Slangen, 2016). The SRA-loop is the process whereby detection (*sensing*) continuously generates new information that is entered into a logic reasoning component (*reason*), which subsequently leads to the resulting actions (*act*) (Lith, 2006).

Robots programmed using the principle of SRA can react to changes in their environment on the basis of sensory information. This requires pupils to think anticipatively in the task solution. Programming according to the principle of sense-reason-act requires problem-solving skills and abstract thinking. Pupils' programming of robots based on the SRA-principle requires skills related to creative thinking and critical thinking, such as: analysing, elaborating, causal reasoning, synthesizing, imagining, etc. This requires the teacher to provide a pedagogical space to apply these skills in practice. This demands an environment that is strongly linked to inquiry-based learning (Slangen, 2016; Valcke, 1985).

Teaching to program SRA needs special teacher strategies and competences and is more than just a transfer of knowledge (Slangen, 2016). Moreover, very few teachers have the experience and skills to conduct this kind of activity (Breed, 2003). Bers, Ponte, Juelich, Viera, and Schenker (2002) have found that teachers who start with a constructivist instructional approach when teaching programming, quickly revert to a more directive way of teaching to provide guidance to learners when solving complex problems. Slangen (2016) recommends that teachers best support such a learning process by means of a scaffolding-based approach and to design a learning environment to support children in their explorations and to scaffold learning (Sullivan & Bers, 2016). In that sense learning to program is more effective when the learner can construct his own knowledge from guided programming experiences. It is the teacher who has to set up a well-defined learning space and should apply appropriate guidance that allows pupils to gain a deeper understanding of how to program (Fanchamps, Slangen, Hennissen, & Specht, 2019). Thus, learning must be active; pupils must construct knowledge assisted by guidance from the teacher and best also with feedback from other pupils (Buitrago Flórez et al., 2017). Furthermore, in order to become familiar with complex programming such as SRA, a scaffold is required or a research-based structure must be followed.

Previous research shows that teacher interventions during programming lessons influence pupils' decision making skills

in solving programming problems (Fanchamps, 2016), and have a positive effect on the development of computational thinking skills of primary school pupils (Pat-El, Tillema, Segers, & Vedder, 2013). Teachers often intervene in a directive style; they start from their own insight and often provide an overload of information too quickly (Petrou & Dimitrakopoulou, 2003). For teachers, it seems difficult to be reluctant when pupils have to solve a problem using programming (Valcke, 1985). This reluctance on the teachers' part seems to be an important prerequisite for pupils to be able to learn inquisitively and to take problem-solving action when working with programming environments (Slangen, 2016). Furthermore, in programming education there is a need for different teacher behaviour (Sentance & Csizmadia, 2017). It is expected that teachers who provide guidance using a scaffolding based approach (asking questions, providing help when needed and in the way needed) will be able to get pupils further in the development of programming than teachers who use a more direct approach.

Considering previous research, we were interested to explore the following research questions:

1. To what extent does the teacher's own estimated instructional approach correspond with the pupils' perception of that instructional approach?
2. Which instructional approach (directive instruction - scaffolding) do teachers mainly use to influence the learning of programming in a robotics SRA learning environment?
3. To what extent does SRA-programming with Lego Robots contribute to the development of computational thinking skills?

2. METHOD

To investigate our research questions we used a pretest-posttest design. This includes a) among pupils pre-measurement of computational thinking skill and, for teachers, the extent of their own self-assessment of support, b) a robotics-intervention, and c) a post-test among pupils measuring computational thinking skills and the extent to which teacher support is perceived. During the execution, the instructional approach used by the teacher and the type of interventions were recorded. Among pupils, the ability to program was measured and the level of quality of the constructed program was assessed. We also measured the difference in terms of the support provided by the teacher and how it was perceived by the pupils.

2.1. Participants

This exploratory research was conducted among pupils and teachers of a Dutch primary school. Programming sessions with both primary school teachers and primary school pupils were organised. To make an inventory of the instructional approaches used, and in order to measure which interventions were mainly applied by the teachers, we chose to incorporate pupils and teachers from grade 3 to grade 8 in this research. We would like to indicate that that these Dutch grade levels correspond with age 6 to 12.

2.2. Materials

We used Lego Robots EV-3 as a SRA programming environment. This environment offers possibilities to control a robot using sensors and actuators. The programming of this environment is characterized by visual programming by means of drag and drop command blocks. By manipulating the programming variables per block and putting them in a specific order, pupils construct their program. This visually programming environment is also suitable for use in primary education (Korkmaz, 2018).

In order to reflect the self-assessment of the teacher's pedagogical level of support, and to determine how pupils experienced this support as such (research question 2), the Assessment for Learning questionnaire was used (Pat-El et al., 2013). This validated questionnaire, consisting of a separate teacher part and a specific pupil part, which measures individually the perception of both teachers and pupils, is based on a 5-point Likert scale (range from *"completely disagree" – "completely agree"*) and consists of two categories: "Experienced monitoring/supervising the learning process" (Perceived monitoring - 16 items) and "Experienced level/application of scaffolding" (Perceived scaffolding - 12 items). This questionnaire includes questions such as: *"The teacher provides opportunities to set learning objectives"* and *"The teacher provides hints to help understand the subject"*. The questionnaire is deliberately presented to both teachers (teacher version) and pupils (pupil version), so that the perception of teachers can be tested against the perception of pupils. This allows, by monitoring differences in teaching and learning, to visualise the influence of a different pedagogical approach (in this case a directive approach compared to a scaffolding approach).

In order to measure an effect on computational thinking among pupils (research question 3) the validated Computational Thinking test (Román-González, Pérez-González, & Jiménez-Fernández, 2017) was used as a pre- and post-measurement among pupils. This questionnaire consists of 28 questions in which pupils have to link programming commands to various situations (and vice-versa), measuring characteristics of computational thinking.

2.3. Procedure

After teachers followed three training sessions provided by the researcher, in which they learned how to program Lego EV3 robots, the teacher independently and without the intervention of the researcher taught five programming lessons using the Lego environment. In these 5 lessons the teacher guided the pupils, who worked together in pairs, to solve various robotics programming problems. Pupils were confronted with 20 programming problems that became more and more difficult. The teacher specified the task the robot had to perform using the computer program created by pupils. The teacher used a personal instructional approach and his or her own interventions to guide pupils through the learning process.

By means of observation to determine which instructional approach (directive instruction or scaffolding) the teacher has used and what kind of interventions the teacher has applied (asking questions, giving a hint, showing/following or taking over the learning process), the researcher recorded a) the type of instructional approach and which type / to what extent interventions have been used by the teacher to identify what the effect was (research question 1), and b) the extent to which SRA-programming by pupils has been applied in solving programming problems.

3. RESULTS AND DATA-ANALYSIS

Examination of the results of the first research question concerning the influence of the instructional approach on learning to program indicates that the instructional approach of

the teacher is of great influence. An analysis of the observations shows that pupils from grade 1 to grade 6 are perfectly capable of functionally programming Lego Robots by using the computer program provided. Pupils who have been guided according to a scaffolding approach frequently make use of the knowledge they have already acquired during new programming tasks are further advanced in a problem-solving approach compared to guidance according to a directive instructional approach.

Further indications show that teachers who use a scaffolding based instructional approach mainly promote a self-management and problem-solving, self-analytical capacity of learners, and in particular use questioning and hints. Teachers who use a directive instructional approach mainly promote procedural thinking, initiate functional programming and enhance pupils' success experience, and mainly make use of showing/following and taking over the learning process, and ask few questions.

The results concerning the second research question about the extent to which the teacher's estimated instructional approach matches the student's perception of that instructional approach, show a difference in estimation (*Table 1*).

Table 1. Assessment for Learning Questionnaire

		Perceived monitoring (16 items)			
Group	*n*	*M*	*SD*	*range*	*Mdn*
Teacher	13	3.93	.45	3.75 – 4.75	3.94
Pupil	21	3.68	.45	2.75 – 4.50	3.75
		Perceived scaffolding (12 items)			
Group	*n*	*M*	*SD*	*range*	*Mdn*
Teacher	13	4.15	.32	3.58 – 4.58	4.17
Pupil	21	3.76	.32	3.25 – 4.42	3.75

Note. n = number of respondents; M = average; SD = standard deviation; range = spread in measurement; Mdn = median

Pupils do not value the teachers' instructional approach as highly as the teachers themselves. Table 1 shows that the averages (*M*) on both categories (monitoring, scaffolding) of the questionnaire are very different and the difference for perceived scaffolding is statistically significant t (32) = 1.57, p = 0,001, 95% CI [0,15, 0,62]. This is indicatively more accurate to deduce from the median (*Mdn*), which for pupils in both categories is significantly lower, and from the range, in which the distribution of pupils' results is more dispersed. A closer, more detailed study of the collected questionnaires makes it quantitatively visible that teachers respond to more questions with the answer *"agree"* and *"completely agree"*, but that pupils give significantly lower scores for this, i.e. they experience the teachers approach differently. A comparison in perception between the two research groups is quite striking and can be a fundamental ground for the accompanying teacher to take a critical look at applied instructional approach and interventions used during programming lessons.

The answer to the third research question on the extent to which SRA-programming contributes to the development of computational skills can be found in Table 2.

Although not significant, the measured results (*Table 2*) show an increase in the characteristics of computational thinking skills (*reformulating problems, problem decomposition, abstraction, algorithms and procedures and parallelisation*) among pupils. In the post measurement, pupils solve more tasks correctly and therefore show a higher level of computational thinking skill compared to the pre-measurement. The computational thinking characteristics *"completion", "debugging" and "sequencing"* show in the post measurement a higher average score (*M*), a lower standard deviation (*SD*) and less spread in the measured values (*range*). Based on these results, the conclusion can be drawn that by application of SRA-programming pupils develop a higher level of computational thinking skills.

Table 2. Development of Computational Thinking Skills

	Pre-test (n = 21)			
Variable	*M*	*SD*	*range*	*Mdn*
Total (28)	1.01	.36	0.25 – 1.42	1.13
CT-skill completion	.33	.11	0.08 – 0.50	0.33
CT-skill debugging	.19	.11	0.00 – 0.33	0.11
CT-skill sequencing	.49	.19	0.17 – 0.75	0.46
	Post-test (n = 21)			
Variable	*M*	*SD*	*range*	*Mdn*
Total	1.14	.22	0.83 – 1.50	1.13
CT-skill completion	.35	.09	0.25 – 0.50	0.33
CT-skill debugging	.25	.10	0.08 – 0.42	0.10
CT-skill sequencing	.54	.13	0.25 – 0.67	0.58

Note. Variable = measurable value; Total = number of questions correct CT-questionnaire; completion = completed by CT; debugging = reformulate problems; sequencing = sequence; M = average; SD = standard deviation; range = spread in measurement; Mdn = median

4. CONCLUSION AND DISCUSSION

This research helps teachers who want to implement programming lessons in the classroom. Because learning to program seems to depend on the instructional approach and the appropriate interventions, the question arises whether a more directive instruction or a scaffolding-based approach is more appropriate for teaching programming. This requires a more inquiry-based approach for pupils, and for teachers knowledge of scaffolding and the guidance to be used as well as the type of interventions to be applied. The necessary competences can be developed through training and further experience.

This research makes the effect of the type of teacher intervention visible when teaching programming robots in the classroom. The results also contribute to sharpening the definition of what computational thinking means for the development of primary school pupils. It leads to four recommendations:

- First, LEGO Robotics can be used as a programming learning environment.
- Second, specific programming lessons can increase classroom yields.
- Third, SRA-programming contributes to the development of computational thinking skills of primary school pupils with a transfer to other disciplines and educational areas/other primary school subjects.
- Fourth, a thorough implementation of teaching programming in the classroom requires a further professionalisation of the teacher on scaffolding and guidance with specific interventions.

4.1. Limitations and further directions

Despite the limited number of respondents in this study, a measurable development (although not significant) in computational thinking skills has been demonstrated. In follow-up research, larger numbers of respondents will be used for which it is expected that significant results can be demonstrated. It is also worthwhile to investigating whether other types of programming environments generate the same yields.

The indications from this research show that it is relevant for teachers to become more aware of the fact that the nature of support is important to help pupils further in SRA-programming. This means, on the one hand, that teachers need to have their own content programming insights and, on the other hand, that they can also use guidance skills.

In order to further develop the construct of the SRA approach theoretically, it is relevant to further investigate and develop the relationship between computational thinking and SRA programming.

4.2. Acknowledgments

This research was financed with a grant obtained for a KIEM-Sia project study (SVB/KIEM.21V01.051). The authors would like to thank Heutink Netherlands for the donation of the required sets Lego Mindstorms and their cooperation.

5. REFERENCES

Bers, M. U., Ponte, I., Juelich, C., Viera, A., & Schenker, J. (2002). Teachers as Designers: Integrating Robotics in Early Childhood Education. *Information Technology in Childhood Education Annual, 2002*(1), 123-145.

Breed, B. (2003). The Reflective Abilities of Expert and Novice Learners in Computer Programming. *Proceedings of British Educational Research Association Annual Conference.* Edinburgh: Heriot-Watt University.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking through Programming. *Review of Educational Research, 87*(4), 834-860.

Fanchamps, N. (2016). *De Invloed van SRA Programmeren op Mathematisch Redeneren en Zelfeffectiviteit met Lego Robotica in Twee Instructievarianten*. Master's thesis, Open Universiteit, Heerlen.

Fanchamps, N., Slangen, L., Hennissen, P., & Specht, M. (2019). The Influence of SRA Programming on Algorithmic Thinking and Self-Efficacy Using Lego Robotics in Two Types of Instruction. *International Journal of Technology and Design Education*, 1-20.

Korkmaz, Ö. (2018). The Effect of Scratch-and Lego Mindstorms Ev3-Based Programming Activities on Academic Achievement, Problem-solving Skills and Logical-mathematical Thinking Skills of Students. *Malaysian Online Jou1rnal of Educational Sciences, 4*(3), 73-88.

Lith, P. V. (2006). *Masterclass Robotica.* Limbricht: Elektuur.

Lye, S. Y., & Koh, J. H. L. (2014). Review on Teaching and Learning of Computational Thinking through Programming: What is next for K-12? *Computers in Human Behavior, 41*, 10.

McCombs, B., Lauer, P., & Peralez, A. (1997). *Researcher Test Manual for the Learner-Centered Battery (Grades 6-12 Version). A Set of Self-Assessment and Reflection Tools for Middle and High School Teachers.* Mid-Continent Regional Educational Lab.

Morehead, P., & LeBeau, B. (2005). The Continuing Challenges of Technology Integration for Teachers. *Essays in Education, 15*(1), 10.

Pat-El, R. J., Tillema, H., Segers, M., & Vedder, P. (2013). Validation of Assessment for Learning Questionnaires for Teachers and Students. *British Journal of Educational Psychology, 83*(1), 98-113.

Petrou, A., & Dimitrakopoulou, A. (2003). Is Synchronous Computer Mediated Collaborative Problem-solving "Justied" only when by Distance? Teachers' Point of Views and Interventions with Co-located Groups, during every day Class Activities. *Proceedings of International Conference on Computer Support for Collaborative Learning.* Dordrecht: Springer, 441-450.

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.

Sentance, S., & Csizmadia, A. (2017). Computing in the Curriculum: Challenges and Strategies from a Teacher's Perspective. *Education and Information Technologies, 22*(2), 469-495.

Slangen, L. (2016). *Teaching Robotics in Primary School. (PhD).* Eindhoven University of Technology, Eindhoven. Retrieved September 9, 2019, from https://pure.tue.nl/ws/files/25754482/20160630_CO_Slangen.pdf

Sullivan, A., & Bers, M. U. (2016). Robotics in the Early Childhood Classroom: Learning Outcomes from an 8-week Robotics Curriculum in Pre-kindergarten through Second Grade. *International Journal Technology and Design Education, 26*, 17.

Valcke, M. (1985). Praktische ervaringen met het leren programmeren in de klas - Karakteristieken van de leerkrachtinterventie. *Proceedings of Programmeertalen en courseware-aanmaak*, 61-90.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 3.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Preservice Teachers' Views of Computational Thinking: STEM Teachers vs non-STEM Teachers

Chee-Kit LOOI[1], Shiau Wei CHAN[2], Wendy HUANG[3], Peter SEOW[4], Longkai WU[5]
[1,2,3,4,5]National Institute of Education, Nanyang Technological University, Singapore
cheekit.looi@nie.edu.sg, shiauwei.chan@nie.edu.sg, wendy.huang@nie.edu.sg, peter.seow@mie.edu.sg, longkai.wu@nie.edu.sg

ABSTRACT
This study was performed to explore the views of preservice teachers of computational thinking (CT) through a pilot survey. A total of 329 preservice teachers from the National Institute of Education Singapore took part in this pilot survey. These preservice teachers were trained to teach STEM and non-STEM subjects. The overall findings showed that the preservice teachers do not yet have an adequate understanding of CT. Most of them perceived CT as logical thinking or reasoning. This is followed by no idea or no understanding or not sure, using ICT or computer, coding or programming, problem-solving and so forth. Besides that, STEM preservice teachers had different views of CT compared to non-STEM preservice teachers. These initial views of CT among the preservice teachers can serve to inform the design of teacher preparation programs, policies and syllabus materials to support the preservice teachers to infuse CT into their future teaching practices.

KEYWORDS
computational thinking, preservice teacher, view, pilot survey

1. INTRODUCTION
In Singapore, developing computational capabilities is one of the key enablers for the Smart Nation initiative. Several programs have been conducted to introduce and enhance computational thinking (CT) skills and coding abilities among the Singaporean, from pre-school students to adults. Nevertheless, one of the main concerns is how to best prepare and support teachers to incorporate CT into their teaching in the classroom (Yadav, Hong, & Stephenson, 2016). We are a research group that explores the design of new programs to train the preservice teachers and in-service teachers in CT. A recent program that has been implemented was CTFest: Sharing and Learning about CT which sponsored by a grant from the Google Data Centre Community Fund. During the CTFest, featured talks and discussions were held for the teachers to learn about the best practices in the teaching of CT. The attendees included teachers of computer science, computer programming and applications, computing, design and technology, and computing-related Applied Learning Programmes; colleagues from Curriculum Planning & Development (CPDD) of MOE, polytechnics lecturers and Singapore Science Centre. Industry partners were also invited to exhibit their work in computing education.

Educational experiences are needed for the teachers from all levels to prepare them well to teach CT concepts effectively. Knowing the standpoints of preservice teachers towards CT can serve as applicable resources for creating teacher preparation programs, policies, and syllabus materials to support the teachers to integrate CT into their teaching practices (Rich, Yadav, & Schwarz, 2019). Thus, this study is intended to determine preservice teachers' views of CT through a pilot survey. It is led by these research questions:

(a) How do preservice teachers view computational thinking?

(b) What are the differences in the view of computational thinking between STEM and non-STEM preservice teachers?

2. LITERATURE REVIEW
CT has the potential to promote problem-solving skills and capabilities among the students as they start to think in new ways (Yadav et al., 2014). Therefore, the students should be taught to understand computational procedures and develop skills for representing and abstracting information (Lu & Fletcher, 2009). Hemmendinger (2010) also claimed that the aim of teaching CT was "to teach them how to think like an economist, a physicist, an artist, and to understand how to use computation to solve their problems, to create, and to discover new questions that can fruitfully be explored" (p. 4). Yadav et al. (2011) asserted that teacher education was one discipline where CT would have a noteworthy effect on K-12 education. This was because if the preservice teachers were able to present their CT ideas in the teaching, the students would have the superior experience of computing in general.

Some works have been executed to determine how preservice teachers view CT. For instance, Chang and Peterson (2018) accomplished a study to identify the perceptions of CT among preservice teachers. The preservice teachers define CT as an important literacy, with elements of thinking in a logical series and steps, thinking for solution and creating strategies, and demonstrating thinking. Furthermore, Bower and Falker (2015) conducted a study to investigate the understanding of CT among preservice teachers. The results indicated that almost one-third of the preservice teachers regarded CT as problem solving using technology, and utilizing technology. Another study was done by Yadav et al. (2014) to evaluate the understanding of CT among preservice teachers. The preservice teachers perceived that CT as heuristics and problem solving, algorithms, use of computers or technology, and critical thinking.

3. METHOD
3.1. Respondents

329 preservice teachers in the National Institute of Education, Singapore who participated in this study. They had just completed their year-long teaching training courses, and were about to go to school for their practicum before graduation as a teacher. They were trained to teach at least two subjects. Most of them (*n*=147, 44.7%) had the curriculum subject of English Language / Literature / General Paper. This is followed by the subjects of Mother Tongue (*n*=92, 28%), Mathematics (*n*=82, 24.9%), Science (*n*=59, 17.9%), History / Social Studies / Geography / Economics (*n*=76, 23.1%), Art / Music / Drama (*n*=21, 6.4%), Computer Applications (*n*=3, 0.9%), Principles of Accounts (*n*=3, 0.9%), Elements of Business Skills (*n*=2, 0.6%), Character and Citizenship Education (*n*=2, 0.6%), Social Studies (*n*=1, 0.3%), and French (*n*=1, 0.3%). 121 of them were trained to teach STEM subjects including Mathematics, Science, and Computer Applications. We considered them as STEM preservice teachers if they were trained and prepared to teach at least one STEM subject. Meanwhile, 208 of them were trained in teaching non-STEM subjects. All of these preservice teachers were required to attend a one and half hour long CT introductory session as part of their Beginning Teacher Orientation Programme.

3.2. Pilot Survey

At the beginning of the session program on CT, the respondents had to complete a pilot survey which consisted of two questions. The first question was in multiple-choice format, and the second question was open-ended. The first question was "What subject areas have you been prepared to teach?" and the second question was "What is your current understanding of computational thinking?" The respondents answered the questions using google forms. The responses of the second question were analyzed using an open coding approach to identify the preliminary analytic categories. If the responses contained multiple features, they were put under two or more categories, for instance 'Problem solving with the use of computers' was included in the categories of 'problem-solving' and 'using ICT/computer' (Bower & Falkner, 2015).

4. FINDINGS

4.1. Preservice Teachers' Views of CT

Table 1 presents the views of CT among preservice teachers. In Table 1, we notice that the majority of the preservice teachers perceived that CT was logical thinking or reasoning with a total frequency of 80. It was surprising that a number of preservice teachers (*n*=43) did not have any idea or understanding of CT. Most of them (*n*=38) also regarded CT involve the use of ICT or computer. 32 of the respondents viewed CT as coding or programming. Besides that, the preservice teachers also thought that CT was related to problem-solving, with a frequency of 30 and CT was systematic or systematic thinking with the frequency of 19. They deemed that CT was thinking or thinking process (*n*=13), computation or calculation (*n*=10), and algorithm (*n*=10). This is followed by mathematics (*n*=8), analytical thinking or analytical thinking (*n*=8), and programming (*n*=8). Six of the respondents conceived that CT was step by step and thinking like a computer. Methodical thinking and analysis were perceived as CT with a frequency of 4 respectively. Furthermore, CT was also considered as computing skills or principles (*n*=3), sequencing or sequential thinking (*n*=3), artificial intelligence (*n*=3), structured or structured thinking (*n*=2), and using software (*n*=2). The other CT views with a frequency of 1 were including stepwise thinking, thinking like a bot, thinking like a coder, rational thinking, IT-related thinking, engineering-related and so on.

Table 1. Preservice Teachers' views of CT

No	CT Views	Frequency
1	Logical thinking/reasoning	80
2	No idea/No understanding/Not sure	43
3	Using ICT/computer	38
4	Coding/Programming	32
5	Problem solving	30
6	Systematic/Systematic thinking	19
7	Thinking/Thinking Process	14
8	Computation/Calculation	10
9	Algorithm	10
10	Mathematics	8
11	Analytical/Analytical thinking	8
12	Steps/Step by step	6
13	Thinking like a computer	6
14	Methodical thinking	4
15	Analysis	4
16	Computing skills/principles	3
17	Sequencing/Sequential thinking	3
18	Artificial intelligence	3
19	Structured/Structured thinking	2
20	Using software	2
21	Algorithmic thinking	1
22	Strategy	1
23	Robots	1
24	JavaScript	1
25	Out of box thinking	1
26	Recursion	1
27	Stepwise thinking	1
28	Giving instructions	1
29	Rational conclusions	1
30	Commands	1
31	Thinking like a bot	1
32	Thinking like a coder	1
33	Thinking like a machine	1
34	Numbers	1
35	Higher order thinking	1
36	Excel	1
37	Statistics	1
38	Permutation	1
39	Combinations	1
40	Configurations	1
41	Decision making	1
42	Directions for machines	1
43	Computer terminology	1
44	Technical	1
45	Algebraic thinking	1
46	Binary codes	1
47	Iterations	1
48	Processing thoughts effectively	1

49	Thinking procedurally	1
50	Procedure	1
51	Mathematical thinking	1
52	Rational thinking	1
53	Memory work	1
54	Managing complexity	1
55	Using models	1
56	Proactive thinking	1
57	ICT lesson	1
58	IT-related thinking	1
59	Optimization	1
60	Function	1
61	Graph theory	1
62	Standardized thinking	1
63	Solutions	1
64	Making teaching easier	1
65	Engineering-related	1

4.2. Comparison of the view of CT between STEM and non-STEM preservice teachers

Based on Table 1, the views of CT that had a frequency of 2 or more than 2 were included in the analysis to compare the differences in the view of CT between STEM and non-STEM preservice teachers. From Table 2 and Figure 1, it can be observed that more STEM preservice teachers (28.9%) viewed CT as logical thinking or reasoning than non-STEM preservice teachers (21.6%). Most of the non-STEM preservice teachers (15.9%) did not know about CT compared to that of STEM preservice teachers (8.3%). When compared to non-STEM preservice teachers, the STEM preservice teachers were more likely to consider CT as coding or programming (10.7%), systematic or systematic thinking (8.3%), thinking or thinking process (5.0%), computation or calculation (5.0%), mathematics (2.5%), analytical or analytical thinking (3.3%), steps or step by step (3.3%), thinking like a computer (2.5%), methodical thinking (1.7%), and using software (0.8%). The percentage for the non-STEM preservice teachers for these ten CT views was 9.1%, 4.3%, 3.8%, 1.9%, 2.4%, 1.9%, 1.0%, 1.4%, 1.0%, and 0.5% respectively. In the contrast, the STEM preservice teachers were less likely to regard CT as using ICT or computer (10.7%), algorithm (1.7%), analysis (0.8%), computing skills or principles (0.8%), and sequencing or sequential thinking (0.8%). The percentage for non-STEM preservice teachers was higher than STEM preservice teachers for these five CT views, i.e. 12%, 3.8%, 1.4%, 1.0%, and 1.0%. Both STEM and non-STEM preservice teachers deemed CT as problem-solving which had the same percentage of 9.1%. Non-STEM preservice teachers considered CT as artificial intelligence and structured or structured thinking with a percentage of 1.4% each, but there was 0% for the STEM preservice teachers.

Table 2. Comparison of views of CT between STEM and non-STEM preservice teachers

CT Views	STEM	Non-STEM
Logical thinking/reasoning	28.9	21.6
No idea/No understanding/Not sure	8.3	15.9
Using ICT/computer	10.7	12
Problem solving	9.1	9.1
Coding/Programming	10.7	9.1
Systematic/Systematic thinking	8.3	4.3
Thinking/Thinking Process	5	3.8
Computation/Calculation	5	1.9
Algorithm	1.7	3.8
Mathematics	2.5	2.4
Analytical/Analytical thinking	3.3	1.9
Steps/Step by step	3.3	1
Thinking like a computer	2.5	1.4
Methodical thinking	1.7	1
Analysis	0.8	1.4
Computing skills/principles	0.8	1
Sequencing/Sequential thinking	0.8	1
Artificial intelligence	0	1.4
Structured/Structured thinking	0	1.4
Using software	0.8	0.5

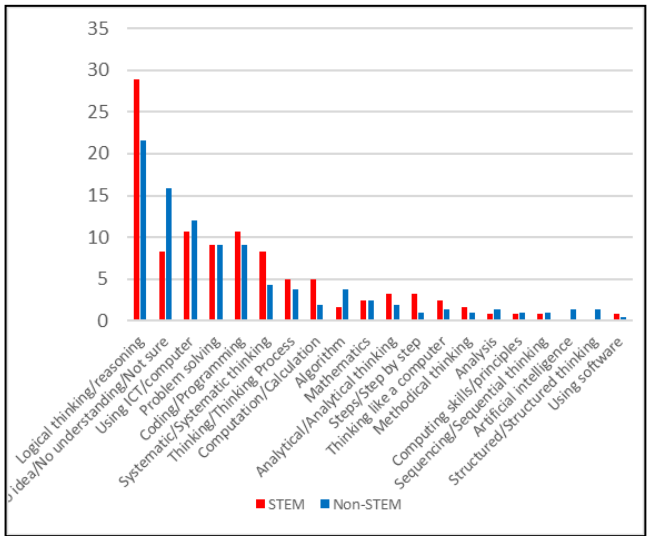


Figure 1. Comparison of views of CT between STEM and non-STEM preservice teachers

5. DISCUSSIONS AND CONCLUSION

The overall findings demonstrated that preservice teachers did not have a sufficient understanding of CT. This indicated that a lack of awareness of how CT skills can be incorporated into their teaching practices, thus implying that more work needs to be put in to expose them to knowledge and practices about the integration of CT in the classrooms. The majority of preservice teachers perceived that CT as logical thinking which is analogous with the result of a study from Chang and Peterson (2018) where CT is seen as thinking in logical steps. The preservice teachers had comparable responses with the study of Sands, Yadav and Good (2018) where CT involved problem-solving, logical thinking, thinking like a computer, mathematics, using ICT or computer, coding or programming, and algorithm. CT was regarded as problem solving and mathematics which is also consistent with the finding of Rich, Yadav and Schwarz's (2019) study. The preservice teachers were capable to determine the types of thinking connected with CT, such as analytical thinking, mathematical thinking, logical thinking, and structured thinking, which is compatible with the study of Bower and Falkner (2015). By referring to Table 1, some of the preservice teachers were able to identify the concepts and elements that related to CT,

for example, algorithmic thinking, iterations, function, using models, sequencing or sequential thinking, and thinking procedurally. However, there were some responses in Table 1 that did not relate to CT or had no clear meaning, such as JavaScript, configurations, memory work, solutions, and graph theory.

Preservice teachers, regardless of STEM and non-STEM, ought to have similar thoughts about CT. However, in this study, it was found that the STEM preservice teachers had different views of CT compared to non-STEM preservice teachers. Unlike the non-STEM preservice teachers, the STEM preservice teachers were more likely to perceive CT as logical thinking or reasoning, coding or programming, systematic or systematic thinking, thinking or thinking process, computation or calculation, analytical or analytical thinking, steps or step by step, thinking like a computer, methodical thinking, and using software. On the other hand, the STEM preservice teachers were less likely to regard CT as using ICT or computer, algorithm, analysis, computing skills or principles, and sequencing or sequential thinking. More non-STEM preservice teachers did not have an idea or understanding concerning CT. This is most likely because STEM preservice teachers may have more exposure to Computing courses in their tertiary education before joining the preservice teaching course. Both STEM and non-STEM preservice teachers had the same response for CT as problem-solving. Two remarkable differences of view of CT between STEM and non-STEM preservice teachers were the artificial intelligence and structured or structured thinking as none of the STEM teachers gave these responses. This could attributed to non-STEM teachers perception that CT is related to the use of technology.

In some countries such as the United Kingdom, efforts have been made to integrate CT into all subjects at all levels, If teachers have pre-conceptions of CT that differ from the concepts of CT, it would be difficult to require teachers to integrate CT into the curriculum, Our findings of this study can serve as useful resources to help create teacher preparation programs, policies, and syllabus materials to help the preservice teachers to embed CT into their future classrooms. It is proposed to implement more teacher preparation programs on CT for the preservice teachers to help them to be more familiar with the CT concepts and have a better grasp on how CT can be employed in their future teaching. The teacher preparation programs play an important role in making a large-scale shift towards embedding CT into K-12 education. Hence, preservice teachers should have opportunities to experience CT during their preservice courses. During the course, tangible or practical examples of how to integrate CT into different subject areas should be provided. Future research needs to include a bigger sample of participants with diverse demographics. Besides, this pilot survey does not tell us much about the views of preservice teachers in detail. In future research, we can further investigate where the teachers are getting their ideas about CT from through in-depth interviews and elaborate on them.

6. ACKNOWLEDGEMENT

This work was supported by a grant (OER 10/18) from the Office of Educational Research, NIE.

7. REFERENCES

Bower, M., & Falkner, K. (2015). Computational Thinking, the Notional Machine, Pre-service Teachers, and Research Opportunities. *Proceedings of the 17th Australasian Computer Education Conference (ACE2015) Sydney*, 37-46.

Chang, Y., & Peterson, L. (2018). Pre-service Teachers' Perceptions of Computational Thinking. *Journal of Technology and Teacher Education*, 26(3), 353-374.

Hemmendinger, D. (2010). A Please for Modesty. *ACM Inroads, 1*(2), 4–7.

Lu, J., & Fletcher, G. (2009). Thinking about Computational Thinking. *SIGCSE 2009, Chattanooga.* ACM, 260-264.

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational Thinking, Mathematics, and Science: Elementary Teachers' Perspectives on Integration. *Journal of Technology and Teacher Education, 27*(2), 165-205.

Sands, P., Yadav, A., & Good, J. (2018). Computational Thinking in K-12: In-service Teacher Perceptions of Computational Thinking. *Computational Thinking in the STEM Disciplines.* Cham: Springer, 151-164.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing Computational Thinking in Education Courses. *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. ACM, 465–470.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education, 14*(1), 1–16.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends*, *60*(6), 565-568.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and IoT

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

CT-6E Model for Developing the IoT Teaching Activity

Hsien-Sheng HSIAO[1*], Chung-Pu CHANG[2*]
[1,2] Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taiwan
[1]Chinese Language and Technology Center., National Taiwan Normal University, Taipei, Taiwan.
[1]Institule for Research Excellence in Learning Sciences, National Taiwan Normal University, Taipei, Taiwan.
etlab.paper@gmail.com, enzoapu@gmail.com

ABSTRACT
Information technology is developing rapidly, and there is a large demand for scientific and technological talents, such as the Internet of Things, Big data, Artificial Intelligence, etc. Therefore, how to cultivate the next generation with information technology concept and program implementation technology ability and computational thinking ability has become a very important and urgent problem to be solved in countries around the world. This study proposes that a set of IoT teaching activities using the CT-6E model will be developed. Through 8 weeks of inquiry learning and practical teaching, students will learn the concepts of the Internet of Things, explore applications in life, write programs and assemble electronic components, and make IoT topics that integrate various modules. In addition, an empirical study is planned to explore whether this course can more effectively improve students' IoT learning effectiveness, computational thinking ability, and self-efficacy than traditional courses.

KEYWORDS
Internet of Things, computational thinking, self-efficacy, 6E model

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

運用 CT-6E 模式發展高中生之物聯網教學活動規劃

蕭顯勝 [1*]，張仲樸 [2*]
[1,2] 科技應用與人力資源發展學系，臺灣師範大學，台灣
[1] 臺灣師範大學學習科學跨國頂尖研究中心，台灣
[1] 臺灣師範大學華語文與科技研究中心，台灣
etlab.paper@gmail.com，enzoapu@gmail.com

摘要
資訊科技發展快速，有大量物聯網、大數據、人工智慧等科技人才需求，因此如何培育具備資訊科技概念和程式實作技術能力、運算思維能力的下一代，成為各國相當重要且急迫需要解決的問題。本研究提出將發展出一套運用 CT-6E模式之物聯網教學活動，透過八週的探究學習及實作教學，讓學生學到物聯網的概念、探索生活中的應用、動手撰寫程式並組裝電子元件，最後實作出整合各模組的物聯網專題。並規劃一實證研究，來探討經由此課程能否比一般課程更有效地提升學生的物聯網學習成效、運算思維能力及自我效能。

關鍵字
物聯網；運算思維；自我效能；6E 模式

1. 前言
科技發展一日千里，隨著物聯網（Internet of Things, IoT）、大數據（Big Data）、人工智慧（Artificial Intelligence, AI）等技術發展（Identifies Top 10 Strategic IoT Technologies and Trends, 2018），未來五年「通訊暨物聯網裝置」、「大數據分析」與「資料服務」等產業，每年人才需求大於 10,000 人（國發會，2017）。面臨快速發展的科技現況及全球化競爭時代來臨，如何強化國家人才能力的培養及產業競爭力，為重要之議題（PwC, 2019）。因此台灣教育部（2019）調整了《十二年國民教育科技領域課程綱要》，將「資訊科技」設為國、高中階段之必修科目，強調學生培養運算思維（Computational Thinking, CT）能力的重要性。

過去大多數研究在 CT 教學活動中採用專題導向式學習、問題專題導向式學習及合作學習（Hsu, Chang, & Hung, 2018），代表這種讓學生透過步驟探究進行學習是有效的策略。6E 模式是一種以學生為中心，同樣為藉由步驟讓學生探究學習的教學策略（Barry, 2014），過程中規劃使學生容易增加信心及成就感的實作練習，可改善學生在程式設計課程上，自我效能低落的問題。

本研究發展出一套運用CT-6E模式的物聯網教學活動，能夠更加地提升學生的物聯網學習表現、運算思維及自我效能，進而培育更多具有未來競爭力的相關人才。研究問題如下：

（1）如何發展 CT-6E 模式的物聯網教學活動？

（2）如何評估學生在 CT-6E 模式與一般教學模式上的學習成效差異？

2. 文獻探討

2.1. 物聯網教學
在物聯網教學環境中，學生將學習到數理科學與工程教育的跨領域知識，同時會獲得科技應用與程式設計等實作機會（Kortuem, Bandara, Smith, Richards, & Petre, 2012）。

Mavroudi、Divitini、Gianni、Mora 與 Kvittem（2018）的研究中，教學者使用體驗式學習策略，讓學生扮演物聯網應用設計師，發想以智慧城市為主題的創意題目和解決方案。最後的調查結果發現，參與者認為學習物聯網知識、智能城市概念和獲得解決問題的技能等方面都非常令人滿意。

2.2. 運算思維
運算思維由 Wing（2006）提出，後續許多研究陸續發表出不同的定義或概念組合。本研究採過去學者或電腦科學組織在定義運算思維時所提到的概念中，較廣為人知的四大概念（Angeli et al., 2016; Barr & Stephenson, 2011; CSTA, 2011; Google for Education, 2015; Grover & Pea, 2013; Hsu, Chang, & Hung, 2018; Selby & Woollard, 2014），其定義如下：

1.分解：將資料、程序或問題分解為更小、易於處理的部分。

2.模式識別：觀察出資料中的模式、趨勢和規律。

3.抽象化：識別並提取可表達主要概念的相關訊息。

4.演算法設計：創建一程序性指令，用於解決相似的問題或執行一個任務。

Lai、Chen、Lai、Chang 與 Su（2019）將運算思維納入物聯網應用的課程規劃中，讓學生通過理解問題並分析解決步驟來找到不同領域問題的解決方案，結果發現能夠有效地提高學生的學習興趣和表現。

2.3. 自我效能
Bandura（1997）說明自我效能是個人在不同情境下的信心、信念程度。自我效能將會影響學生在解決問題時所表現出的努力程度（Gandhi & Varma, 2009; Psycharis & Kallia, 2017）。

Tsai 等人（2019）根據 Berland 與 Lee（2011）的運算思維框架，發展出一個更加泛化的自我效能量表，用以檢驗學生的程式設計自我效能感，有益於教育者或課程設計者在教學實踐和教育研究中應用。

2.4. 6E 模式

6E 模式是由美國國際科技與工程教師學會（International Technology and Engineering Educators Association, ITEEA）在 2013 年提出，包括六大步驟：參與（engage）、探索（explore）、解釋（explain）、實作（engineer）、深化（enrich）、評量（evaluate），教師扮演輔助的角色，讓學習者透過探究的過程，理解真實情境的問題，經由思考、設計並實作出解決方案（Barry, 2014）。

Chen 等人（2019）運用 6E 模式發展出四軸飛行器實作活動，讓學生動手寫程式、連結控制板與馬達，透過藍芽控制，實驗結果表明，能夠有效地提升學生的學習成效。Hsiao 等人（2019）則是運用 6E 模式發展出 S4A 機器人實作課程，實驗結果表明，學生的學習成效和運算思維能力均有所提高。

2.5. 文獻評析

經由前述文獻整理與探討可知，以物聯網為主題的教學活動或課程能夠培養學生的資訊科技各領域概念及運算思維；運用 6E 模式發展之課程，能夠提升學生在物聯網相關課程上的學習成效。因此，本研究欲將運算思維的概念融入課程，並與 6E 模式個步驟做搭配，最後發展出共八週之物聯網教學活動，以及後續實證研究規劃。

3. 教學設計與教案規劃

3.1. 物聯網課程

課程以物聯網為主軸，分為三個學習階段，學習階段一：進行物聯網基礎教學，帶領學生學習 Arduino 程式設計和電子元件；學習階段二：進行物聯網進階教學，讓學生透過每週的模組實作，學習更多感測器應用、網路通訊功能、雲端服務串接等，共有四個模組：「自動照明裝置」、「自動風扇開關」、「危害氣體偵測器」、「智慧門鎖」；學習階段三：進行「智慧家庭」物聯網專題實作，學生應用前兩階段所學得之知識與技能，整合各模組之功能，組裝房屋並設置模組，完成最終的專題作品。各階段之學習內容說明如表 1 所示。

表 1 學習內容說明

項目	學習內容
Arduino 入門	認識 Arduino、Arduino 開發環境介紹、程式語言、基礎電路、電子元件
程式設計入門	Arduino 控制結構、資料型態、變數、運算子、序列阜、條件式、迴圈、函式
自動照明裝置	訊號處理、紅外線感測器監控、LED
自動風扇開關	電壓調節、溫溼度感測器監控、風扇模組
危害氣體偵測	CO 感測器數據監控、蜂鳴器控制、Wi-Fi 無線通訊、ThingSpeak 平台串接
智慧門鎖	磁簧開關控制、伺服馬達控制、串接 IFTTT 服務、LINE 通知訊息發送
人工智慧	微軟 AI 認知服務（影像、語音、語言）
智慧家庭	雷切房屋組裝、模組設置與整合、多感測
解決方案	裝置監控

3.2. 教學策略

本研究將運算思維四大概念與 6E 模式的六步驟做對應，提出「CT-6E 模式」，每週之教學活動皆使用以此進行課程設計，圖 1 為教學策略執行內容圖，說明各步驟中教師教學和學生學習的方式。

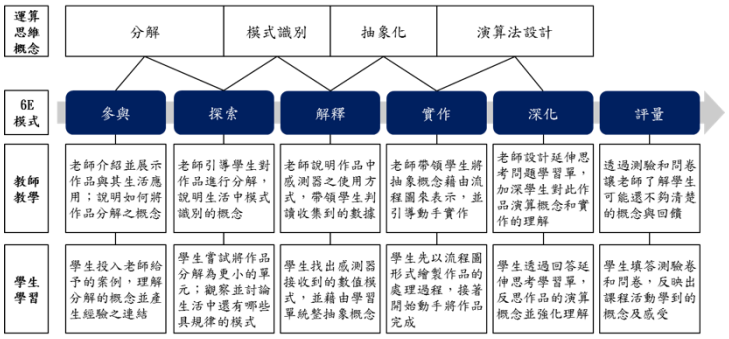


圖 1 CT-6E 模式執行內容圖

學習階段二，每個模組之教學內容皆對應運算思維四大概念，表 2 以「自動照明裝置」為例說明之。

表 2 教學內容與 CT 概念對照表

CT 概念	說明
分解	問題拆解為(1)偵測人的距離，判斷是否靠近；(2)自動開啟照明
模式識別	• 感測器會回傳一範圍區間之數值資料 • 經過換算可用以衡量受測物之距離 • 觀察並定義出人是否靠近的距離範圍
抽象化	• 使用紅外線感測器取得受測物之距離 • 使用 LED 燈作為照明裝置
演算法設計	• 將任務以程序表示，如：取得感測器數值→是否在某距離內→點亮燈數秒 • 畫流程圖並用程式實作出來

4. 研究方法

4.1. 研究架構

採用準實驗研究法，探討不同教學模式之物聯網教學活動，對於學生物聯網學習表現、運算思維及自我效能之影響。研究架構圖如圖 2 所示。

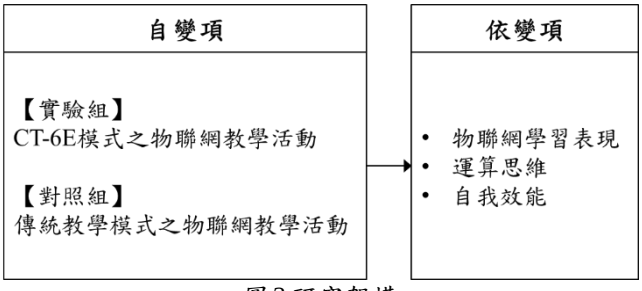


圖 2 研究架構

4.2. 研究對象

將以台北市某所高級中學一年級學生為實驗對象，實驗組、對照組各 2 個班，每個班 40 個人，共 160 人。

每個班學生 2 人為一組，共分 20 組參與實驗教學和專題實作。

4.3. 實驗設計與實施

實驗含前後測共 10 週，每週有 100 分鐘，第一週進行物聯網學習表現、運算思維和自我效能之前測，並說明上課規範與計分方式。第二週與第三週為學習第一階段，接受物聯網基礎教學，學習 Arduino 程式設計和電子元件。第四週至第七週為學習第二階段，接受物聯網進階教學，讓學生透過每週的作品實作，學習更多感測器應用、網路通訊功能、雲端服務串接等。第八週與第九週為學習第三階段，進入連續兩週之物聯網專題實作。第十週課程活動結束後，進行物聯網學習表現、運算思維和自我效能之後測。圖 3 為本研究之實驗設計流程說明。

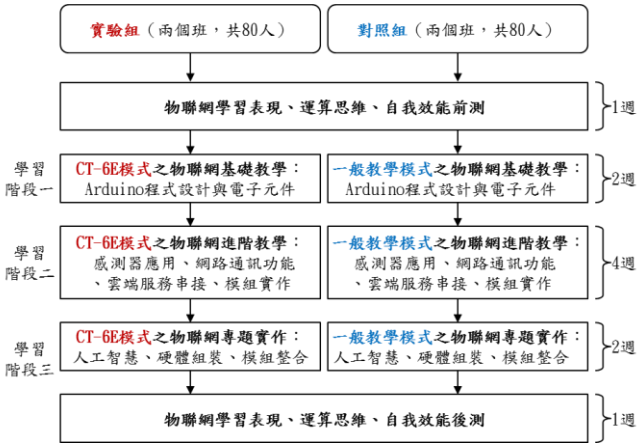


圖 3 實驗流程

4.4. 研究工具

4.4.1 物聯網學習表現測驗卷

本研究將自行編製物聯網學習表現測驗卷，內容包括物聯網、電子電路、感測器、程式設計等。將請兩位有教學經驗之資訊科技老師進行專家審查，達到內容效度標準；將於某高中進行預試，剔除內部一致性系數未達.5 之題項。

4.4.2 國際運算思維測驗

採用國際運算思維能力測驗（International Bebras Contest），挑選近五年題庫中，符合「分解」、「模式識別」、「抽象化」、「演算法設計」概念之題目進行編制。題目皆設計為情境式，讓學習者利用自己既有的知識及運算思維概念進行解題。

4.4.3 自我效能量表

採用 Tsai 等人（2019）發展之程式設計自我效能量表（Computer Programming Self-Efficacy Scale, CPSES）分為邏輯思維、演算法、偵錯、控制和合作等五個構面，共 16 題，以李克特六點尺度計分，整體之 Cronbach's α 為 0.96，各構面 α 值的範圍從.84 到.96，具有高度的內部一致性。

4.5. 預期成果

透過準實驗研究法來探討兩者對於高中生物聯網學習表現、運算思維和自我效能之影響。使用共變數分析，比較實驗組與對照組在物聯網學習表現、運算思維和自我效能之結果。並預期運用 CT-6E模式之實驗組，在物聯網學習表現、運算思維和自我效能上，都能顯著高於一般教學模式的對照組。

5. 參考文獻

國家發展委員會（2016）。**107-109 年重點產業人才供需調查及推估彙整報告**。臺北市：國家發展委員會。

教育部（2019）。**十二年國民基本教育課程綱要綜合型高級中等學校-科技領域**。台北：教育部。

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Journal of Educational Technology & Society, 19*(3), 47-57.

Bandura, A. (1997). *Self-efficacy: The exercise of control.* New York, NY: WH Freeman and Company。

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?. *Inroads, 2*(1), 48-54.

Barry, N. (2014). The ITEEA 6E Learning byDeSIGN™ Model. Technol. *Eng. Teach, 73*, 14-19.

Chen, J. C., Huang, Y., Lin, K. Y., Chang, Y. S., Lin, H. C., Lin, C. Y., & Hsiao, H. S. (2019). Developing a Hands-on Activity Using Virtual Reality to Help Students Learn by Doing. *Journal of Computer Assisted Learning*. doi: 10.1111/jcal.12389

Computer Science Teachers Association (CSTA). (2011). *CSTA K-12 computer science standards. The ACM K-12 Education Task Force.* Retrieved November 20, 2019, from https://www.csteachers.org/page/standards

Gandhi, H., & Varma, M. (2009). Strategic Content Learning Approach to Promote Self-regulated Learning in Mathematics. *Proceedings of epiSTEME, 3*, 119–124.

Google for Education. (2015). *Exploring Computational Thinking*. Retrieved November 18, 2019, from https://edu.google.com/resources/programs/exploring-computational-thinking/

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43.

Hsiao, H. S., Lin, Y. W., Lin, K. Y., Lin, C. Y., Chen, J. H., & Chen, J. C. (2019). Using Robot-based Practices to Develop an Activity that Incorporated the 6E Model to Improve Elementary School Students' Learning Performances. *Interactive Learning Environments*, 1-15.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to Learn and How to Teach Computational Thinking: Suggestions based on a Review of the Literature. *Computers & Education, 126*, 296-310.

Kortuem, G., Bandara, A. K., Smith, N., Richards, M., & Petre, M. (2012). *Educating the Internet-of-Things Generation. Computer, 46* (2), 53-61.

Lai, Y. H., Chen, S. Y., Lai, C. F., Chang, Y. C., & Su, Y. S. (2019). Study on Enhancing AIoT Computational Thinking Skills by Plot Image-based VR. *Interactive Learning Environments*, 1-14.

Mavroudi, A., Divitini, M., Gianni, F., Mora, S., & Kvittem, D. R. (2018). Designing IoT applications in Lower Secondary Schools. *Proceedings of 2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1120-1126.

Omale, G. (2018). *Gartner Identifies Top 10 Strategic IoT Technologies and Trends.* Retrieved November 7, 2018, from https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends

PricewaterhouseCoopers (PwC). (2019). *Talent Trends 2019: Upskilling for a Digital World*. Retrieved August 1, 2019, from https://www.pwc.tw/zh/publications/events-and-trends/cover-story/c334-cover.html

Barry, N. (2014). The ITEEA 6E Learning byDeSIGN™ Model. *Technol. Eng. Teach, 73*, 14-19.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy. *Education. Journal of Educational Computing Research, 56*(8), 1345-1360.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and STEM/STEAM Education

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

A Study on Influential Factors of Primary School Students' Computational Thinking in Interdisciplinary STEM Teaching

Pinghong ZHOU[1*], Yi ZHANG[2], Wei MO[3], Jue WANG[4]
[1,2,3]School of Educational Technology, Central China Normal University, China Wuhan
[4] School of teacher education, Huzhou University， Huzhou Zhejiang
phzhou@mail.ccnu.edu.cn, zhangyi@mail.ccnu.edud.cn, 22089463@qq.com, wangjue@zjhu.edu.cn

ABSTRACT
This study explores the relationship among learners' metacognition, learning motivation and engagement and Computational Thinking in interdisciplinary STEM teaching. A questionnaire survey was carried out in a primary school of Wuhan Economic Development Zone. 593 samples were collected from 6 classes in grade 3, 4 and 5 of the primary school. The results of structural equation model analysis showed that: (1) metacognition, learning motivation, engagement and computational thinking were significantly positively correlated; (2) the direct and indirect effects of Metacognition on Computational Thinking were significant. Indirect effect includes two paths: through the partial mediating role of learning participation and through the chain mediating role of learning motivation and engagement; (3) the direct effect of learning motivation on Computational Thinking is not significant, but it can affect Computational Thinking through the mediating role of engagement. This conclusion provides the relevant strategy reference for the cultivation of students' computing thinking ability.

KEYWORDS
computational thinking, metacognition, learning motivation, engagement

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

跨学科 STEM 教学中小学生计算思维影响因素研究

周平红[1*]，张屹[2]，莫尉[3]，王珏[4]
[1,2,3]华中师范大学教育信息技术学院，中国武汉
[4]湖州师范学院教师教育学院，浙江湖州
phzhou@mail.ccnu.edu.cn，zhangyi@mail.ccnu.edud.cn，22089463@qq.com，wangjue@zjhu.edu.cn

摘要
本研究探讨在跨学科STEM教学中学习者的元认知、学习动机、参与度与计算思维之间的关系。研究在武汉市经济开发区某小学开展问卷调查，收集了该小学三年级、四年级、五年级共计6个班593份样本，结构方程模型分析结果显示：（1）元认知、学习动机、参与度与计算思维两两之间均显著正相关；（2）元认知对计算思维影响的直接效应和间接效应均显著。间接效应为通过学习参与度的部分中介作用和通过学习动机与参与度的链式中介作用；（3）学习动机只能通过参与度的中介作用影响计算思维。这一结论为培养学生的计算思维能力提供了相关策略参考。

关键词
计算思维；元认知；学习动机；参与度

1. 问题的提出
计算思维（Computational Thinking，简称CT）是21世纪的关键技能之一。美国国际教育技术协会（ISTE）确定的21世纪学生标准包括解决问题、合作、创造力和批判思维等高级技能，计算思维直接与这些技能相关。Wing（2006）认为计算思维是利用计算机科学的基本概念来解决问题，设计系统并理解人类行为，与编程和数学思维有着密切的联系。此外，计算思维涉及问题解决过程中复杂和高阶的技能，是一种通过分解、抽象、泛化、算法设计、调试和迭代等技能思考和采取行动的方式，被认为有助于在各个领域发展知识和理解概念，具有发展问题解决技能的巨大潜能。由于计算思维的重要性，许多国家将计算思维纳入K12教育中，并且关注促进学生计算思维能力提升的教学方法和研究设计。为了提高学生的能力，将计算思维和STEAM课程相结合是恰当的选择，同时考虑这两个领域可能共享的大量主题材料，因为通过跨学科整合有助于提升学生的问题解决能力（ISTE，2016）。在计算思维教育中如何让学生参与有意义的学习以培养他们有用的思维技能和数字能力还需进一步探索。

为此，在面向计算思维培养的K12教育和培训过程中确定有效的影响因素非常重要。但是在跨学科整合的STEM课程中，对哪些因素显著影响学生计算思维的发展、计算思维与其他变量之间关系的相关研究较少。本研究从变量间链式中介关系的角度，探讨个体内部心理变量元认知、学习动机、参与度如何影响计算思维。即研究主要探讨经过跨学科整合的STEM课程教学后，检验学生在问题解决过程中计算思维技能与其学习动机、参与度、元认知等变量之间的关系。

2. 文献综述与研究假设
2.1. 文献综述
2.1.1. 计算思维的培养及其影响因素
计算思维目前没有统一的定义。最早Papert通过关注儿童在LOGO环境中进行编程开发的过程性思维提出了计算思维的想法。Wing认为CT不仅涉及编程，还具有使用计算机科学的基本概念来理解人类行为的技能。在2010年，她重新引入“计算思维”一词，认为计算思维是解决问题及其解决方案所涉及的思想过程，以便以一种可用信息处理代理有效执行的形式来表示解决方案（Wing, 2006）。Brennan和Resnick提出了一个框架，从CT概念、CT实践和CT观点三个维度对CT进行概念化（Brennan & Resnick, 2012）。

张屹等构建了以计算思维培养为核心，以计算机编程作为载体，跨学科STEM整合培养学生计算思维的理论框架。该理论框架分为学科内容层（STEM）、跨学科大概念层（即凝练抽象与具体、数量与比例、图式与模式、结构与功能、原因与结果等）和学习思维三层（张屹, 李幸, 黄静等, 2018）。

目前培养学生计算思维能力的课程有编程、数学、自然科学、社会科学和语言艺术，根据计算思维技能而不仅仅是程序设计或编程教学来培养学生的问题解决、抽象思维、程序思维及类似能力。Durak和Sarıtepeci对安卡拉的156名公立学校学生计算思维能力影响因素的研究表明，教育水平、数学成绩、对数学课程的态度和对科学课程的态度是影响计算思维的主要因素（Durak & Saritepeci, 2018）。也有相关研究结果表明在计算思维教育中对编程更感兴趣的学生会认为编程更有意义，具有更大的创作自我效能感和更大的编程自我效能感。此外，在面向计算思维技能培养的视频游戏中认知和态度对学生会有影响。以上相关研究显示，影响计算思维的因素既有认知层面的（态度、兴趣、自我效能感等），也有知识层面的因素，如成绩。

2.1.2. 面向问题解决的元认知与计算思维
元认知概念最早由美国儿童心理学家弗拉维尔提出，是指个体关于自己的认知过程、结果以及任何相关事物的知识，另一方面指个体对自己认知过程的主动监控、结果的调整以及对整个过程的协调。许多研究强调计算思维是一种认知过程，将其描述为解决问题的方法，强调元认知在计算思维过程中的作用，并且通过关注计算机信息的自动化来了解计算思维与其他思维方式的不同之处。Aho认为计算思维是解决问题的思维过程，其解决方案可以表示为计算步骤和算法（Aho,

2012）。从心理学的角度出发，形成问题的心理表征、计划和选择解决方案的适当策略、检查错误和调试、思考如何改进等组成了元认知的计划、评价和监控部分。

相关研究者强调元认知和计算思维的关系。 Resnick 提出，建构性的学习环境需要为学习者提供迭代设计解决方案并反思自己学习过程的机会，以促进学习计算思维技能（Resnick, 2007）。Yasemin Allsop 认为计算思维是一个认知过程，受元认知实践的监管，涉及一系列计算概念的应用，包括对学习行为的利用（Allsop, 2019）。牟连佳等立足计算思维的元认知视角，认为在整个计算思维的心理操作序列中，学习者有机会试用元认知监控信息特性、陈述性和程序性知识以及认知经验，以维持个体在各种情境下解决问题的动机（指计算思维）（牟连佳等，2015）。为此，学习者的元认知对其计算思维的发展具有一定的影响作用。

此外，一些研究者对元认知与学习动机的关系进行了探究。汪玲等借助于结构方程模型中的路径分析方法，得到元认知受动机变量的调节和制约，动机变量（如学业自我概念、考试焦虑、掌握定向、内部归因等）对元认知活动具有“供能”作用的结论（汪玲和郭德俊，2003）。Ghaleb 等人的研究表明，掌握目标的学生可能拥有掌握信息所需的高级元认知技能和策略；使用高级元认知最终会提高学习动机（Ghaleb，Ghaith & Akour, 2015）。为此，学习者的元认知与其学习动机会相互作用和相互促进。

2.1.3. 学习动机与计算思维

动机是使个人从事某种特定行为的内在力量，是激发、引导和维持行为的内部过程，是使学习者开始行动、维持行动，并决定其行动的方向（格雷德勒，2007）。具有强烈学习动机的学生会运用更高级的认知活动，学习和记忆更多的内容。一个相信完成一项任务具有更大影响力的人将会有更多的内在动力，并且更有可能为完成任务而付出努力。计算思维教育具有使学生从技术的使用者转换为技术的生产者的潜能。相关研究表明，培养学生计算思维的教学方法主要有游戏教学法、探究式学习、基于问题的学习和建构主义以及配对编程，需要创建基于任务的学习环境以增强动机和认知成果（Belland, Kim & Hannafin, 2013）。在这些教学方法下构建的高阶思维培养环境能促进学生进行信息处理、反思活动的计算实践。为此，学习动机能够正向影响计算思维能力水平。

2.1.4. 参与度与计算思维

参与度是学生参与学习活动过程中所表现出的行为强度（如付出的时间与努力程度）与情绪的质量（如享受或厌倦学习过程）。目前，被普遍认同的学习参与度包括认知参与、行为参与和情感参与。相关研究表明，学习动机是影响学习投入的关键因素，参与度来自学生的动机，内在动机可以预测参与度，学生的内在动机和学生的自我效能感高度相关，倾向参与的学生具有内在动机（Dunn & Kennedy, 2019）。参与度是自主学习成功的关键因素之一，K. Sharma 等认为协作和参与能在学生参与面向计算思维培养的编程活动中对学生的学习态度起有效的调节作用，该研究强调为儿童设计高度协作和参与的编码活动的重要性，并指出参与度可以调节参与意向与认知学习之间的关系（Sharma, Papavlasopoulou & Giannakos, 2019）。计算思维作为一种解决问题的学习方式，通过简化、嵌入转换或模拟等来重构和解决问题，学生积极且可持续的参与对于面向计算思维培养的教学来说非常重要。促进适应性和参与度，为所有年龄的学习者提供计算思维发展的内在动力是关键。

综上所述，国内外已有相关研究主要探讨计算思维培养过程中学生的元认知、学习动机、参与度的重要性与关联性，而对学生的元认知、学习动机和参与度如何以及在多大程度上影响计算思维水平的问题研究较少。本研究将建立计算思维能力培养的影响因素模型，并通过数据对此理论模型进行验证分析。

2.2. 研究假设

本研究旨在探究跨学科 STEM 课程学习中学习者的计算思维水平与学习者内在心理变量即学生的元认知、学习动机和参与度的关系，进而探讨影响学生计算思维能力的因素，研究假设模型如图 1 所示，参与度在元认知、学习动机和计算思维之间起着中介作用。

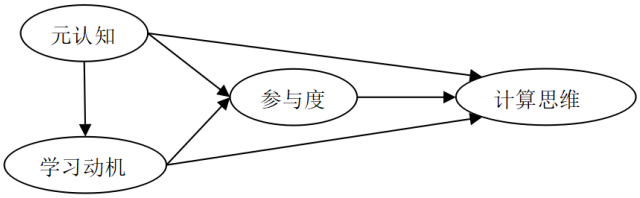


图 1 研究假设模型

研究模型中每条路径的研究假设为：

H1：在面向计算思维培养的教学中学生的元认知会正向影响其学习动机和计算思维；

H2：在面向计算思维培养的教学中学生的学习动机会正向影响其参与度；

H3：在面向计算思维培养的教学中学生的元认知会正向影响其计算思维；

H4：元认知会通过参与度影响计算思维，参与度是元认知和计算思维的中介变量；

H5：学习动机会通过参与度影响计算思维，参与度是学习动机和计算思维的中介变量；

H6：元认知会通过学习动机和参与度影响计算思维，学习动机和参与度在元认知和计算思维之间起着链式中介效应。

3. 研究设计

本研究是促进小学生计算思维培养研究项目的一部分。在该项目研究中强调跨学科整合的 STEM 课程设计及其对学生计算思维能力培养的重要性。在项目开始前期对教师进行基于设计的跨学科 STEM 教学培养学生计算

思维能力的相关培训，在学期结束后对学生的计算思维能力及其学习动机、元认知、参与度等进行后测。

3.1. 研究工具

本研究借鉴国内外已有的相关量表，通过改编量表、质性访谈等方法设计调查问卷。问卷共分五部分，包括四个子量表，分别为计算思维子量表、元认知子量表、参与度子量表、学习动机子量表。每个量表的题目以李克特五级量表形式进行设定（非常不同意=1，不同意=2，一般=3，同意=4，非常同意=5），得分越高表明学习者某一维度的水平越高。

3.1.1. 计算思维量表

本研究参考 Korkmaz2017 提出的计算思维量表（Korkmaz, Çakir & Özden, 2017），在此基础上进行改编，形成包括创造力、算法思维、批判性思维、问题解决和合作能力等五个维度的计算思维量表。其中创造力维度包括 8 个指标、算法思维维度包括 5 个指标、批判性思维包括 5 个指标、问题解决能力包括 6 个指标、合作能力包括 8 个指标。在本研究中该量表的一致性系数 Coronhach'α 为 0.912。

3.1.2. 元认知量表

本研究中元认知量表采用的是 Schrawhe 和 Dennison 编写的元认知水平问卷（Schraw & Dennison, 1994），包括认知知识和认知调节两个维度，其中认知知识包括陈述性知识、程序性知识和条件性知识，认知调节包括计划、信息管理、监控、调节和评价等。本研究根据需要编制了五个题目，对学生的认知知识和认知调节进行测量，其中认知知识 2 个题目，认知调节 3 个题目。在本研究中该量表的一致性系数 Coronhach'α 为 0.858。

3.1.3. 学习动机量表

本研究中学习动机量表参考期望——价值动机理论而设计，该理论认为个体对当前学习任务价值的知觉程度，是学习投入和学习成就的有力预测因素（Wigfield & Eccles, 2000）。问卷主要从能力信念、对成功的期望和任务价值（包括当前学习任务的实用性、重要性与趣味性）等维度来考察学习者的动机情况。本研究根据需要编制了 7 个题目，其中任务价值 3 个题目，能力信念 2 个题目，对成功的期望 2 个题目，在本研究中该量表的一致性系数 Coronhach'α 为 0. 780。

3.1.4. 参与度量表

本研究中参与度测量采用的是 Barbara A. Greene、Miller 等人（Greene, 2015）编制的学生认知参与测量问卷以及 Reinhard Pekrun 等人（Pekrun, Goetz & Frenzel, 2011）编制的情感参与测量问卷。问卷主要从认知参与、情感参与两个维度来测量学习者在课堂中认知和情感的投入程度。该问卷共包含 7 个题目，其中认知参与 3 个题目，情感参与 4 个题目，问卷采用李克特量表进行自我评定（1=非常不同意，5=非常同意），得分越高表明学习者在课堂学习中的参与效果越好。在本研究中该量表的一致性系数 Coronhach'α 为 0.799。

3.2. 调研样本

采用分层随机抽样法，在湖北省武汉市经济开发区某小学的三年级、四年级和五年级共计六个班进行问卷调查，前测回收问卷 773 份，后测回收问卷 720，剔除无效样本，进行前后测配对后获得的有效配对样本为 593 份。其中男生 280 名（47.2%），女生 313 名（52.8%）。三年级学生 189 名（31.9%），四年级学生 211 名（35.6%），五年级学生 193 名（32.5%）。本研究被试所在各个年级都接受面向计算思维培养的跨学科 STEM 课程教学，包括基于单学科的 STEM 课程、基于统整课程和社团课的跨学科 STEM 课程教学。

3.3. 数据处理与分析

研究采用有中介的结构方程模型对数据进行建模和统计分析。首先，利用 SPSS22.0 工具对问卷数据进行预处理和基本描述性统计分析，删除一些不合理的数据。然后，利用 AMOS21.0 工具构建结构模型，对测量模型部分进行验证性因素分析，对结构模型部分进行路径分析。最后，利用 Bootstrap 法对模型的中介效果进行检验和分析。

4. 研究结果与分析

4.1. 测量模型的信度和效度检验

本研究包含元认知、学习动机、参与度和计算思维四个潜在变量，每个潜在变量下设有相应的外显测量指标。为了保证模型的有效性，需要先对模型的信度和效度进行检验。信度检验采用克隆巴赫信度系数（Cronbach's α）和组合信度（CR 值）对测量模型进行信度分析。效度检验采用平均方差萃取量（AVE 值）检验收敛效度以及 AVE 的平方根检验区分效度，AVE 体现的是潜变量解释测量指标变异量的程度。

通过 SPSS22.0 软件计算问卷的整体一致性信度系数，问卷整体 Cronbach's α 系数为 0.946，大于 0.9 表明问卷整体一致性较好。通过 AMOS21.0 软件对模型参数进行估计，计算出各测量模型的因子负荷量，在此基础上进一步计算组合信度和区分效度，具体分析结果如表 1 所示。

表 1 中的数据分析结果表明，本模型中各潜变量的组合信度值在 0.752-0.913 之间（CR 值>0.7），表明本模型的信度达到要求，模型的内在质量理想。从观察变量的标准化因子负荷值可知，所有观察指标的估计值都在 0.539-0.891 之间，所有显变量在潜变量上的因子载荷 t 值从 13.036 到 25.929 之间，达到显著性水平（Z 值大于 1.96），且无负的误差变异数，表明模型符合拟合评价标准，平均方差萃取量（AVE）值在 0.505-0.678 之间（均大于 0.5），表明测量模型具有良好的收敛效度。

此外，AVE 的平方根值若大于潜在变量间的相关系数，则表示各潜在变量之间具有区别效度。如表 2 所示，元认知与学习动机正相关（r=0.392，p<0.01），元认知与参与度正相关（r=0.345，p<0.01），元认知与计算思维正相关（r=0.562，p<0.01），参与度与学习动机正相关（r=0.555，p<0.01），计算思维与学习动机正

相关（r=0.650，p<0.01），计算思维与参与度正相关（r=0.667，p<0.01）。各潜在变量的 AVE 的平方根值（即斜对角线的值）大于大多数潜在变量间的相关系数，说明测量模型具有较好的区别效度。

表 1 信度和效度分析表

潜在变量	观察变量	标准化因子载荷	Z 值（Z-value）	Cronbach's a 值	组合信度（CR 值）	平均方差萃取量（AVE 值）
元认知	MC1	0.748	——	0.858	0.848	0.528
	MC2	0.753	18.272***			
	MC3	0.713	17.201***			
	MC4	0.701	16.976***			
	MC5	0.717	17.297***			
学习动机	LM1	0.539	——	0.780	0.802	0.583
	LM2	0.844	13.036***			
	LM3	0.865	13.056***			
参与度	LE1	0.789	——	0.799	0.752	0.505
	LE2	0.699	17.848***			
	LE3	0.635	15.923***			
计算思维	CR	0.787	——	0.912	0.913	0.678
	AT	0.713	18.984***			
	CT	0.840	23.717***			
	PS	0.891	25.929***			
	CO	0.874	21.503***			

注：N=593，*P<0.05，**P<0.01，***P<0.001，组合信度 $CR=\frac{(\sum\lambda)^2}{[(\sum\lambda)^2+\sum(\theta)]}$，$\lambda$ 为指标变量在潜变量上的标准化参数估计值，θ 为观察变量的误差变异量。

表 2 各变量均值、标准差、相关系数及区别效度分析

变量	M	SD	1	2	3	4
1.元认知	4.186	0.696	**0.727**			
2.学习动机	4.287	0.579	0.392**	**0.764**		
3.参与度	4.372	0.700	0.369**	0.632**	**0.711**	
4.计算思维	4.409	0.565	0.463**	0.629**	0.722**	**0.823**

注：N=593，*P<0.05，**P<0.01，***P<0.001，斜对角线粗体字为 AVE 的平方根值。

4.2. 模型整体拟合效果与假设检验

研究采用 AMOS 软件中的极大似然法进行模型的拟合，得到的拟合度指标数据显示：卡方值/自由度＝2.984；渐进残差均方和平方根 RMSEA＝0.057；标准化残差均方和平方根 SRMR＝0.037；良适性适配指标 GFI＝0.947；基准线比较指标 IFI＝0.972、TLI＝0.963、CFI＝0.972。上述各指标表明，研究的模型为可接受模型。为了验证研究假设是否成立，模型拟合过程中，根据修正指数对模型进行了适当修正，修正后的模型路径如图 2 所示。图中显示的是路径系数显著的路径，即研究假设成立的路径，而不显著的路径表明研究假设不成立，所以在图中没有显示。

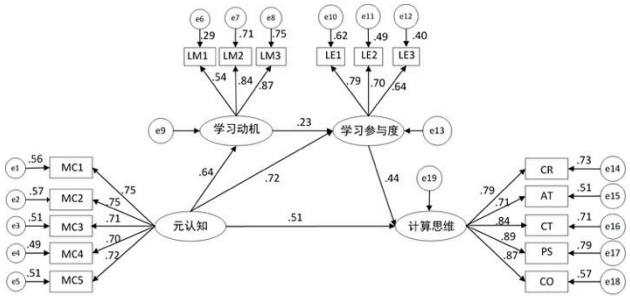


图 2 学习者元认知、学习动机、参与度对计算思维的影响

图 2 所示的结构模型中变量关系显示，在面向计算思维培养的 STEM 课程中，学习者的元认知对学习者的学习动机有显著影响（β=0.636，P<0.001）；学习者的元认知（β=0.717，P<0.001）、学习动机（β=0.234，P<0.001）对学习者的参与度均有显著影响。其次，学习者的元认知（β=0.514，P<0.001）、参与度（β=0.437，P<0.001）对其计算思维水平有显著影响。而学习者的学习动机对计算思维影响的路径系数因参数检验的显著性概率值不显著（P>0.05）被移除。

4.3. 中介效应分析

研究参考温忠麟等研究者提出的中介效应检验方法，采用 Bootstrap 法进行中介效应检验。该方法根据标准误的理论概念，将样本容量很大的样本当作总体，进行有放回抽样（抽样次数可以自己定），从而得到更为准确的标准误。本研究使用 AMOS 进行 Bootstrap 的中介效应检验，将样本量设置为 1000，选择 95%的置信区间，观察有偏置信区间（Bias-corrected percentile method）估计的上限和下限值是否包含 0，如果包含 0，则中介效应不显著，如果不包含 0，则中介效应显著（温忠麟和叶宝娟, 2014）。中介效应分析显示各中介路径的上限和下限值及效应值如表 3 所示。

表 3 Bootstrp 法估计的中介效应及效应值

路径	Bias-corrected 95%置信区间		间接效应值	总间接效应	直接效应	总效应
	下限	上限				
元认知—学习参与度—计算思维	0.142	0.375	0.313	0.378	0.514	0.892
元认知—学习动机—学习参与度—计算思维	0.053	0.182	0.065			
学习动机—学习参与度—计算思维	0.055	0.266	0.102	0.102	0	0.102

中介效应的分析结果表明（如表 4 所示），元认知可直接作用于计算思维，直接效应为 0.514，占总效应的 57.6%；学习参与度和学习动机在元认知与计算思维之间起到了部分中介作用，中介效应值 0.378，占总效应的 42.4%。具体来看，中介效应由两条路径产生的间接效应组成：通过元认知——学习参与度——计算思维产生的间接效应 1（0.313），通过元认知——学习动机——学习参与度——计算思维产生的间接效应 2（0.065）。间接效应 1 的 Bootstrap95%有偏置信区间[0.142，0.375]和间接效应 2 的 Bootstrap95%有偏置信区间[0.142，0.375]均不包含 0，表明两个中介效应显著，间接效应分别占总效应的 35.1%和 7.3%。

此外，学习动机影响计算思维过程中，由于学习动机对计算思维的路径系数未达到显著性水平，所以学习动机未能直接作用于计算思维，学习参与度在学习动机和计算思维之间起到了完全中介效应，其间接效应值为 0.102，表明学习动机对计算思维的影响完全通过中介变量学习参与度起作用，学习动机对计算思维没有直接影响。

5. 讨论与启示

5.1. 讨论

本研究探讨STEM教育中个体内部心理变量元认知、学习动机、参与度与计算思维之间的相互影响关系。研究发现，元认知对学生计算思维能力影响的直接效应和间接效应都显著。其中的间接效应通过两条中介作用途径产生：第一，通过学习参与度的独立作用；第二，通过学习动机和参与度的共同作用。其次，学习动机对学生计算思维能力培养的直接效应不显著，间接效应显著，间接效应通过参与度的独立作用产生。

5.1.1. 学生的元认知与计算思维的关系

根据本研究的分析结果看，学生的元认知对计算思维能力有正向显著的直接影响。由于元认知涉及个体在解决问题过程中动机、策略和目标等的自我监控，这与计算思维的本质内涵即利用计算机来解决问题的思维过程联系尤为密切，在计算机科学中大多数算法选择问题代表了一个经典的元认知任务。学习思维方式将提高学生的创造性学习技能，并提高他们解决问题和抽象等技能的能力。为此，在本研究构建的模型中元认知成为预测计算思维水平的重要因素，有效提高学习者的元认知技能将能使他们更成功的掌握计算技能。

5.1.2. 参与度在元认知和计算思维之间的中介效应分析

本研究结果表明，参与度作为学习者主动学习的重要特征，对学生计算思维能力有正向显著的直接影响，并且是元认知影响计算思维的重要途径，其中介效应达到 35.1%，这意味着元认知对计算思维的影响，有35.1%是通过提高学生的参与度来达成的。在面向计算思维能力培养的STEM课程中学生积极、可持续地参与活动和学习过程非常重要。尤其是在融入编程等活动的课程中参与度是一个不可忽视的变量，当学生面对复杂的问题解决产生负面情绪时时，需要采用恰当的元认知策略激发学生的参与度，促进学生进行有效的协作、问题解决和批判性思维，达到培养计算思维的能力。

5.1.3. 学习动机和参与度在元认知和计算思维之间的链式中介效应分析

学习者的主动学习除了体现在参与度上，还体现在学习动机上。研究结果发现学习动机不能直接影响计算思维，但是能够通过学生的参与度对计算思维产生正向的显著影响。此外，学习动机也受元认知的正向影响，并且学习动机和参与度在元认知与计算思维之间形成了链式中介效应。这表明学习者元认知技能的掌握能在学习动机上起着积极的促进作用，即促进学习者对任务价值的理解和期望、提升学习者的能力信念，进而表现出积极的参与，并促进计算思维能力的提升。

5.2. 启示

本研究发现在跨学科STEM教学中，学习者的元认知、学习动机和参与度等内部心理变量对计算思维能力的提升有正向显著的影响。学习者对认知和元认知的监控、调节，加强学生对计算概念的理解、反思，培养学生在问题解决过程中的元认知技能，能促进动机状态的调节，进而带动学生的认知参与和情感参与，提升学生的计算思维能力。

本研究也存在某些局限性，需要在以后的研究中加以改进，本研究仅考虑的学习者内部心理变量对计算思维能力的影响，未考虑学习者的性别、年龄、先前知识经验对计算思维能力的影响，另外在计算思维能力培养中相较于传统的教学，基于设计的STEM教学更有自己的特色，不同干预策略的影响效果还需进一步研究。

6. ACKNOWLEDGMENT

本研究受 2018 国家自然科学基金项目“促进小学生计算思维培养的跨学科 STEM+C 教学理论与实证研究（71874066）”资助。

7. 参考文献

张屹, 李幸, 黄静等（2018）。基于设计的跨学科 STEM 教学对小学生跨学科学习态度的影响研究。**中国电化教育**，(7)，81-89。

牟连佳、李丕贤和邵洪艳（2015）。计算思维与巴斯德象限——计算思维融入信息技术教育的研究框架。**科技创新导报**，(25)，20-23。

汪玲和郭德俊（2003）。元认知与学习动机关系的研究。**心理科学，26**(5)，829-833。

格雷德勒（2007）。**学习与教学: 从理论到实践**。中国轻工业出版社。

温忠麟和叶宝娟（2014）。中介效应分析: 方法和模型发展。**心理科学进展，22**(5)，731-745。

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal, 55*(7), 832-835.

Allsop, Y. (2019). Assessing Computational Thinking Process Using a Multiple Evaluation Approach. *International Journal of Child-computer Interaction, 19*, 30-55.

Belland, B. R., Kim, C., & Hannafin, M. J. (2013). A Framework for Designing Scaffolds that Improve Motivation and Cognition. *Educational Psychologist, 48*(4), 243–270.

Brennan, K., & Resnick, M. (2012, April). 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, 1*, 25.

Dunn, T. J., & Kennedy, M. (2019). Technology Enhanced Learning in Higher Education; Motivations, Engagement and Academic Achievement. *Computers & Education, 137*, 104-113.

Durak, H. Y., & Saritepeci, M. (2018). Analysis of the Relation between Computational Thinking Skills and Various Variables with the Structural Equation Model. *Computers & Education, 116*, 191-202.

Ghaleb, A. B., Ghaith, S., & Akour, M. (2015). Self-efficacy, Achievement Goals, and Metacognition as Predicators of

Academic Motivation. *Procedia-Social and Behavioral Sciences, 191*, 2068-2073.

Greene, B. A. (2015). Measuring Cognitive Engagement with Self-report Scales: Reflections from over 20 Years of Research. *Educational Psychologist, 50*(1), 14-30.

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.

Pekrun, R., Goetz, T., Frenzel, A. C., Barchfeld, P., & Perry, R. P. (2011). Measuring Emotions in Students' Learning and Performance: The Achievement Emotions Questionnaire (AEQ). *Contemporary educational psychology, 36*(1), 36-48.

Resnick, M. (2007, June). All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. *Proceedings of the 6th ACM SIGCHI conference on Creativity & cognition*. ACM, 1-6.

Schraw, G., & Dennison, R. S. (1994). Assessing Metacognitive Awareness. *Contemporary educational psychology, 19*(4), 460-475.

Sharma, K., Papavlasopoulou, S., & Giannakos, M. (2019). Coding Games and Robots to Enhance Computational Thinking: How Collaboration and Engagement Moderate Children's Attitudes?. *International Journal of Child-Computer Interaction.*

Wigfield, A., & Eccles, J. S. (2000). Expectancy–value Theory of Achievement Motivation. *Contemporary educational psychology, 25*(1), 68-81.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33–35.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Confronting Frame Alignment in CT Infused STEM Classrooms

Connor BAIN[1*], Sugat DABHOLKAR[2], Uri WILENSKY[3]
[1,2,3] Northwestern University, United States
connorbain@u.northwestern.edu, sugat@u.northwestern.edu, uri@northwestern.edu

ABSTRACT
While the Next Generation Science Standards (NGSS) have presented computational thinking (CT) as an integral part of scientific inquiry, little work has been done to explicitly enable this connection in classrooms. We report on the efforts of one such design-based implementation research project which, with participation from local teachers, has been implementing CT infused STEM units in biology and chemistry classrooms. Using teacher reflections facilitated by an external evaluator, research field notes, and interviews, we identify possible issues of frame alignment in our implementations–that CT practices, particularly using computational models, were valued but would not enable students to gain a deeper understanding of scientific content. We then use this analysis and Schulman's definition of teacher case knowledge to design a new element of the project that aims to enable teachers to promote collaborative scientific practice using computational models in the classroom that we call Lesson 0. We conclude with the discussion of a pilot implementation of this new lesson.

KEYWORDS
computational thinking, STEM education, teacher learning, computational modeling

1. INTRODUCTION
For many years, Computational Thinking (CT) practices have tended to only be featured in standalone computer science (CS) courses, resulting in unequal access for students from historically underrepresented groups in CS, such as women and racial minorities (Margolis & Fisher, 2003). However, in our increasingly computational world, CT has become a necessary and integral part of nearly every discipline, particularly STEM disciplines (Weintrop et al., 2015). In recent years, the Next Generation Science Standards (NGSS) have made clear that using computational thinking (CT) is a cornerstone of modern science education (Quinn et al., 2012; Wilensky, Brady & Horn, 2014). By embedding CT practices into high school STEM classrooms like biology, chemistry and physics, we can simultaneously improve access to CT for all students, particularly those underrepresented in CS, while also providing a more authentic STEM experience for students in these classes.

This work is part of a research practice partnership between a Midwestern U.S. research university and a network of urban high schools in a large Midwestern U.S. city. In this paper we analyze and discuss the experiences of 6 teachers who taught one of our CT-embedded curricula during the academic year in the 2nd iteration of a design-based implementation research (DBIR) project, where research and practice are collaborative, iterative, and systematically analyzed (Fishman et al., 2013). We identify shortcomings of our previous curricular design and professional development program that may have caused an issue in frame alignment between scientific inquiry and CT. We then propose a new introductory lesson to our curricula which attempts to address these differences by framing CT as an authentic part of scientific inquiry.

2. THEORETICAL FRAMEWORK
The character of CT practices in the science disciplines is not yet well understood, nor is how to create curriculum and assessments that develop and measure these practices (Grover & Pea, 2013). To address this gap, our group has explicitly characterized core CT practices through a taxonomy of CT practices in STEM (Weintrop et al., 2016). The taxonomy consists of practices related to *Data*, *Modeling and Simulation*, *Computational Problem-Solving*, and *Systems Thinking*. We translated our taxonomy into a set of learning objectives and used these to guide the development of the two CT science curricular units, one biology and one chemistry, used in this study. Our curricular approach, which aligns with that of the NGSS, emphasizes figuring out core ideas through engaging in CT practices, rather than treating the dimensions separately (NRC, 2012).

In this manner, we see *frame alignment* as one of the major roadblocks to integrating CT into STEM classes (Farrell et al., 2018). Frame alignment refers to "the linking of two ideologically congruent but structurally unconnected frames regarding a particular issue or problems" (Benford & Snow, 2000, p. 624). While NGSS embeds CT as one of its core practices, competing frames of promoting scientific discourse in the classroom, integrating CS for all ideas, and even simply encouraging student agency in using CT for inquiry can all be vying for precedence in a teacher's sensemaking of new curricula.

3. METHODS
As part of the second iteration of the DBIR project, 6 teachers, 2 biology and 4 chemistry across 2 high schools, implemented one of our two week (10 class period) curricular units during the 2016-2017 school year. The two schools (one urban and one suburban) were all located near a large Midwestern U.S. city. Each of the teachers had at least five years of experience in their respective subject. Prior to their implementation, each of the six teachers participated in a professional development program which defined CT practices in STEM, familiarized the teachers with the curricular units they would implement through selective enactment, and allowed teachers to review and redesign the curricula with edits and tweaks based on their particular classroom needs.

3.1. CT Science Curricular Units
Both the chemistry and biology curricular units were explicitly designed to teach traditional subject matter

content through the enactment of CT practices. The units focused on helping students develop practices for Modeling and Simulation through exploring NetLogo (Wilensky, 1999b) models. NetLogo models were chosen because the agent-based representations in this modeling environment make complex systems phenomena (like population dynamics in ecosystems), more accessible (Wilensky, 2001). The chemistry unit covered the basics of the Ideal Gas Laws through exploring how micro-level particle interactions give rise to the macro-level effects like pressure and temperature (Wilensky, 1999a). The biology unit focused on the principles of ecosystems and evolution with students designing and interacting with models of competition between species to discover how ecosystems reach equilibrium.

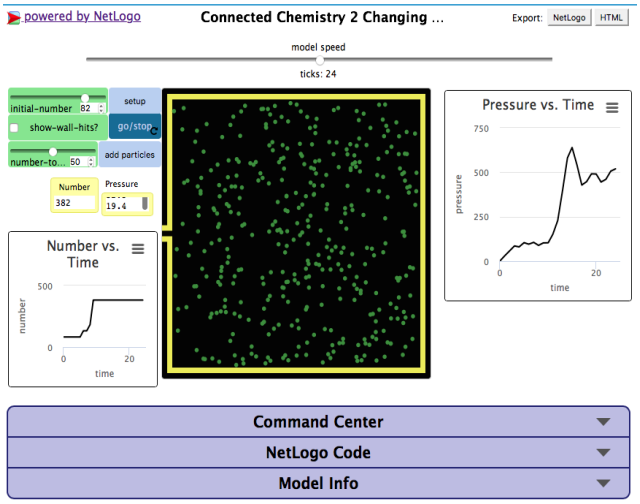


Figure 1. A chemistry model of gas particles colliding with the walls of a box which gives rise to the emergent phenomenon known as pressure (link blinded for review).

In both units, students explore the relationship between micro elements of the models and how they give rise to system level effects. Students observe trends within their data, use models to make and test predictions, and follow the steps of scientific inquiry in order to construct a deeper understanding of these phenomena (Wilensky & Reisman, 2006). These units are intentionally designed so that students engage in CT practices as part of an authentic scientific inquiry experience (NRC, 2012). The units are presented in the form of guiding questions, which encourage students to use either their prior knowledge or the exploration of a computational model to engage with the curricular content.

3.2. Data Collection

Data collection took place across twenty-two classes amongst our 6 teachers. Class periods were videotaped resulting in around 118 hours of video data. In addition, at least one researcher attended each class period and recorded written field notes. Because the curricula were hosted on our website, all student responses were recorded digitally). Finally, the teachers participated in interviews with an external evaluator about their experience with the professional development program and curricular implementation. For this paper, we use these teacher reflections and field notes to discuss frame alignment issues and motivate our new design efforts to mitigate those.

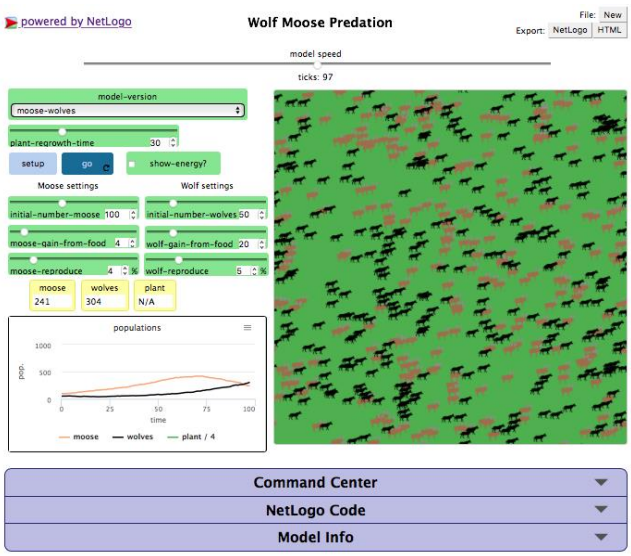


Figure 2. A biology model which allows students to manipulate behaviors of wolf and moose and reason about their emergent population dynamics (link blinded for review).

4. RESULTS

We had theorized that the students would emergently collaborate using this shared curriculum and computational models, with the teacher acting as a facilitator and modeling classroom talk (McNeill & Pimentel, 2010). In this manner, students would be participating in teams for scientific discovery–to discover the core ideas of gas laws and ecosystem stability. Computational models have been shown to be fruitful for this sort of classroom-level knowledge building (Wilkerson et al, 2007). In addition, in a pre-survey, 484 of 526 student participants agreed with the statement "People who have careers in science or computing need to work well in teams." In essence, we expected that students would use the computational models of anchoring phenomena for classroom talk and construct knowledge at the classroom level.

While we did see episodes of students debating computational methodologies in order to solve problems, we rarely saw classroom-level discussions of using a computational model for scientific inquiry. Some teachers facilitated classroom discussions at the end of each period on the "takeaways" (i.e. "organisms can compete indirectly if they are sharing a finite resource") for the day–an activity they classified as "usual practice" in their classrooms. Although these takeaways served as fruitful points of classroom discussion, none of the teachers explicitly talked about CT practices in these wrap-up discussions. In fact, one chemistry teacher Veronica saw the CT and chemistry content in direct conflict with each other.

I felt that, if the purpose is for them to see CT within content, yeah, but content was—I don't think it was as cohesive. Like the idea [was supposed to be], "Okay, so we're gonna teach gas laws and incorporate CT." It was more, "We're using that law to teach you computational—to teach you how to—to show you how models work."

Even the most experienced teacher in our study Ulyana, who was the head of the biology department at her school,

admitted that her number one concern during her implementation was to teach her students the biology content of the unit. Francine, another chemistry teacher made a similar comment,

I like the use of models in the classroom... I would have liked to see more of them walking away with more of the typical expectations for gas laws that you would expect students to get in those kind of conversations, but I like the use of models and the learning that they had with the models.

As such, Francine followed up the two-week implementation with a lecture-based repeat of gas laws to each of her classes. Her interview suggested she saw the models as a way of *reinforcing* a concept rather than an *introductory* or *exploratory* instrument. While each of our teachers saw the need to have CT embedded in the classroom, there was no indication that they saw our curricular approach, emphasizing figuring out core ideas through engaging in CT practices, as aligned with their content-specific goals.

While we believed our curricular design would help teachers elicit student thinking about both content and CT, from these results, we see that there was a significant discrepancy between how our team and how the teachers/students saw the alignment between content and the computational models and activities. We became interested in how to address this lack of frame alignment and whether we could design an introductory lesson that would provide a frame from which all the goals could be seen as aligned. In the rest of the paper, we describe our proposed solution.

5. PROPOSED SOLUTION AND PILOT OUTCOMES

Our analysis of teacher reflections revealed that the lack of clarity about connections between content and CT to the students and teachers may have led to the lack of collaboration and discussion related to CT practices and scientific inquiry. To use Schulman's (1986, pg. 11) term, we had provided teachers with a small amount of case knowledge–a *parable* which conveyed CT practices as the norm of the scientific community–without providing the associated *prototype* and *precedent* (1986). We used this framing from Schulman to design a new preparatory element for each of the curriculum we call *Lesson 0: How to Learn with Computational Models* (see it here: link removed for blinding). The lesson is meant to be used by both teachers and students as a sort of rehearsal of learning with computational models in order to get ready for the more discipline specific content coming later in each curriculum.

New science standards and reforms articulate a commitment to greater student agency with a disciplinary focus: that students should take on increased responsibilities for deciding what to figure out in science classrooms and how (Berland et al., 2016). In our curricular implementations, these frames seemed to conflict with the frame of CT as a way of scientific inquiry. As such, *Lesson 0* is designed with three main principles: 1. Scaffold students into discussions of how scientists use models; 2. Engage students with computational models as a method of scientific experimentation; 3. Demonstrate how to develop new understandings of using a computational model.

The lesson centers on a computational model of a forest fire and is divided into four sections meant to make explicit the ways in which computational models can be used to explore scientific concepts and engage in scientific inquiry practices. It was designed to scaffold teacher and student sensemaking with Schulman's (1986) three types of case knowledge in mind. In Step 1 (Using models to learn science), we make explicit the *precedent* that scientists use models, and specifically computational models, as methods of inquiry. In Step 2 (A not-so-sneak peek into the code), we encourage classroom-level discussion of debugging as a *parable*, establishing discussions about the code behind computational models are a valued norm of a CT classroom. In Step 3 (Systematically investigating the spread of wildfire), we present an implementation of a *prototype* of scientific inquiry, where students make hypotheses, design computational experiments, and draw conclusions based on the computational models. Finally, in Step 4 (Constructing knowledge by engaging in scientific inquiry practices), we further enforce the *parable* of the classroom as an arena for knowledge construction through discussion of both experimental conclusions as well as computational model design.

We implemented this new lesson with a group 8 High School science teachers at a Computational Thinking in STEM workshop hosted at a large Midwestern U.S. University. The second author served as the instructor, taking on the role as teacher educator. Teachers were asked to "play-as" students with the teacher educator serving as the teacher with the goal of the teachers entering into a participatory relationship with the lesson.

Ulyana, the same teacher from the prior iteration of the study, was one of the participants in this workshop. In addition to participating in the lesson as a student during the workshop, she also implemented the very same lesson in her classroom as the very first lesson of her biology unit. When asked about her experiences teaching the unit in this new iteration, Ulyana reflected upon her new understanding of what it meant to use computational models in the learning process:

So...in my head, my models were always the ones I did with very physical models. I never thought about using computational models until I met you guys. And those are even more important, because they can then use those computational models. That it can be seamless that you can take the concepts that you're already going to teach and put them into this model...and show the kids the value of computational models. Yeah, I mean, they were I felt like they what I learned is that they were [doing] what a real scientist would do in collecting the data.

In addition to seeing students participate in the practice of real science, Ulyana singled out how framing debugging and code inspection as an expected classroom practice, as is done in Lesson 0, allowed students to interact with models in a deep way:

…we are going into the code and fixing any problem there was so yeah, the kids, I can see that you could put a bug in,

and the kids can fix it. And sometimes there were bugs in accidentally, and we still had to fix so and that it isn't like the end of the world is a win. It's just a code. You fix the code. So nothing was ever really broken.

She also remarked how her students performed well on her typical AP-style assessments after completing the CT curricula: "So, they not only learned how to use a computational model, they learned the content I needed for their AP test." In short, Ulyana saw the computational models as opportunities for students to engage simultaneously in both science and CT. In the coming months, additional teachers will be implementing a similar curricular structure featuring Lesson 0 as the beginning of a CT infused STEM unit. We hope to continue to analyze student and teacher data to further learn how we can refine Lesson 0 to support CT as a normal classroom practice.

6. CONCLUSIONS

In this paper, we presented an analysis of data from an iteration of DBIR project that suggested that frame alignment was an obstacle in our goal of allowing students to use computational models to discover core disciplinary content ideas. We then presented a modification to our curricula: a prepended lesson to help both teachers and students better understand how computational models might serve as tools (and objects) of scientific inquiry. In order to assist teachers in integrating CT into STEM classrooms, we see a need to provide explicit prototypes, precedent, and parables in order to help teachers align the seemingly competing frames of teaching expected content, scientific inquiry practices, and computational thinking. We see Lesson 0 as one possible method of allowing both teachers and students to make sense of how these frames align in service of a new form of scientific learning.

7. REFERENCES

Benford, R. D., & Snow, D. A. (2000). Framing Processes and Social Movements: An Overview and Assessment. *Annual review of sociology*, *26*(1), 611-639.

Berland, L. K., Schwarz, C. V., Krist, C., Kenyon, L., Lo, A. S., & Reiser, B. J. (2016). Epistemologies in Practice: Making Scientific Practices Meaningful for Students. *Journal of Research in Science Teaching*, *53*(7), 1082-1112.

Farrell, C. C., Davidson, K. L., Repko-Erwin, M., Penuel, W. R., Hill, H. C., & Herlihy, C. (2018). Goals and Challenges of Research-Practice Partnerships for Improvement Efforts.

Fishman, B. J., Penuel, W. R., Allen, A. R., Cheng, B. H., & Sabelli, N. O. R. A. (2013). Design-based Implementation Research: An Emerging Model for Transforming the Relationship of Research and Practice. *National society for the study of education*, *112*(2), 136-156.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38-43.

Margolis, J., & Fisher, A. (2003). *Unlocking the clubhouse: Women in computing*. MIT press.

McNeill, K. L., & Pimentel, D. S. (2010). Scientific Discourse in Three Urban Classrooms: The Role of the Teacher in Engaging High School Students in Argumentation. *Science Education*, *94*(2), 203-229.

National Research Council. (2012). *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. Washington, DC: The National Academies Press.

Schweingruber, H., Keller, T., & Quinn, H. (2012). A framework for K-12 science education: Practices, crosscutting concepts, and core ideas. *Tech. Rep.*

Shulman, L. S. (1986). Those Who Understand: Knowledge Growth in Teaching. *Educational researcher*, *15*(2), 4-14.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147.

Wilensky, U. (1999a). GasLab—An extensible modeling toolkit for connecting micro-and macro-properties of gases. In *Modeling and simulation in science and mathematics education*. New York, NY: Springer, 151-178.

Wilensky, U. (1999b). *NetLogo*. Retrieved Dec 1, 2019, from http://ccl.northwestern.edu/netlogo/

Wilensky, U., & Reisman, K. (2006). Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology through Constructing and Testing Computational Theories-An Embodied Modeling Approach. *Cognition and Instruction*, *24*(2), 171–209.

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering Computational Literacy in Science Classrooms. *Communications of the ACM*, *57*(8), 24-28.

Wilkerson, M., Shareff, B., Gravel, B., Shaban, Y., & Laina, V. (2017). Exploring Computational Modeling Environments as Tools to Structure Classroom-Level Knowledge Building. Philadelphia, PA: International Society of the Learning Sciences.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

CT-based Collaborative Storytelling for Learning Programming Concepts in Python

Nicol Hui Yi PHUAN[1], Chien-Sing LEE[2*], Ean-Huat OOI[3]
[1,3] Universiti Tunku Abdul Rahman, Malaysia
[2] Sunway University, Malaysia
nic.phy76@gmail.com, chiensingl@sunway.edu.my, jamesooieh@gmail.com

ABSTRACT
With Industrial Revolution 4.0, the need to develop interdisciplinary skills has been highlighted in many curricula globally. Programming skills is one of the core skills. Aiming to develop creativity and computational thinking skills through collaborative storytelling using blocks of Python codes in a system called *FunPlay Code*; technology acceptance factors have been identified in an earlier study. This paper presents the initial developed prototype. A work-in-progress, future work involves user testing and further development, possibly, involving more open co-design.

KEYWORDS
FunPlay Code, computational thinking, Python, storytelling, collaborative

1. INTRODUCTION
Industries and trends are constantly changing and evolving. The Fourth Industrial Revolution (IR4.0) further poses an ever-looming potential threat of the loss of certain jobs in the foreseeable future. To address these concerns, much emphasis has been included in curricula to encourage the learning of programming concepts and principles and to develop programming skills to cope with the immense growth of technology in the 21st century.

In addition, Deloitte and other TechTrend analysts have encouraged developing flexibility and transfer. For instance, Stubbings in PricewaterhouseCooper's (PwC) 2018 analysis report, agrees with the general sentiments of countries and industries. She notes that "*The secret for a bright future seems to lie in flexibility and in the ability to reinvent your-self.*" As such, in her projection of the future of work in 2030, she emphasizes the need to broaden mindsets and perspectives across different knowledge branches. "*Think about yourself as a bundle of skills and capabilities, not a defined role or profession*".

1.1. Problem Statement
The scenarios introduced earlier imply that we need to be not only predictive but also adaptive and agile across disciplines. It is no longer adequate to concentrate only on a single way of thinking, learning and even working. There is a need to bridge the gap between the Arts/Humanities and the ever-expanding field of technology. This applies to every field from education, to art, fashion and even politics.

Scratch and Alice (Figures 1a and 1b) are examples of such endeavours. They combine graphical blocks but remove the obstacle of traditional programming code syntax and debugging-oriented graphical user interface. In that way, programming is made more understandable, clearer, accessible and more appealing to a broader audience.

Figure 1a. Scratch User Interface

Figure 17b. Alice User Interface

Lee and Jiang's (2019) study further assesses computational thinking skills of students' fractal Scratch projects based on Dr. Scratch's assessment rubric. Findings indicate that the main difference between experts and novices is abstraction, different perspectives and different types of media. This confirms the viability of combining both logical thinking and design thinking through collaborative storytelling. Furthermore, in line with computational thinking, the experimental playground needs to be programming-related.

The hypothesis is that since stories are logic-based, they may provide an easier and more interesting entry for novices. Furthermore, if eventually computational art comes into the picture, it may be even more motivating.

1.2. Project Objectives
This system is developed pursuant to Lee and Jiang's (2019) study and Lee and Ooi's (2019) *FunPlay Code* conceptualization study. It is intended as collaboration between three universities, two in Malaysia and one in China. Lee and Ooi's (2019) study seeks to identify design factors, which would encourage young people to code, given a collaborative storytelling system. Findings indicate that perceived ease of use, perceived usefulness and social factors are likely to influence technology adoption.

This Python-based application, *FunPlay Code for the Web*, aims to narrow the gap between design and science by:

a) encouraging youth to think logically and motivating them to adopt and/or adapt codes to create their own digital stories in a more creative way;
b) encouraging re-evaluation and/or reframing and/or

c) traversal between the Arts/Humanities and Science.

Correspondingly, in this *FunPlay Code* Python-based Web application, participants can:

a) continuously create stories or blocks of stories or contribute to others' stories or blocks of stories in Python codes;
b) compile the whole 'story' and run upon request;
c) allow users to 'like,' comment and share a specific story, or part of it onto the user's own homepage;
d) reframe the codes to form another perspective to fit their objectives; recreating something new.

1.3. Project Solution

Python is the intended language when utilizing *FunPlay Code*. To make the integration of the project less complicated, the Application Programming Interface (API) will also be coded in Python. In order to allow the use of Object-Relational Mapping in Python, Krebs' (2017) *SQLAlchemy* will be used.

1.4 Project Approach

The initial collaborative storytelling system utilizes social media and intelligent recommendation of resources (Wong & Lee, 2011). It does not involve learning of codes. *FunPlay Code's* design challenge is to tell experiences using codes, to imitate social media functions such as like, share, comment, reuse and modify others' codes to create a continuous logical collaborative story. Preliminary user requirements reported in Lee and Jiang (2019) stress on perspectives and abstraction. Lee and Ooi (2019) indicate the importance of human factors to technology adoption.

Hence, Agile Methodology is used to carry out this project. It is chosen as it allows opportunities to assess the project's direction and allows room for change throughout its development (Gonçalves, 2019).

1.5 Scope of the Project

FunPlay Code will be a platform that allows participants to create their own digital stories, adapt and reuse codes. It also allows editing of existing codes and functions to like, share, comment. The program must also be able to recognise patterns and the semantics of programming logic. It should however, only allow Python codes.

2. LITERATURE REVIEW

In 2013, 'Higher Order Thinking Skills' (HOTS) are emphasized in the Malaysian curricula across primary, secondary and tertiary education to transform education from the traditional 'drill-and-kill' method of learning to nurturing flexible, inventive mindsets (Rajaendram, 2018). The importance of HOTS has increased since then. Some theoretical foundations are presented below.

2.1. Creative Learning

Creative thinking can be seen via Problem-Based Learning. "Psychological research and theory suggest that by having students learn through the experience of solving problems, they can learn both content and thinking strategies" (Hmelo-Silver, 2004). She states that in Problem-Based Learning, a student learns not only through facts and textbooks. Instead, it is centered on addressing complicated questions that have no fixed answers. Furthermore, students are encouraged to work cooperatively in order to determine what they need to solve the problem.

In his paper *Sowing the Seeds for a More Creative Society*, Resnick (2007) opines that success does not fully depend on one's knowledge. It also depends on one's "ability to think and act creatively". He thus urges modern-day students to learn to "think creatively, plan systematically, analyse critically, work collaboratively, communicate clearly, design iteratively, and learn continuously". This gives rise to the "creative thinking spiral" (Figure 2), to guide them to "imagine" more, in multiple iterations.

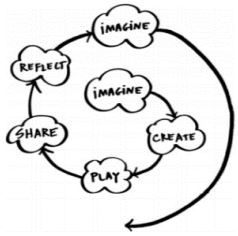


Figure 7 Creative Thinking Spiral

2.2. Computational Thinking (CT)

Another aspect of creativity is highlighted by Wing (2006). Wing describes computational thinking as a necessary skill for everyone, from young children to working adults, even if they are not in the Information Technology field.

CT involves asking questions that are frequently encountered in software patterns and even software development: What is the problem in this situation? How difficult is it to solve this problem? What is the optimum method to solve it? These are questions that build the theoretical foundations of computer science. They can also function as a set/list to solve an existing issue. It allows us to break down large numbers of probabilities and information into smaller, digestible portions.

> *"Computational thinking is thinking recursively. It is parallel processing. It is interpreting code as data and data as code."*
>
> *"Computational thinking is using abstraction and decomposition when attacking a large complex task or designing a large complex system."*

Wing's definitions help to form the bases of what the system should do. By mapping computational thinking to how a user would relate to the flow of the software, it would help enhance user's experience and assist them in understanding code logic. It also makes it easier to weigh the benefits and consequences of choices that we make.

2.3. Waterfall Methodology

The Waterfall Methodology does not allow room for the development to adapt to changes, when it is far into the last stages. For the *FunPlay Code* project, Waterfall Methodology would not be recommended as there is a high probability of changes to the functional requirements in the future. Since the project would be a collaboration between multiple universities, changes due to feedback are much expected especially later in the development process.

2.4. Agile Development Methodology

Agile Methodology centres around the ability to accept and adapt to change. It allows software development to progress and smoothly work through uncertainties faced. According to Gonçalves (2019), there are four vital values to Agile Development:

- Focus on individuals and interactions, less, on the development processes and the tools.
- Prioritize properly functioning software over overly thorough documentation.
- Build cooperative relationships between customer and developer more than contract negotiation.
- Enable flexible development and responsiveness to change, not just follow a strict plan and schedule.

With values that heavily emphasize cooperation and collaboration between the developing team and the client, it allows potential for the development team to respond to a client's feedback throughout the development process. This is especially important since *FunPlay Code* is mainly a collaborative project between Universiti Tunku Abdul Rahman and Sunway University at this point.

3. METHODOLOGY

3.1. Development Methodology

The first objective of *FunPlay Code for Web* is to build a platform that can act as a bridge between the Arts/ Humanities and Science. The methodology selected for this project is the Dynamic Systems Development Model Methodology (DSDM). DSDM is an agile iterative, incremental framework (Buehring, 2019). DSDM is chosen because DSDM focuses on a project's specific goals and objectives; shaping the project's development around its goals.

For instance, besides the set functions of creating, contributing, deleting and social sharing of digital stories, DSDM allows sufficient space for improving and adding features without compromising the main key features of the software. This is done by specifically prioritising each requirement with DSDM's principle of using MoSCow: keeping track of a requirement's priority by labelling them with 'MUST', 'SHOULD'. 'COULD' and 'WILL NOT'. Thus, the flow of the project development would focus on fulfilling the requirements before moving on to what the software is further capable of.

Furthermore, since *FunPlay Code* is intended to be interactive and used by users who may have little to no Information Technology knowledge, it is vital that the interface of the software be easily navigational and understandable to the users. DSDM's principle of prototyping ensures that for every prototype created, users would be involved to test it in order to ensure it is functioning and 'user-friendly'; allowing early discovery of flaws and bugs, room for change and possibilities and software development grows over time. Hence, as DSDM allows user involvement, and changes to be implemented during development, the outcome should be better.

3.2. Development Tools

Software development tools are used by developers for the purpose of accomplishing a specific task such as compiling, testing, maintenance or debugging. The subsections below state and describe the tools used for this project. Development tools are ReactJS and GitHub.

GitHub is good for tracking as it is possible to list down a series of functions that the software must have and should have by using a feature called GitHub Issues. This relates to the DSDM's principle of prioritising features to ensure that the project meets its stated requirements. The way that GitHub Issues function is by creating an 'issue' and specifically tagging them (Figure 3) with certain labels.

Figure 3. GitHub Issues obtained from ROBINPOWERED

GitHub would ease the load of project documentation. As project development grows, it becomes easier to forget smaller notes or minor bug fixes. Bigger loads mean more things to remember, and with GitHub feedbacks, notes can be made as reminders and flagged once completed. This ensures that most problems can be tracked and recalled more easily rather than leaving it to manual documentation. As everything would be stored and noted on GitHub, these would help developers check if what is done during development matches the requirements.

4. PROJECT INITIAL SPECIFICATION

FunPlay Code for Web is a web-based application that allows users to create their own digital stories and contribute to digital stories created by others. The software must allow sharing, commenting and 'liking' of stories.

4.1. System Requirements

4.1.1. Login
4.1.2. Create Digital Stories
4.2.3. Contribute to Digital Stories
4.2.4. View Digital Stories
4.2.5. Share Digital Stories
4.2.6. Like and Comment on Digital Stories

A sample use case description is presented in Table 1.

Table 1. 'Create Digital Stories' Use Case Description

No.	2
Use Case Name	Create Digital Stories
Actor(s)	User with Account
Short Description	Authorized Users (users with a valid account) can create new digital stories.
Trigger	User clicks the "create" button
Preconditions	Action only valid for users with existing account
Flow	1. User logs into account 2. User clicks 'Create New Story'

5. INITIAL PROTOTYPE DESIGN

The initial prototype design is as illustrated in Figures 5a, b, c. This initial prototype has not been tested yet by users as we are concerned with technology acceptance factors

identified earlier in Lee and Ooi (2019). Further development is for Android (Vegean, Lee & Ooi, 2019) and user testing. Figure 5d shows a sample screenshot of *FunPlay Code* for Android.

Figure 5a. Home Screen

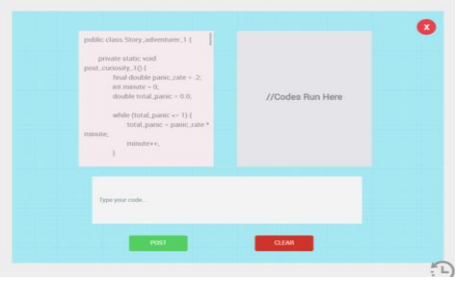


Figure 5b. Create Screen

Figure 5c. View Screen

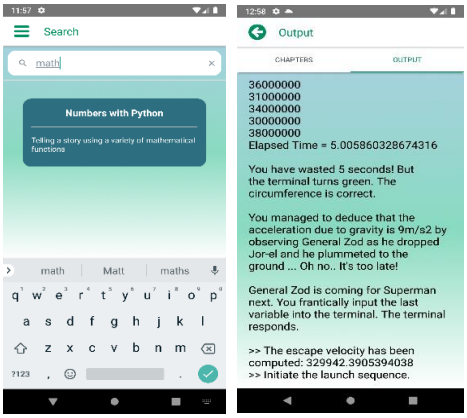


Figure5d. Sample screens from the Android version

6. CONCLUSION

The world demands innovation, creativity; a combination of design, and logic. With the theoretical foundations and methodologies in mind, we hope to minimize mental blocks involving more open co-design and to appreciate the power of computer science and its relevance in diverse aspects of our daily lives. *FunPlay Code's* success/failure will depend partly on socio-technological factors but we hope it would develop further.

7. ACKNOWLEDGEMENT

This paper is extended from Nicol's final year project carried out in Universiti Tunku Abdul Rahman, Malaysia. We thank CTE for increasing our awareness of CT research globally.

8. REFERENCES

Buehring, S. (2019). *The DSDM Principles: A Visual Guide*. Retrieved 24 March 2019, from https://www.knowledgetrain.co.uk/resources/practice/the-dsdm-principles

Dunn, Z. (n.d.), *How we Organize GitHub Issues: A Simple Styleguide for Tagging*. Retrieved 22 March 2019, from https://robinpowered.com/blog/best-practice-system-for-organizing-and-tagging-github-issues/

Gonçalves, L. (2019). *What is agile methodology*? Retrieved 26 February 2019, from https://luis-goncalves.com/what-is-agile-methodology/

Krebs, B. (2017). *SQLAlchemy ORM Tutorial for Python Developers*. Retrieved 24 February 2019, from https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/

Lee, C. S. & Jiang, B. (2019). Assessment of Computational Thinking (CT) in Scratch Fractal Projects: Towards CT-HCI Scaffolds for Analogical-fractal Thinking. *International Conference on Computer Supported Education, (1), 192-199.*

Lee, C. S., & Ooi, E.H. (2019). Design to Encourage Reframing and Lean Design through Art Science Digital Storytelling/Transformations and Analogical Thinking. *International Conference on Engineering Technology*. July 6-7, Terengganu, Malaysia.

Vegean, N., Lee, C. S. & Ooi, E. H. (2019). *FunPlayCode for Android.* Extended Universiti Tunku Abdul Rahman final year project.

Resnick, M. (2007). Sowing the Seeds for a More Creative Society. *Learning & Leading with Technology*, *35*(4), 18-22.

Stubbings, C. (2018). *Workforce of the future: The competing forces shaping 2030*. London, United Kingdom: PricewaterhouseCoopers.

Wing, J.M. (2006). Computational Thinking. *Communications of the ACM - Self managed systems*, *49*(3), 33-35.

Wong, Y. L. & Lee, C. S. (2011). Creative Storytelling Enhanced through Social Media and Intelligent Recommendation. *Creativity and Cognition*, Georgia Tech, Atlanta, November 3-6, 2011, 399-400. [Multimedia University final year student project]

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Artificial Intelligence Education

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Experiences from Teaching Actionable Machine Learning at the University Level through a Small Practicum Approach

Natalie LAO[1*], Irene LEE[2], Hal ABELSON[3]
[1,2,3] Massachusetts Institute of Technology, United States of America
natalie@mit.edu, ialee@mit.edu, hal@mit.edu

ABSTRACT
Machine learning (ML) courses have traditionally been taught through a math-first approach. They generally begin by establishing mathematical theories behind ML, such as the perceptron algorithm, logistic regression, and backpropagation, and then use these building blocks to motivate more complex structures such as neural networks. Such educational resources may not be sufficient or preferable for audiences who wish to use ML to build useful artifacts but do not have a strong mathematical or programming background. In this paper, we introduce a new framework for teaching actionable ML that combines three components in a Use-Modify-Create progression: (1) technical modules taught through hands-on labs, (2) a capstone project, and (3) supplemental lectures for new areas of research. This framework was applied in two iterations of a semester-long practicum at Massachusetts Institute of Technology (MIT) as a beginner-accessible course aimed at helping a broad range of students gain the ability to ideate and implement independent ML projects. We present the curriculum, student projects, pre and post-course survey responses, assignment grades, reflective discussions, and learnings from both iterations of the course. Our results indicate that the proposed actionable pedagogical framework for ML along with the content and practices of the course were effective for increasing students' practical self-efficacy in ML and computational identity as developers of ML applications. The findings of this study illuminate patterns of interaction with ML systems that support a practical approach to teaching ML in order to increase knowledge acquisition, future learning ability, and motivation in beginner students.

KEYWORDS
machine learning, deep learning, actionable pedagogical framework, experiential learning, small practicum

1. INTRODUCTION
As artificial intelligence (AI) and machine learning (ML) have gained prevalence in public education over the past decade, many interpretations of the two terms have been presented. We define ML as models trained on large amounts of data to inductively find patterns while AI also includes algorithms crafted from general deductive principles to solve specific problems (e.g. alpha-beta pruning and minimax); and deep learning as a subtopic of ML that uses neural networks with more than one hidden layer (Lao, Lee, & Abelson, 2019). In order for an ML system to "work", it is dependent on the availability of high quality data, scientific insights on features, appropriate model architectures, and computational processing power. Instead of being generally deterministic programs, ML results in powerful statistical algorithms that can be hard to debug and understand on a detailed level, such as when analyzing a single error in a large neural network (Shapiro, Fiebrink, & Norvig, 2018). Therefore, it may not be the most effective to teach ML in the style of other algorithms courses if we want to educate critical thinkers from a wide variety of backgrounds. In designing our teaching framework, our questions were:

- Can students with no/minimal ML or CS experience quickly apply ML to interesting and suitable problems without being explicitly taught the underlying mathematical theories?
- What human and computational resources are needed for an introductory, projects-based ML course?

This paper serves as an experience report that describes the pedagogical learnings from designing and implementing a small-scale, project-focused practicum that was successful at helping students of various technical backgrounds develop self-efficacy as machine learning project creators (Lao, Lee, & Abelson, 2019).

2. BACKGROUND

2.1. Theoretical vs. Practical Approaches
Most current ML courses teach the mathematics of ML during lessons (e.g. the perceptron algorithm or linear regression), and ask students to work on proofs or math-heavy problems for homework, which may involve translating the relevant math into code (Dror & Ng, 2018; see also Mohri, 2018). However, such methodologies may not work well for students who do not yet have a strong foundation in probability, calculus, or linear algebra. In contrast, practicums are often run as laboratory classes where students work on assignments and projects during class time with the support of mentors and/or teaching assistants. This Deep Learning Practicum course is an example of an ML *practicum* targeted towards university students of a broad range of backgrounds that takes a *practical* approach—its focus is on the "doing and use of ML" and the creation of personal projects and applications.

2.2. Experiential Learning: Use-Modify-Create
In experiential learning, or "learning through reflection and doing" (Kolb, 2014), learning can be elicited through direct manipulation of objects or systems as "objects to think with" (Papert, 1980). In our course design, experiential learning exercises are combined with a capstone project through the Use-Modify-Create Progression (Lee et al., 2011). We posit that Use-Modify-Create can help students deepen understanding of ML concepts and master practical skills: (1) students use ready-made ML models within fast-response

and user-friendly interfaces to develop high-level intuitions about training, testing, and the importance of data, (2) students manipulate the models directly to understand how architectural modules, hyperparameters, and datasets impact results for different problems, (3) students scope a problem suitable for ML and create their own application.

This methodology can now be applied to ML due to the creation of libraries that support in-browser ML experiments such as TensorFlow.js (Smilkov et al., 2019), and user-friendly applications such as Teachable Machine (Stoj.io et al, 2018) and ModelBuilder (Google, 2018). These tools allow students to modify powerful ML models (either through a user interface or code) and test the results in real-time, which enable novices to quickly gain experience through direct manipulation of ML systems. Students can quickly iterate through building a model, inputting data, training the model, and analyzing results. Furthermore, these "laboratory" experiences with ML systems provide students with experiences that directly relate to the future of work at the human-machine frontier.

2.3. Self-Efficacy and Engagement

This practicum's framework for teaching actionable ML incorporates several mechanisms for engagement: empowering students, creating meaningful experiences through scaffolded, inquiry-based learning, and authentic learning opportunities (Wu & Huang, 2007). The framework also emphasizes self-efficacy, a belief in one's chances of successfully accomplishing a task and producing a favorable outcome (Bandura, 1977). Students with high self-efficacy develop deeper interests in the tasks at hand and are more motivated to learn challenging material (Bandura et al., 2001). Recent work shows that self-efficacy is developed and strengthened through seeing others like themselves succeed, being persuaded by respected friends and advisors, and reflecting on one's own capabilities (Bandura, 2004). As such, our framework was designed to emphasize collaborative work, work with near-peer mentors, and exposure to ML professionals of diverse demographics (gender, age, and race/ethnicity).

3. INSTRUMENTS AND ASSESSMENT

The data sources used to analyze the course's impact included anonymous responses to pre and post surveys and analysis of capstone projects. At the time of the study, there were no validated instruments for measuring self-efficacy in ML. We created our post survey instrument based on validated instruments for measuring self-efficacy in general sciences, including Children's Science Curiosity Scale (Harty & Beall, 1984) and Modified Attitudes Towards Science Inventory (Weinburgh & Steele, 2000).

4. DEEP LEARNING PRACTICUM V1

The first version of the course ran for 1.5 hours 2x a week over a 15-week semester in spring 2018 at Massachusetts Institute of Technology (MIT). The course did not count towards core undergraduate requirements and was an elective course. In pre-registration, the instructors emphasized that the course was meant for students who did not feel comfortable working with ML and not experts hoping to gain advanced techniques. Class size was restricted due to the personalized, project-based nature of instruction. Twelve students completed the course.

During the course, instructors aimed to ground theoretical constructs of ML in hands-on applications that spanned different topics. Six genres were covered in the pilot that included predictive and generative applications of ML. The order of genres followed the historical development of ML, and naturally presented a progression in the sophistication of ML models. There were 3 starter topics followed by 3 advanced topics. The instructors gave short explanatory technical lectures (<15 min.) with in-class exercises in TensorFlow.js that students ran on their own laptops. The activities often leveraged existing datasets, pre-built models, and web-based tools for ML. For each set of exercises, students were asked to discuss their findings with a partner or with the class. Student teaching staff provided technical and instructional support. Weekly take-home assignments provided an extension to the environment and the exercises introduced during class.

The last 9 weeks of class focused on capstone projects and guest lectures (GLs) from ML professionals and researchers. Students chose a problem that personally interested them and was suitable for an ML application. Mentors were paired to each project. A week-by-week map of the version 1 curriculum is presented in Table 1.

Table 1. V1 of the curriculum annotated with the ITEST Use-Modify-Create progression per week.

wk.	Topics	Progression
1	K-Nearest Neighbors	**Use:** Teachable Machine webapp (Stoj.io et al, 2018). **Modify:** Confidence algorithms in source code.
2	Multilayer Networks	**Use:** Model Builder webapp (Google, 2018). **Modify:** Starter TensorFlow.js and HTML code for programming multilayer networks.
3	Convolutional Neural Networks (CNNs)	**Use:** Model Builder webapp, filter visualization webapp (Harley, 2015), Fast Style Transfer webapp (Nakano, 2018). **Modify:** Starter code for programming CNNs.
4	Generative Models and Embeddings	**Use:** Embedding Projector webapp (TensorFlow, 2018), Latent Space Explorer (deeplearn.js., 2018). **Modify:** Feature projection functions in Latent Space Explorer source code.
5	Generative Adversarial Networks (GANs)	**Use:** GAN Playground webapp (Nakano, 2017). **Modify:** Starter TensorFlow.js and HTML code for programming GANs.
6	Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTMs)	**Use:** RNN text generation webapp (Karpathy, 2015), SketchRNN webapp (Ha, Jongejan, & Johnson, 2019). **Modify:** Architecture and parameters in webapp source code. **Create:** Music generation

		RNN application through TensorFlow Python notebook.
7	Project Overview	**Create:** Students scoped and presented 3 project ideas.
8	Spring Break	**Create:** Teams/individuals worked on the project proposal writeup.
9	Project Mentor Matching, GL: Healthcare	**Create:** Teams submitted proposals and were matched with industry mentors.
10	GL: Fairness, GL: Testing and Training Tools	**Create:** Teams work on projects.
11	GL: Interpretability	**Create:** Teams work on projects.
12	Project Midpoint Checkpoint Presentations, GL: Art & Music	**Create:** Teams presented 5-minute project progress reports in class, received feedback.
13	GL: People + AI Research	**Create:** Work on projects.
14	Final Presentation Dress Rehearsal, GL: Adversarial Examples	**Create:** Teams presented a practice run of their final 10-minute project presentations in class, received feedback.
15	Final Presentation Showcase, Project Writeup Due	**Create:** Present projects in front of industry professionals and submit project writeups in the form of instructional blog posts.

4.1. Student Demographics

Of the 12 students, there were 2 (17%) second-years, 4 (33%) third-years, 5 (42%) fourth-years, and 1 (8%) graduate student. Nine (75%) majored in EE/CS, 1 in Math, 1 in Math & Physics, and 1 in Humanities. There were 3 black women, 3 Asian men, 2 Asian women, 2 white women, and 1 white man. Ten students (83%) had basic exposure to AI or ML, but wrote in the pre- survey that they wanted to take another introductory course because they did not feel that they could build practical applications. All students had at least some coding experience, but only 8 (67%) had experience in JavaScript.

4.2. Teaching Staff and Industry Mentors

There were 6 student staffers who helped debug in-class exercises for each topic, answer questions, and lead reflective discussions that directed towards learning goals for the exercises. For the 9 projects in the class, 3 of the staff mentored 1 project each and 3 mentored 2 projects each. There were 9 industry mentors. We reached out to companies and researchers in the area to ask for volunteers who have experience with ML projects. We invited all volunteers to a mixer with the students after project teams had formed. At the beginning of the mixer, each mentor gave a brief overview of their expertise and each student team summarized their project goals. After the mixer, student teams submitted their preferences for mentors and were matched. Mentors met with teams during the beginning and middle of their project cycles to help with high level ideas, resources, and project scoping.

4.3. Capstone Projects

Within this "Create" stage of the course, students marshaled the tools and techniques at their disposal along with mentorship to create capstone projects. Students were instructed to choose a project that they were personally interested in, but were also cautioned that *a realistic project implemented well and evaluated thoroughly is better than a half-implemented ambitious project with no result*. Projects could be a real-world **Application** of ML, an **Exploration** of properties of neural networks, or a **Replication** of an ML paper. To scaffold project scoping, students were given a "3 Ideas" assignment in which they presented 3 project ideas in class. For each idea, students defined a "Safe" goal that they were confident they could achieve by the end of the semester, a "Target" goal that they hoped to achieve, and a "Stretch" goal that would be good to achieve if extra time was available.

Students had the option of finding a project partner after the presentations. There were 9 projects consisting of 3 pair projects and 6 solo projects. 7 projects were in the Application category, 1 in Replication, and 1 in both Application and Exploration. All teams achieved their Safe goals. One team continued working on their project after the class ended and was able to publish a paper.

4.4. Learnings for V2

Feedback was obtained through surveys and a discussion-style post-mortem on the last day of class. Due to the small class size, quantitative analyses are not presented to preserve anonymity. Overall, students loved the interactive lab style of the modules in the class. Two students with no prior JavaScript experience felt that the course was surprisingly JavaScript-independent, although some coding experience was helpful. Students felt that the small class size was beneficial in creating an environment that made them feel comfortable speaking during the open reflective discussions that accompanied in-class exercises. Nearly every student felt that there was not enough time for project implementation, but students also said that it was the most valuable and enjoyable part of the course. Students suggested that the course should cover data collection, data processing, and using external computational resources to better scaffold the projects. Students enjoyed the guest lectures and thought that they helped "*put what we learned into a much bigger picture.*" Students noted that some guest lectures may have been useful before starting their final projects and would have provided additional context for project choices.

Several students said that the course demystified ML and made it more approachable. Two students mentioned their increased concern over bias in ML algorithms as well as a deeper understanding of how to resolve some of these issues: "*Before this course, I thought of computer programs more linearly – as if [programmers] were mostly in control of a program's results. Now I have a much greater understanding of how ML programs can be biased and unfair... I learned the importance of providing good, varied input data and how this data can have significant impact on a network and ultimately the world.*"

5. DEEP LEARNING PRACTICUM V2

5.1. Changes from V1

The second version of Deep Learning Practicum was offered at MIT in fall 2018, the semester following the pilot. There were 6 main changes from version 1: (1) the final project was introduced at the beginning of the semester and ran in parallel to the modules portion of the course, (2) there were two additional scaffolding workshops for the final project (data mining and using computing clusters), (3) students were required to have a partner for their project unless given permission, (4) guest lectures were more interspersed throughout the course instead of all at the end, (5) the staff-to-student ratio decreased from 6:12 to 6:26, and (6) an additional unit on reinforcement learning was added. The full set of version 2 curricula, lectures, assignments, and final projects can be found online at http://mit.edu/6.s198 (Lao & Abelson, 2018). A weekly summary of the version 2 curriculum is presented in Table 2 (Lao, Lee, & Abelson, 2019).

Table 2. V2 of the curriculum annotated with the ITEST Use-Modify-Create progression per week.

wk.	Topics	Progression
1	K-Nearest Neighbors, Transfer Learning	Module from version 1 wk. 1 with more emphasis on transfer learning techniques.
2	Multilayer Networks, Project Overview	Module from v1 wk2. **Create:** Scope 3 ideas for capstone final project.
3	CNNs, GL: Adversarial Attacks	Module from v1 wk3. **Modify:** Starter adversarial attack TensorFlow code on CNNs.
4	3 Ideas Project Workshop, Data Mining Workshop	**Use:** Kaggle to find datasets (Kaggle Inc., 2019). **Modify:** 3 project ideas based on feedback. **Create:** Web scraping scripts using Beautiful Soup (Python Software Foundation, 2019).
5	Generative Models and Embeddings, Computational Resources workshop, Project Mentor Matching	Module from v1 wk4. **Use:** Holyoke Computing Cluster tutorial (MGHPCC, 2018).
6	GANs	Module from v1 wk5.
7	Project Data Review, Reinforcement Learning	**Use:** Metacar webapp (thibo73800, 2019), OpenAI Gym webapps (OpenAI, 2019). **Modify:** TensorFlow starter code for RL. **Create:** Data review document to describe project dataset details.
8	RNNs and LSTMs	Module from v1 wk6.
9	Informal Project Checkpoint	**Create:** Work on projects and discuss progress with staff.
10	GL: Art & Music, GL: People + AI Research	**Create:** Finish project proposal.
11	Formal Project Checkpoint	**Create:** Work on projects and show basic working demo to staff.
12	GL: Healthcare, Project Practice Lightning Talks	**Create:** Present 2-minute project lightning talks, receive feedback for final showcase.
13	Office Hours	**Create:** Work on projects.
14	Final Presentation Showcase	**Create:** Presented capstone projects to an audience of varying ML experience with lightning talks, then individual booths.
15	Project Writeup Due	**Create:** Submit project writeups in the form of instructional blog posts.

5.2. Student Demographics

Of the 26 students, there were 5 (19%) second-years, 6 (23%) third-years, 11 (42%) fourth-years, 3 (12%) graduate students, and 1 (4%) post-doc. Twenty students (77%) majored in EE/CS, 2 in Architecture, 2 in Physics & EECS, 1 in Materials Science & Engineering, and 1 in Biological Engineering & Math. There were 9 Asian women, 6 Asian men, 5 white men, 3 Hispanic men, 1 black woman, 1 white woman, and 1 black man. Similar to V1, 21 students (81%) had basic exposure to AI or ML, but commented that they wanted to participate in the course due to self-perceived lack of ability to apply theory and math in building practical applications.

5.3. Capstone Projects

For V2 of the course, students were asked to work in groups of two for the final project due to the decrease in the staff-student ratio. Students started work on the projects in wk. 2 of the course, so they had not been exposed to all of the topic modules. The instructors were concerned that students may avoid later topics and tried to mediate this by giving lightning talks and sample use cases for the topics that would be presented later. There were 14 projects, all of which completed their Safe goal. All three project categories were represented with the majority being Application projects. More projects bridged multiple project categories than in V1, likely due to students having more time.

The first project workshop was the 3 Ideas Workshop during wk. 4, which changed in format from the pilot due to the increased number of students: The staff ran two 30-minute sessions of guided group presentations. For each session, the class was divided into four groups of 5-6 students based on shared project topic interests. One to two staffers led each group, where students took turns presenting their 3 ideas. During the final 30 minutes of the class, students were encouraged to talk to others they had met and form groups. After the 3 Ideas Workshop, there were two workshops given on project skills: data collection and how to connect to computing resources. There was also a data review checkpoint assignment due in wk. 7 to confirm that students had completed data collection and processing in a timely manner.

5.4. Post Survey Responses

The post survey was emailed out after the course ended and received 17 responses (65%). Demographic results indicated that the participants were representative of the class in terms of grade level, major, gender, and ethnicity. Table 4 presents the responses to all linear scale questions, where 5 =

"Strongly agree" or "Completely confident" and 1 = "Strongly disagree" or "No confidence."

Table 4. Means and standard deviations of post survey linear scale question responses.

	Item	Mean	S.D.
1	I felt that I was successful in this class.	4.4	0.6
2	I am proud of what I was able to accomplish in my final project.	4.2	0.8
3	I will be able to complete an ML project (of a similar level and scale to my final project) on my own.	4.6	0.5
4	In this class, I saw people like me succeed at learning ML	4.2	0.6
5	When I saw people like me succeed in ML, it made me feel that I could succeed as well.	4.3	0.7
6	How confident do you feel about describing your project to a non-technical person?	4.6	0.5
7	The project work made me feel uncomfortable	1.6	0.9

As a follow-up to Question 1, we asked "What did you use to determine your sense of success in the class?". The majority of responders attributed their sense of success to *work on the final project* (94%) and *understanding of the concepts presented in class* (88%). Responses to "Which of the following elements from the course did you use in your project work?" also indicated that the modules and workshops were helpful. More than half of responders said they *used concepts/architectures from the units* (82%), *used independent researching skills [developed] through the assignments* (59%), or *used the 3 Ideas Workshop to [help] improve or refine [their] project idea* (59%).

When asked "How can you see yourself using the tools, techniques, and methods presented in the class?", all responders gave multiple use cases. The most prevalent were: *Applying ML to new domains* (82%); *Be(ing) able to talk about it with experts* (77%); *Being able to talk about it with non-experts* (77%); *Using it for fun* (65%); *Developing my final project further* (65%); *Using it [for] another class* (65%); and *Using it as part of a job* (65%).

When asked "How did your views on ML change through taking this course?", 53% mentioned a "*personal realization of the easy application potential of ML*"; 18% had "*increased enjoyment of the field*"; 18% wrote "*realizing limitations of ML*"; and 12% were "*excited…the field is rapidly evolving*".

6. DISCUSSION

The course aimed to help students with some coding background and none to novice AI or ML knowledge gain self-efficacy in ML. In general, students highly enjoyed the course, felt that it helped demystify ML, and were encouraged to pursue independent, personal ML projects in the future. We felt that both iterations of the course were successful in our goals, with V2 allowing students more time for projects. Survey responses from V2 indicate that successful completion of capstone projects most heavily influenced development of self-efficacy in ML, and that the modules portion of the course was successful at preparing students for the projects. While our results are promising, we recognize limitations to replication: there was a high staff-to-student ratio and many students had exposure to ML/AI before the course (although we found no significant difference in performance between students of varying levels of ML and coding backgrounds).

We believe that the following 4 components of the course best contributed to its success: First, while the modules did not teach all of the skills and concepts students needed for every type of ML project, we hypothesize that the hands-on, exploratory lab work for each application helped students feel more comfortable playing with new architectures. This encouraged students to conduct research and learn on their own – 3 teams from V2 (21%) even applied methods not taught in the class to their projects. Second, TensorFlow.js allowed beginners to dive directly into exploring complex and visually appealing ML applications – modifying ML models in the browser allowed for near-instantaneous feedback and reduced infrastructure problems. Third, mentors for each project greatly assisted students in properly scoping problems and finding resources. Fourth, the blog post style for the final project writeup helped students learn disciplinary sharing norms, situate their work in the community, and identify with other ML developers, enthusiasts, and researchers. Thirteen of the 14 projects from V2 shared their project blog posts publicly on the web (Lao & Abelson, 2018).

In replicating this course, the advanced modules (wk. 4+) can be replaced based on the types of projects instructors want to encourage (e.g. more types of RNNs, LSTMs, and GANs for an arts-focused ML class). Additionally, we found that transfer learning was extremely useful – students were able to adjust and retrain high quality pre-built models with great results for repurposed use instead of spending a long time trying to create (often ineffective) models from scratch. We recommend encouraging students to research and experiment with different architectures as often as possible. Many of the students without coding experience also suggested that a version of the course could be adapted for high school students.

7. REFERENCES

Bandura, A. (1977). Self-Efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological Review, 84*(2), 191–215.

Bandura, A. (2004). Health Promotion by Social Cognitive Means. *Health Education Behavior: The Official Publication of the Society for Public Health Education, 31*(2), 143–164.

Bandura, A., Barbaranelli, C., Caprara, G. V., & Pastorelli, C. (2001). Self-Efficacy Beliefs as Shapers of Children's Aspirations and Career Trajectories. *Child Development, 72*(1), 187-206.

Deeplearn.JS. (2018). *Font-Explorer*. Retrieved November 18, 2019, from https://github.com/mintingle/font-explorer

Dror, R., & Ng, A. (2018). *CS229: Machine Learning*. Retrieved November 18, 2019, from http://cs229.stanford.edu/syllabus.html

Ha, D., Jongejan, J., & Johnson, I. (2019). *Draw Together with a Neural Network*. Retrieved November 18, 2019, from https://magenta.tensorflow.org/sketch-rnn-demo

Harley, A. (2015). *3D Visualization of a Convolutional Neural Network*. Retrieved November 18, 2019, from http://scs.ryerson.ca/~aharley/vis/conv/

Harty, H., & Beall, D. (1984). Toward the Development of a Children's Science Curiosity Measure. *Journal of Research in Science Teaching*, *21*(4). 425-436.

Kaggle Inc. (2019). *Kaggle: Your Home for Data Science.* Retrieved November 18, 2019, from https://www.kaggle.com/

Google. (2018). *People + AI Research Initiative: Deeplearn.JS Model Builder Demo.* Retrieved November 18, 2019, from http://courses.csail.mit.edu/6.s198/spring-2018/model-builder/src/model-builder/

Karpathy, A. (2015). *RecurrentJS Sentence Memorization Demo*. Retrieved November 18, 2019, from https://cs.stanford.edu/people/karpathy/recurrentjs/

Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*. Pearson FT Press, New Jersey. Print.

Lao, N., & Abelson, H. (2018). *6.S198: Deep Learning Practicum, Fall 2018*. Retrieved November 18, 2019, from http://mit.edu/6.s198

Lao, N., Lee, I., & Abelson, H. (2019). A Deep Learning Practicum: Concepts and Practices for Teaching Actionable Machine Learning. *Proceedings of 12th Annual International Conference of Education, Research and Innovation (ICERI'19)*. International Academy of Technology, Education and Development (IATED), 10.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads*, *2*(1), 32-37.

The MGHPCC. (2018). *The Massachusetts Green High Performance Computing Center*. Retrieved November 18, 2019, from http://www.mghpcc.org

Mohri, M. (2018). *Foundations of Machine Learning – CSCI-GA.2566-001*. Retrieved November 18, 2019, from https://cs.nyu.edu/~mohri/ml18/

Python Software Foundation. (2019). *Beautiful Soup, version 4.8.1*. Retrieved November 18, 2019, from https://pypi.org/project/beautifulsoup4/

Nakano, R. (2017). *GAN Playground – Explore Generative Adversarial Nets in your Browser*. Retrieved November 18, 2019, from https://reiinakano.github.io/gan-playground/

Nakano, R. (2018). *Deeplearn.JS Style Transfer Demo*. Retrieved November 18, 2019, from https://reiinakano.github.io/fast-style-transfer-deeplearnjs/

OpenAI. (2019). *Gym*. Retrieved November 18, 2019, from https://gym.openai.com/

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. Basic Books, New York. Print.

Shapiro, R. B., Fiebrink, R., & Norvig, P. (2018). How Machine Learning Impacts the Undergraduate Computing Curriculum. *Communications of the ACM*, *61*(11), 27-29.

Smilkov, D., Thorat, N., Assogba, Y., Yuan, A., Kreeger, N., Yu, P., Zhang, K., Cai, S., Nielsen, E., Soergel, D., Bileschi, S., Terry, M., Nicholson, C., Gupta, S. N., Sirajuddin, S., Sculley, D., Monga, R., Corrado, G., Viégas, F. B., & Wattenberg, M. (2019). TensorFlow.js: Machine Learning for the Web and Beyond. *CoRR*.

Stoj.io, Use All Five, Creative Lab, and PAIR team at Google. (2018). *Teachable Machine*. Retrieved November 18, 2019, from https://teachablemachine.withgoogle.com/v1/

TensorFlow. (2018). *Embedding projector – visualization of high-dimensional data*. Retrieved November 18, 2019, from http://projector.tensorflow.org/

thibo73800. (2019). *Metacar: A reinforcement learning environment for self-driving cars in the browser*. Retrieved November 18, 2019, from https://www.metacar-project.com/

Weinburgh, M. H., & Steele, D. (2000). The Modified Attitudes Toward Science Inventory: Developing an Instrument to Be Used with Fifth Grade Urban Students. *Journal of Women and Minorities in Science and Engineering*, *6*(1). 87-94.

Wu, H. K., & Huang, Y. L. (2007). Ninth-Grade Student Engagement in Teacher-Centered and Student-Centered Technology-Enhanced Learning Environments. *Science Education*, *91*(5), 727-749.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Using Transfer Learning, Spectrogram Audio Classification, and MIT App Inventor to Facilitate Machine Learning Understanding

Nikhil BHATIA[1*], Natalie LAO[2*]
[1,2] Massachusetts Institute of Technology, USA
nwbhatia@mit.edu, natalie@mit.edu

ABSTRACT
Recent advancements in deep learning have brought machine learning and its many applications to the forefront of our everyday lives. As technology has become more and more integrated into our educational curriculum, researchers have focused on creating deep learning tools that allow students to interact with machine learning in a way that incites curiosity and teaches important concepts. Our research contribution focuses on applying transfer learning and spectrogram audio classification methods to teach basic machine learning concepts to students. We introduce the Personal Audio Classifier (PAC), a web interface that allows users to train and test custom audio classification models that can classify 1-2 second sound bites recorded by the user. We also contribute a custom App Inventor extension that allows users to use the output of the web interface to create App Inventor applications that rely on their trained custom audio classification model.

KEYWORDS
machine learning, transfer learning, App Inventor

1. INTRODUCTION
From personal voice assistants to self-driving cars, machine learning applications have permeated every aspect of our daily lives. Much of these advances are thanks to the subfield of machine learning known as deep learning, a field primarily concerned with building large neural networks to perform specialized tasks. Yet as researchers began to make significant advancements in deep learning during the past decade, it became clear that computational complexity, training time, and esoteric development tools could pose as a deterrent to widespread development of deep learning applications. Transfer learning was born out of this deficiency, spurred by Yosinski's work (Yosinki, 2014) on transferable features in deep neural networks.

1.1. Transfer Learning
Transfer learning is a machine learning method where an existing deep learning model is used as the starting point to train a model specialized for a slightly different task. The ability to start with a pre-trained model allows new developers to apply deep learning to solve novel problems without the vast compute and time resources normally needed to train neural networks from scratch. While the conventional model-training process is likely only accessible to researchers or institutions with deep pockets, the result is one that should be available to developers of all levels and even students of any age. Transfer learning has allowed for just this, giving machine learning enthusiasts around the world the ability to build their own models using complex models as a starting point.

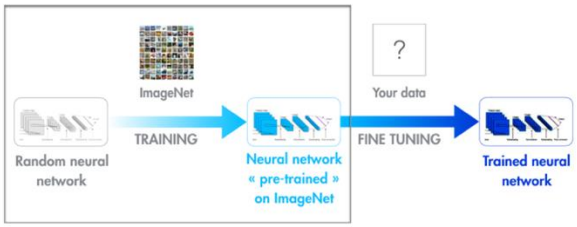


Figure 1. Transfer learning starts with a pre-trained model and fine-tunes the output layers to specialize towards a new task.

1.2. TensorFlow.JS and App Inventor
Wwe introduce two important technologies, Tensorflow.js (Tensorflow.js, 2015) and MIT App Inventor (MIT App Inventor, 2010), that this project utilizes to help students develop exposure to machine learning concepts without requiring a deep computer science background. Tensorflow.js is a Javascript machine learning library that has recently found success in the niche bridging machine learning implementation and educational tools. It allows for deep learning models to be trained and run right in the browser, and when combined with a well-designed web GUI, can hide the complexities of programming syntax while still allowing users to interface with machine learning models. Similarly, MIT App Inventor is a free open-source web platform that allows users to create mobile applications via a drag-and-drop interface, requiring little to no programming experience while still offering rich application functionality. App Inventor also offers the ability to add custom extensions to any app, allowing us to build an audio classification extension that students could upload and use to help build his private diary app. With these two technologies, we've created a web app that blends PIC (Tang, 2018) and Teachable Machine (Google, 2019), allowing users to train an audio classification model that can recognize 1-2 second audio clips. After using this web app to train a custom model, users will be able to download this model and plug it into MIT App Inventor as an extension to build apps with custom audio-classification functionality.

2. APPROACH

3.1. Personal Audio Classifier
We present a web application (Personal Audio Classifier, or PAC) that allows users to train a custom audio classifier using Tensorflow.js within the browser. The application is available to the public at https://c1.appinventor.mit.edu. This section will detail the basic functionality, as well as the machine learning tools that were used to implement an in-browser audio classifier. First, users are prompted to add custom labels that the classifier will attempt to differentiate between. Users can then record an unlimited number of audio clips for each label that will be used to train the

internal model. Each audio clip is one second long, and client side JavaScript is used to up-sample each audio clip to 384,000 Hz. Each element in the audio buffer is passed through a Fast Fourier Transform to draw the audio frequencies onto a single pixel sliver of our output spectrogram. This spectrogram provides a visual representation of the recorded audio bite, and is attached to the corresponding label so that the user can view each audio clip in the browser.

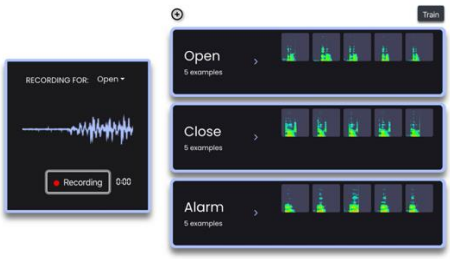


Figure 2. The label view allows users to add custom labels and record corresponding audio clips. Audio clips are up-sampled
and converted to spectrograms in the browser.

After inputting a number of labels and recording the corresponding audio clips, the user is prompted to train a custom model using their provided training data, specifying hyperparameters like Learning Rate, Optimizer, Epochs, and Training Data Fraction. The web application then proceeds to load a pretrained ImageNet model (MobileNet) and train a custom machine learning model in the browser using the activations outputted from passing the training data through the pretrained model. After experimenting with a variety of model architectures, we decided to standardize the custom model to have a single convolutional layer, a single flatten layer, and two dense layers. The output of the model is then passed through a SoftMax layer to generate probabilities that correspond to the user-inputted labels.

A separate page allows the user to use this custom trained model as a classifier, recording audio clips that are passed back through the model and assigned to one of their original labels. The corresponding label confidences are displayed after each clip is recorded, and we aggregate the test results so the user can analyze the success of their custom classifier, and even download the custom model for use in the App Inventor extension.

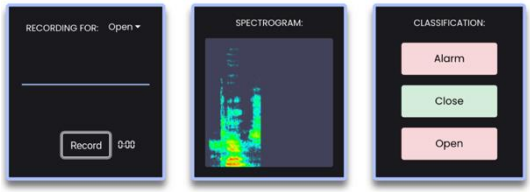


Figure3. The test view allows users to record and classify audio clips that are converted to spectrograms and passed through the custom classifier.

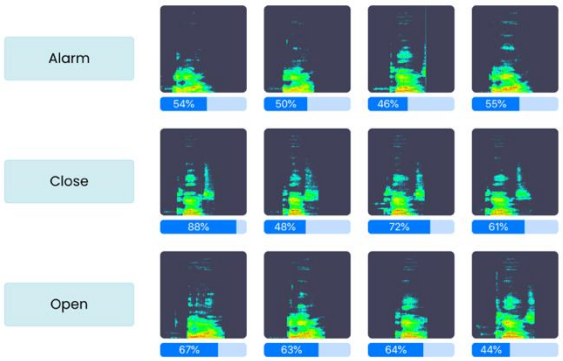


Figure 4. The test view also provides the aggregated results from classifying user-recorded audio clips.

3. REFERENCES

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, Hartwig. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* Retrieved December 12, 2019, from https://arxiv.org/abs/1704.04861

Jamarshon et al. (n.d.). *Pytorch.* Retrieved December 12, 2019, from https://github.com/pytorch/audio/blob/master/torchaudio/transforms.py

MIT (n.d.). *App Inventor*. Retrieved December 12, 2019, from https://appinventor.mit.edu/

MIT (n.d.). *Personal Image Classifier*. Retrieved December 12, 2019 from https://classifier.appinventor.mit.edu/

PyTorch (n.d). *Homepage.* Retrieved December 12, 2019 from https://pytorch.org/

Stakky, Lamberta, B., & Daoust, M. (n.d.). *Tensorflow Audio Recognition*. Retrieved December 12, 2019, from https://github.com/tensorflow/docs/blob/master/site/en/r1/tutorials/sequences/audio_recognition.md

Tensorflow (n.d.). *Homepage.* Retrieved December 12, 2019 from https://tensorflow.org/

Tensorflow.js (n.d.). *Homepage*. Retrieved December 12, 2019 from https://tensorflow.org/js

Tensorflow (2019). *Speech Commands.* Retrieved December 12, 2019, from https://github.com/tensorflow/tfjs-models/tree/master/speech-commands

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). *How transferable are features in deep neural networks?* Retrieved December 12, 2019, from https://arxiv.org/abs/1411.1792

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Artificial Intelligence Education: A Balanced Approach Using both Classical AI and Modern AI

Kwong-Cheong WONG
College of Professional and Continuing Education, PolyU, Hong Kong
kwongcheong.wong@cpce-polyu.edu.hk

ABSTRACT
This paper presents an approach to AI education, which combines both Classical AI and Modern AI. It argues that this approach can enhance students' computational thinking through explicit programming. The applicability of this approach is illustrated with the design of a short course aimed at introducing AI to secondary school students.

KEYWORDS
computational thinking, artificial intelligence education, classical AI, modern AI, chatbots

1. INTRODUCTION
In recent years, in responding to the international call for incorporating computational thinking and AI into school education, many school educators have started to design AI courses for their students; see, e.g., (Holmes, Bialik, & Fadel, 2019) and (Touretzky, Gardner-McCune, Martin, & Seehorn 2019). However, these AI courses tend to teach *exclusively* Modern AI (which is based on Machine Learning, particularly Deep Learning), at the expense of Classical AI (which is based on symbolic logic). This tendency is hardly surprising, given that Deep Learning is currently the most powerful and high-profile approach to AI, and has generated a lot of hype. However, it is my contention that Classical AI still has its merits in AI education and we should take a balanced approach, combining *both* Classical AI and Modern AI. There are several advantages for adopting this balanced approach, the main one being that Classical AI is better than Modern AI in teaching computational thinking to school students.

2. CLASSICAL AI *VS* MODERN AI
Classical (Symbolic) AI, also called GOFAI ("Good Old-Fashioned AI"), was born in the now famous Dartmouth Conference of 1956 (Haugeland, 1989). It is also called the Logic-Based AI as it is based on symbolic logic, and its idea, according to John McCarthy, one of the pioneers of AI, is that "an agent can represent knowledge of its world, its goals and the current situation by sentences in symbolic logic and decide what to do by inferring that a certain action or course of action is appropriate to achieve its goals" (Minker, 2000, p. 39). One distinctive feature of Classical AI is that *intelligence is explicitly programmed*, say in the form of a comprehensive list of if-then-else rules. Consequently, the designer of a Classical AI system needs to think carefully through all the possible combinations and devise a rule-based system that can make decisions by traversing through the pre-defined rule path. In stark contrast, Modern (Sub-symbolic) AI is based on Machine Learning, which can be defined, according to Andrew Ng, as "the process of inducing intelligence into a system or machine *without explicit programming*". Deep Learning is just a particular type of Machine Learning that deals with powerful algorithms inspired by the biological structure of the human brain, so-called deep neural networks, to endow machines with intelligence. Consequently, the designer of a Modern AI system does not need to encode the system with a comprehensive list of all possible rules; all he does is let the system learn on its own from the data.

Based on modelling logical reasoning, Classical AI, had, in its early years, developed systems that successfully solve interesting and important problems in specialized domains (Neapolitan & Jiang, 2018, p. 4), e.g., the rule-based expert system MYCIN and the rule-based chatbot ELIZA, both in the restricted medical domain. Despite these early successes, Classical AI in its traditional form is now widely agreed to have failed in building *true* artificial intelligence (Miracchi, 2019, p. 594). In stark contrast, Modern AI, powered by Deep Learning, has, in recent years, made extraordinary advances in a broad range of varied pattern recognition tasks, including classification, object detection, speech recognition, etc. – though, importantly, reasoning tasks still elude Deep Learning (Skansi, 2018, p. 13). As a result, Modern AI has recently replaced Classical AI as the most promising technology to realize *true* artificial intelligence.

However, Modern AI has its drawbacks, one of which concerns *explainability* (or *interpretability*) – it is still not very clear as to exactly how a problem is being solved, especially for Deep Learning, since deep neural nets are still poorly understood mathematically, though Explainable AI or Interpretable AI is a hot research topic (Molnar, 2019). Consequently, most users often treat a Modern AI system as a black box. But this is unacceptable when the decision provided by the system affects the person, e.g., a medical diagnosis, in which the reasoning behind the decision is also important (Kelleher, 2019, p. 245). In stark contrast, the inner working of a Classical AI system, due to its being explicitly programmed, is fully explainable.

3. A BALANCED APPROACH TO AI EDUCATION
Based on the aforementioned differences between Classical AI and Modern AI, I hereby propose a balanced approach to teaching AI, chiefly in school education. This approach combines *both* Classical AI and Modern AI. While the inclusion of Modern AI hardly needs justification – it is, after all, the focal point where all the current fascination and excitement about AI lie, the inclusion of Classical AI, a widely regarded out-of-fashion approach, demands some justifications and explanations. All in all, there are four reasons (or advantages) for teaching Classical AI in school education: the pedagogical reason, the practical reason, the

historical reason, and the philosophical reason. First, and the foremost, unlike Modern AI, which is often treated as a black box by the learner, Classical AI by its very nature requires explicit programming from the learner. This therefore can provide a valuable chance for students to practice programming when building a Classical AI system, e.g., a rule-based expert system or chatbot. Consequently, teaching Classical AI can help train and enhance students' computational thinking skills. Second, unlike Modern AI, which is based on advanced mathematics, Classical AI is based simply on symbolic logic (Kowalski, 2011) which, by its logical nature, should be more accessible to school students. Evidence, e.g., (Yuen, Reyes, & Zhang, 2019), has shown that school students can learn symbolic logic effectively through logic programming. Third, Classical AI is an important part of the history of AI. It had made many achievements in the past years, which are regarded as the milestones in the human's quest for artificial intelligence, e.g., ELIZA, the Logic Theorist, the General Problem Solver, MYCIN, and Deep Blue, just to name a few. All these should be told to the students of AI so that they can have a more complete picture of the development of AI as a discipline. Fourth, the fact that Classical AI has been good at reasoning tasks and Modern AI has been good at pattern recognition tasks has made philosophers speculate that reasoning is inherently rule-based and cannot be learned. So perhaps Classical AI and Modern AI are complementary to each other and one can never replace the other.

4. THE COURSE

To illustrate its applicability, I designed a short course using this approach. The goal of this course is to introduce AI to Form 3 and Form 4 students who have had some experience in programming (e.g., Scratch). The duration of the course is 15 hours, divided into two main parts, with the first part about Classical AI and the second part about Modern AI; see Figure 1. In the first part, the instructor teaches students how to program in the logic programming language Prolog. With support from the instructor, students are then asked to implement a simple rule-based expert system in Prolog (Bramer, 2013), and a simple ELIZA-like rule-based chatbot (O'Keefe, 1990). The second part of the course teaches students the basic ideas of neural networks (which can be introduced as extensions of linear regression). With support from the instructor, students are asked to implement a shallow, and then a deep, neural network in R to recognize handwritten digits (Liu & Maldonado, 2018), which involves very little coding, and to build a deep learning chatbot *without* coding using a free online platform. At the end of the course, the students will be able to compare and contrast the two different approaches to AI, thereby enhancing their understanding of both.

5. CONCLUSION AND FUTURE WORK

I have proposed a balanced approach to AI education in school. This balanced approach has the advantage that students can learn computational thinking through explicit programming in Classical AI. As planned, this short course will be delivered to a cohort of secondary school students. Feedback about this approach will then be collected and evaluation followed.

Part 1. Classical AI
1.1 The History of AI
1.2 Programming in Prolog
1.3 Implementing an Expert System and a Chatbot

Part 2. Modern AI
2.1 Implementing a Shallow Neural Network and a Deep Neural Network for Handwritten Digit Recognition
2.2 Building a Deep Learning Chatbot
2.3 The Future of AI

Figure 1. Contents of the course

6. REFERENCES

Bramer, M. (2013). *Logic Programming with Prolog*. Secaucus: Springer.

Haugeland, J. (1989). *Artificial Intelligence: The Very Idea*. MIT press.

Holmes, W., Bialik, M., & Fadel, C. (2019). *Artificial Intelligence in Education: Promises and Implications for Teaching and Learning*. Boston, MA: Center for Curriculum Redesign.

Kelleher, J. (2019). *Deep Learning*. MIT Press.

Kowalski, R. A. (2011). *Artificial Intelligence and Human Thinking*. Cambridge University Press.

Liu, Y. H., & Maldonado, P. (2018). *R Deep Learning Projects: Master the Techniques to Design and Develop Neural Network Models in R*. Packt Publishing Ltd.

Minker, J. (ed.) (2000). *Logic-Based Artificial Intelligence*. Boston, MA: Springer.

Miracchi, L. (2019). A Competence Framework for Artificial Intelligence Research. *Philosophical Psychology, 32*(5), 589-634.

Molnar, C. (2019). *Interpretable Machine Learning*. Lulu. com.

Neapolitan, R. E., & Jiang, X. (2018). *Artificial Intelligence: With an Introduction to Machine Learning*. Chapman and Hall/CRC.

O'Keefe, R. A. (1990). *The Craft of Prolog*. MIT press.

Skansi, S. (2018). *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer.

Touretzky, D., Gardner-McCune, C., Martin, F., & Seehorn, D. (2019). Envisioning AI for K-12: What Should Every Child Know about AI?. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*, 9795-9799.

Yuen, T., Reyes, M., & Zhang, Y. (2017). Introducing Computer Science to High School Students Through Logic Programming. *Theory and Practice of Logic Programming, 19*(2), 204-228.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Analysis of the Current Situation and Hotspots of Artificial Intelligence Education in China ——Visual Analysis based on Chinese Literature from 2015 to 2019

Zhihui GONG[1], Qiuping HU[2], Junjie SHANG[3*]
[1,3] Learning Science Laboratory in College of Education, Peking University, China
[2] Chaoyang Branch of Beijing Institute Of Education, China
1801213879@pku.edu.cn, hqpdan@163.com, jjshang@pku.edu.cn

ABSTRACT
The rapid development and widespread application of artificial intelligence have attracted great attention from the education community. The integration of artificial intelligence and education has played a huge role in educational reform in mainland China. The article takes 7 major journals of educational technology majors in mainland China as data sources and uses a bibliometric method to visually analyze articles on the subject of "artificial intelligence education" from 2015 to 2019, and summarizes research on artificial intelligence education in mainland China. Status and research hotspots. Through analysis, it is found that research on artificial intelligence education in mainland China mainly focuses on how to develop education in the era of artificial intelligence, how to organize teaching, how students learn, and the application of artificial intelligence education supported by new technologies.

KEYWORDS
artificial intelligence, artificial intelligence education, educational application

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

中国大陆人工智能教育研究现状及热点分析
——基于 2015 - 2019 年中文文献的可视化分析

龚志辉[1]，胡秋萍 2，尚俊杰 3*
[1,3]北京大学教育学院学习科学实验室，中国
[2]北京教育学院朝阳分院，中国
1801213879@pku.edu.cn, hqpdan@163.com, jjshang@pku.edu.cn

摘要
人工智能的迅速发展和广泛应用已经引起了教育界的极大关注，人工智能与教育的融合对中国大陆的教育变革起到了巨大的作用。文章以中国大陆 7 本教育技术学专业重要期刊为数据源，采用文献计量的方法对 2015-2019 年以“人工智能教育”为主题的文章进行了可视化分析，总结梳理中国大陆人工智能教育研究的现状及其研究热点。通过分析发现，中国大陆对人工智能教育的研究主要集中在人工智能时代教育如何发展，教学如何组织，学生如何学习以及新技术支持下的人工智能教育应用研究。

关键词
人工智能；人工智能教育；教育应用

1. 前言
国务院印发的《新一代人工智能发展规划》指出，随着人工智能的快速发展，教育呈现出深度学习、跨界融合、人机协同、群智开放、自主操控等新特征，人工智能技术在教育中的应用越来越重要（高丹阳，2019）。近年来，人工智能技术与教育的融合，也推动了中国大陆的教育教学变革，智能教育、学习分析、机器学习、计算思维等正不断渗透并影响着教育教学系统。在此背景下，研究中国大陆教育技术学领域所关注的人工智能教育热点问题，有希望为推动人工智能与教育的融合创新提供借鉴和指导。

2. 关键词聚类分析
1. 数据来源和研究工具
本研究的数据来源于中国大陆 7 本教育技术学重要期刊，分别是《中国电化教育》、《电化教育研究》、《中国远程教育》、《开放教育研究》、《现代教育技术》、《现代远程教育研究》和《远程教育杂志》，时间范围是 2015 年 1 月——2019 年 12 月，检索条件是主题为“人工智能教育”，经过剔除其中新闻稿、征稿通知等文章，共得到 287 篇文献。研究选择对教育技术学专业的重要期刊进行分析，目的是总结和梳理教育技术研究者近五年所关注的人工智能教育问题，希望对后续开展人工智能教育研究的学者提供参考和借鉴。

2. 关键词聚类结果
利用 CiteSpace 对文献数据进行关键词聚类分析，可以帮助探索该研究领域的研究热点和研究前沿。研究将时间分割定为 1 年，将引文关键词作为网络节点进行分析，聚类图谱收集了排名前 50 的关键词。经过聚类计算后模块值(Q 值)为 0.485，平均轮廓值(S 值) 大于 0.5377，意味着划分出来的图谱结构是显著的。出现频次排名前 10 的关键词及其中心度如表 1 所示。中心度指一个结点担任其它两个结点之间最短路的桥梁的次数。一个结点充当“中介”的次数越高，它的中心度就越大。

表 1 频次排名前 20 的关键词

序号	频次	中心度	关键词
1	148	0.46	人工智能
2	25	0.3	智能教育
3	22	0.24	教育信息化
4	20	0.21	智慧教育
5	16	0.14	大数据
6	14	0.12	教育应用
7	13	0.11	学习分析
8	11	0.1	个性化学习
9	11	0.08	教育信息化 2.0
10	10	0.08	计算思维

关键词共现图谱是指根据所引文献中关键词共现的情况绘制，两个关键词出现在同一篇文献中即视为一次合作，主要依据关键词共现频次矩阵。在“人工智能教育”领域中，利用谱聚类算法，共生成 9 个主要的聚类，分别是: 教育信息化、智能教育、计算思维、学习分析、人工智能教育应用、教育、创客教育、人才培养和知识图谱。聚类后的共现关键词图谱如图 3 所示。

图 1 聚类后的共现关键词图谱

由 CiteSpace 基于聚类关键词生成的时间线图也可视化地表现了这一现状。基于关键词和聚类的时间线图如图 4 所示。基于聚类关键词生成的时间线图可以显示出每个聚类里关键词的发展情况。例如计算思维这一类的发展情况可以概括为由计算思维到智能技术再到儿童编程教育。

图 2 基于关键词聚类的时间线图

3. 人工智能教育的研究热点和前沿分析

通过关键词聚类结果和对高被引文献进行内容分析，进一步将中国人工智能教育的研究现状总结归纳为理论探索和应用研究两方面，理论探索主要研究人工智能对教育产生的影响、作用机制，以及国内人工智能教育的发展路径等。应用研究体现为新技术支持下的人工智能教育应用现状及对策研究，例如大数据、学习分析、计算思维和机器学习等。

1. 人工智能教育理论研究

通过聚类，我们发现教育信息化、智能教育、智慧教育等内容在人工智能教育的研究中占有极大的比重，而这类研究主要集中在在如今的技术环境下人工智能时代教育如何发展，教学如何组织，学生如何学习以及中国大陆人工智能教育发展的过程和趋势等内容。例如，钟绍春讨论了人工智能时代推进教育信息化 2.0，实现教育创新发展的方向与目标、实施路径、面临的机遇和挑战。并从技术支持教与学瓶颈性问题的解决策略出发，构建促进策略实施的智慧支撑系统，设计基于智慧系统的教学活动实施方案等（钟绍春，2018）。贾积有认为从教育的本质特征和人工智能的研究领域来分析人工智能与教育的关系，可以发现教育是提高人的自然智能的过程和系统；人工智能是在机器上实现的教育，人工智能必将对人类的教育与学习方式产生重大影响（贾积有，2018）。吴永和从人工智能+教育的孕育条件、特征、作用三方面阐述了“人工智能+教育”的内涵，从应用形态、技术架构、业态趋向等要素构建了“人工智能+教育”的生态系统，并阐述了“人工智能+教育”的人才培养体系（吴永和，2017）。

2. 新技术支持下的人工智能教育应用研究

在关键词聚类图谱中，我们发现大数据、学些分析、计算思维、5G 等关键词也占有极大的比重。伴随着人工智能技术的不断发展和成熟，新技术支持下的人工智能教育应用也逐渐走进教育研究者的视野。大数据领域的研究主要集中在大数据时代教育教学的变革，教育技术研究新范式的提出以及基于大数据分析的学科教学路径等。学习分析技术是测量、收集、分析和报告有关学生的学习行为以及学习环境的数据，用以理解和优化学习及其产生的环境的技术（顾小清，2012），学习分析技术可作为教师教学决策、优化教学的有效支持工具，也可为学生的自我导向学习、学习危机预警和自我评估提供有效数据支持，还可为教育研究者的个性化学习设计和增进研究效益提供数据参考。随着图形化编程和机器人编程教育的不断普及，计算思维的培养也受到了来自学者和一线教师的持续关注，目前计算思维的研究主要集中在理论探索和培养方案研究两方面。理论探索主要讨论了计算思维的概念演变、构成要素、测评方式等。培养方案研究则关注中小学计算思维培养模式及课程实践，同时还有一部分学者关注基于大学信息技术基础课程的计算思维培养和发展研究。

4. 总结

为促进教育信息化的不断深化，还需对人工智能教育进行深入研究，促进教育和人工智能的深度融合，构建信息化学习环境和数字化学习资源，借助新技术创新教育研究范式，探讨新技术在教育教学中的应用。需要指出的是，研究还存在一些不足，文献数据只选择了教育技术学领域的期刊文章，大量其他学科领域的文章未予采用，在研究热点的总结上可能存在偏差。在人工智能教育的研究热点和前沿分析上，阅读的文献基数较小，存在一定的主观性和概括性。人工智能在和教育融合的过程中，会不断出现值得探讨的问题，我们期待能通过人工智能提升教师的教学，加强学生的学习，丰富教育研究的手段，让教育一直充满新的活力。

5. 参考文献

王亚飞和刘邦奇（2018）。智能教育应用研究概述。**现代教育技术，**6-12。

张剑平（2003）。关于人工智能教育的思考。**电化教育研究，(1)**，24-28。

张坤颖和张家年（2017）。人工智能教育应用与研究中的新区、误区、盲区与禁区。**远程教育杂志，35**(5)，54-63。

吴永和。刘博文和马晓玲和构筑"人工智能+教育"的生态系统。**远程教育杂志，35**(5)，27-39。

杨现民、张昊、郭利明、林秀清和李新（2018）。教育人工智能的发展难题与突破路径。**现代远程教育研究，153**(3)，32-40。

钟绍春和唐烨伟（2018）。人工智能时代教育创新发展的方向与路径研究。**电化教育研究，39(10)**，17-22，42。

高丹阳和张双梅（2019）。人工智能在教育领域的研究现状与特征分析。**中国教育信息化，(13)**。

顾小清、张进良和蔡慧英（2012）。学习分析：正在浮现中的数据技术。**远程教育杂志，30**(1)，18-25。

贾积有（2018）。人工智能赋能教育与学习。**远程教育杂志，36**(1)，39-47。

梁迎丽和刘陈（2018）。人工智能教育应用的现状分析、典型特征与发展趋势。**中国电化教育，(3)**，24-30。

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking Development in Higher Education

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Teaching Computational Thinking and Python Programming for Business Students: A Preliminary Study of the Alignment of Teaching and Learning Strategies with Bloom's Taxonomy of Learning Outcomes

Gabriel Chun-Hei LAI[1]*, Ron Chi-Wai KWOK[2], Joseph Siu-Lung KONG[3]
[1,2,3]City University of Hong Kong, Hong Kong
gabrilai2-c@my.cityu.edu.hk, isron@cityu.edu.hk, joseph.kong@cityu.edu.hk

ABSTRACT
Teaching computational thinking for business students at the University level has been challenging because business students normally have little and/or heterogenic computer programming background. Also, there are very few literature that examines the alignment of appropriate teaching and learning theories/strategies with different levels of cognitive processes/learning outcomes for teaching business students computational thinking. This preliminary study is to address these gaps by proposing and exemplifying an alignment of six teaching and learning strategies with the six levels of the Bloom's taxonomy of learning outcomes for teaching business students, with different computer programming background, Python programming at the University level. University lecturers could use these six proposed teaching and learning strategies as a guideline to design their course contents and materials for teaching business students Python programming at the University level. Further research direction was discussed.

KEYWORDS
computational thinking, Python programming, Bloom's Taxonomy, teaching and learning strategies

1. INTRODUCTION
Technology has been blooming and improving over the past decade, computational thinking and programming experience has become highly desirable skillsets required by business industries. There are many programming languages in the market, such as C++, Java, Matlab, etc. However, Python stands out from other programming languages and is growing in recent years.

Therefore, many business schools have been trying to include Python into their curriculum to teach business students computational thinking concepts and programming skills. This raises the question of how to teach students, especially business students, to learn Python effectively. There are plenty of literature introducing various teaching and learning theories and strategies in general subjects such as marketing and economics, but teaching Python is comparatively new in business schools. In particular, teaching computational thinking for business students at the University level has been challenging.

More specifically, one of the greatest challenges of teaching Python is that students are having heterogenic programming experience. Students may have experiences with different programming languages prior to taking a Python programming class. For instance, some students may have learned different programming languages, while other students may have never learned any programming language at all. This makes it difficult for lecturers to prepare teaching materials for students with differing levels of programming experience. The heterogenic background of students poses a challenge for lecturers to prepare class content or the syllabus of the course, which definitely has an impact on students' learning experience. Thus, it is important to investigate ways to manage the class to fit a wide range of students.

Wang and his colleagues (2017) have written a paper about teaching computer programming with Python for industrial and systems engineers. The paper basically illustrates the experiences of teaching and learning Python with an academic setting. It also shows some analyses regarding the learning preference of students with different background like gender, class standing, and attendance differences. For instance, Wang and his colleagues find that the learning performance is slightly different for female and male students. Yet, while Wang et al. solely provide statistics about the relationship between learning experience and different attributes of students, no teaching theory is proposed or examined. To extend this line of research on teaching Python, this preliminary study is to address these gaps by proposing and exemplifying an alignment of six teaching and learning strategies with the six levels of the Bloom's taxonomy of learning outcomes for teaching business students, with different computer programming background, Python programming at the University level.

2. THE BLOOM'S TAXONOMY
In this study, the revised Bloom's Taxonomy (2001) was applied to adopt a set of teaching and learning strategy for teaching business students Python at the University level in the Semester A of the academic year of 2019/20. The revised Bloom's Taxonomy is an ordering of cognitive processes and learning outcomes, which is based on earlier version of Bloom's Taxonomy (1956) created by Bloom and Krathwohl. Bloom's Taxonomy had been used as a guide in learning, teaching, and assessing learning outcomes for the past 50 years or so. It illustrates the cognitive path of learning from the beginning to a more advanced level of thinking with respect to the ordering of cognitive processes and learning outcomes. The Bloom's Taxonomy has also been a staple in teacher training and professional preparation, especially for a class of students with heterogenic background, addressed by this study.

Table 5. The proposed alignment of the six teaching and learning strategies with Bloom's taxonomy of learning outcomes.

Bloom's Taxonomy: Levels and Definitions (Anderson & Krathwohl, 2001)		**Proposed Teaching and Learning Strategies**	**Proposed Definitions**
Remembering	Memorize and recall learned materials like basic concepts, terminology, and facts.	Learn-by-typing (Mitamura et al., 2012)	Learn by typing the given codes to recall the learned computational thinking and programming concepts and syntaxes to complete simple programming tasks.
Understanding	Establish understanding of learned materials by comparing, translating, interpreting main concepts.	Learn-by-appreciating-examples (Guibert et al., 2004)	Learn by reading, appreciating and comparing the given examples of codes based on the computational thinking concepts.
Applying	Apply learned knowledge to tackle practical problems in certain situation.	Learn-by-modifying-open-sourced-codes (Saeed et al., 2011)	Learn by exploring and modifying the open-sourced and/or given codes to complete the computational thinking and programming tasks.
Analyzing	Break down information to identify and make inferences on relationship or causes of different factors.	Learn-by-partial-coding (Garner, 2002)	Learn by breaking a complex program into sub-programs (modules) and making use of the given partially completed codes to complete the complex computational thinking and programming tasks.
Evaluating	Make judgment and decisions after considering factors interfering the situation.	Learn-by-debugging (Lee, 2014)	Learn by evaluating flaws of the given codes and make corrections based on computational thinking concepts.
Creating	Gather ideas and information to propose valid alternative solutions.	Learn-by-problem-solving (Chao, 2016)	Learn by creating programs with designated purposes to solve problems or provide alternative solutions based on computational thinking concepts.

3. PROPOSED ALIGNMENT OF SIX TEACHING AND LEARNING STRATEGIES WITH BLOOM'S TAXONOMY OF LEARNING OUTCOMES

In the revised Bloom's Taxonomy, six cognitive processes/learning outcomes are identified, including ***remembering***, ***understanding***, ***applying***, ***analysing***, ***evaluating***, and ***creating***. In this study, we propose and exemplify an alignment of six levels of the Bloom's taxonomy of learning outcomes with six teaching and learning strategies for teaching business students computational thinking and Python programming at the University level. The alignment table is illustrated in Table 1. The concept of the proposed alignment will be illustrated by giving an example of the learning and assessment task in regard to each of the teaching and learning strategies in the following sub-sections. The given examples are adopted and modified from a textbook of the python course (Schneider, 2016).

3.1. Remembering: Learn-by-typing

In this paper, *learn-by-typing* is defined as learning by typing the given codes to recall the learned computational thinking and programming concepts and syntaxes to complete simple programming tasks. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

Type the following lines of code and run to determine the output.

```
listA = [5, -3, 6, 33, -10]
listA.sort()
print (listA)
```

In general, students are required to type out codes and display the output. This teaching and learning strategy is appropriate for students with no computational thinking and programming experience.

3.2. Understanding: Learn-by-appreciating-examples

In this paper, *learn-by-appreciating-examples* is defined as learning by reading, appreciating and comparing the given examples of codes based on the computational thinking concepts. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

Identify the pros and cons of the following two sets of codes with the same expected output.

Expected output:

```
0123456789012345678901234567890123456789012345678
9
week no  event \ holiday                    date
2        day following mid-autumn festival  14/09
5        national day                       01/10
5        graduation date                    02/10
6        chung yeung festival               07/10
```

First set of code:

```
print ("0123456789"* 5)
print("{0:<9s}{1:<36s}{2:>5s}".format("week no",
"event\holiday", "date"))
print("{0:^9s}{1:<36s}{2:>5s}".format("2", "day
following mid-autumn festival", "14/09"))
print("{0:^9s}{1:<36s}{2:>5s}".format("5",
"national day", "01/10"))
print("{0:^9s}{1:<36s}{2:>5s}".format("5",
"graduation date", "02/10"))
print("{0:^9s}{1:<36s}{2:>5s}".format("6", "chung
yeung festival", "07/10"))
```

Second set of code:

```
print ("0123456789"* 5)

print ("week
no".ljust(8),"event\holiday".ljust(33),"date".rju
st(7))

print('2'.center(7), '  day following mid-autumn
festival'.ljust(35), '14/09'.rjust(8), sep="")

print('5'.center(7), '  national day'.ljust(35),
'01/10'.rjust(8), sep="")

print('5'.center(7), '  graduation
date'.ljust(35), '02/10'.rjust(8), sep="")

print('6'.center(7), '  chung yeung
festival'.ljust(35), '07/10'.rjust(8), sep="")
```

In general, students are required to appreciate and compare given sets of codes to identify their pros and cons. This teaching and learning strategy is appropriate for students with limited computational thinking and programming experience.

3.3. Applying: Learn-by-modifying-open-sourced-codes

In this paper, *learn-by-modifying-open-sourced-codes* is defined as learning by exploring and modifying the open-sourced and/or given codes to complete the computational thinking and programming tasks. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

Write a program that requests a person to input his/her first name, last name, hourly rate and number of hours worked in Company ABC. Then the program calculates and displays person's gross exactly same output as below:

```
Enter your first name: Tai Man

Enter your last name: CHAN

Enter hourly rate: 55

Enter number of hours worked: 40

The gross pay for Tai Man CHAN: $ 2,475.00
```

Tips: *Please modify the function given below for calculating the gross pay in Company ABC that employees should be paid "time-and-a-half" for work in excess of 30 hours in a week.*

The function for calculating the gross pay in Company DEF, paying "time-and-a-half" for work in excess of 40 hours in a week is given below:

```
def calGrossPay(rate, hours):

  if hours <= 40:

    grossPay = rate * hours

  else:

    grossPay = (rate * 40) + (1.5 * rate * (hours
- 40))

return grossPay
```

In general, students are required to modify the given set of codes (acts as open-sourced and/or given codes), and complete the program. Thus, students do not have to spend too much time on writing the entire program from scratch. This teaching and learning strategy is appropriate for students with limited computational thinking and programming experience.

3.4. Analyzing: Learn-by-partial-coding

In this paper, *learn-by-partial-coding* is defined as learning by breaking a complex program into sub-programs (modules) and making use of the given partially completed codes to complete the complex computational thinking and programming tasks. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

There are missing lines of code in the following program, please fill in the missing lines of code to complete the program with no errors.

```
## totalScore.py

def aboutSystem():

    print ("This program calculates your total
score and letter grade.")

    print ("Please input your mid-term, and
final-exam score.")

    print ("This program is made by CHAN Tai Man,
12345678")

## Task 1: Please add a line of missing code here

    midterm = float(input("Enter your mid-term
score: "))

    ## Task 2: Please add a line of missing code
here

    totalScore = midterm*0.3 + exam*0.7

    return round(totalScore,2)

## letterGrade.py

def getLetterGrade(total):

    if total >= 90:

        return "A"

    elif total >= 80:

        return "B"

    ## Task 3: Please add a line of missing code
here

        return "C"

    elif total >= 60:

        return "D"

    else:

        return "F"

## getYourGrade.py

from totalScore import aboutSystem

from totalScore import getTotalScore

## Task 4: Please add a line of missing code here

aboutSystem()

total = getTotalScore()

letter = getLetterGrade(total)


## Task 5: Please complete the missing code below

print ("Your total score is " + _______ + ", and
your letter grade is "_________ "." )
```

In general, students are given a set of incomplete coding and were asked to fill in lines of codes or fill in the blanks to

complete the program. This teaching and learning strategy is appropriate for students with considerable computational thinking and programming experience.

3.5. Evaluating: Learn-by-debugging

In this paper, *learn-by-debugging* is defined as learning by evaluating flaws of the given codes and make corrections based on computational thinking concepts. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

In the following lines of code, identify all errors.

```
line = ("The", "only", "way", "to", "do",
"great", "work", "is", "to", "hate", "what",
"you", "do")

line[9] = "love"

print (" ".join(line))
```

In general, students are asked to find out flaws and error of the codes provided. This teaching and learning strategy is appropriate for students with considerable computational thinking and programming experience.

3.6. Creating: Learn-by-problem-solving

In this paper, *learn-by-problem-solving* is defined as learning by creating programs with designated purposes to solve problems or provide alternative solutions based on computational thinking concepts. An example of the learning and assessment task in regard to this teaching and learning strategy is shown below:

Mr. Lee just started his own business with very limited budget. Although it is a small store, he has lots of products needed to be managed. Without a store management system, it is very difficult for him to keep track on his product in store and carry out any stock control. Yet, he does not have spare money to purchase one. To help Mr. Lee to solve this business problem, you are asked to create a program using Python that can perform basic store management function, including creating invoice table in a database file, insert data into the invoice table in a database file, make query and request information corresponding to certain criteria. The entities and the data types should be included in the system are shown in the table below.

Columns	Data Type
InvoiceID	INTEGER PRIMARY KEY (AUTOINCREMENT)
ContractNo	INTEGER FOREIGN KEY (ContractNO of the Contract Table)
InvoiceDate	DATE
BillingAddress	CHAR (100)
Amount	FLOAT

In general, students are asked to solve a business problem by using computational thinking and programming skills. This teaching and learning strategy is appropriate for students with rich computational thinking and programming experience.

4. FEEDBACKS FROM STUDENTS AND INSTRUCTORS

All six teaching and learning strategies were addressed and demonstrated through examples from the learning and assessment tasks given to students of the Python course in the Semester A of the academic year of 2019/20. After the semester was ended, we collected feedbacks from both business students and instructors about the Python course. Some of the comments were captured and shown in the following subsections.

4.1. General Comments from Students

Some feedbacks are captured from the students of the Python course via an e-learning platform and presented in the following:

- I like this course as it provides a basic knowledge of Python, which help me understand how python works.
- I can catch up the lesson because of the uploaded examples and exercise. It is easy for me to follow the class. I think the examples, exercise and assignment are really useful for me to understand the chapter.
- Also, source codes are given to us, so we do not have to work from scratch, but to understand how to apply the programming languages to different scenarios.
- More actual examples and application of alternating items in a text file and analyzing the Data in a CSV File with a List as personally they are the most challenging sections in the course, but they are useful and essential skills applied on workplace.

4.2. General Comments from Instructors

Some feedbacks are captured from the instructors of the Python course via an interview, and presented in the following:

- Students were from a wide range of programming experiences. Some students had rich experience in other programming languages and struggled to accommodate the syntax that they learned in other programming courses to Python programming syntax. Examples and open-sourced codes help students to accommodate in using Python programming language.
- Students appreciated practical examples and scenarios that can solve problems or facilitate works for people in business settings. The assignments for problem solving also showed how students utilize open source code, acquired programming knowledge, and their creativity to provide alternative solution for the situations.
- At the beginning of the course, students needed more time for each assignment, even for those who had some programming training prior to the course. But as the course goes on, students with experience in programming started to overcome their legacy, they tried to help students who are new to programming. As students begin to help each other, collective programming happens which lowers the workload and burden from the teaching assistants' perspective. Time used for each assignment significantly decreased.

- A fixed marking scheme is preferred at the beginning of the course as to ensure students to learn the correct syntax of Python. Yet, after students get used to writing programming language, especially for those students with previous programming experience, they tried to combine or implement what they have learned in previous programming courses to the Python class, which leads to unexpected learning outcomes. Thus, fixed marking schemes might not be applicable at this point of the course.

5. CONCLUSION AND FUTURE WORK

To conclude, the main contribution of this paper is to propose and exemplify an alignment of a set of six teaching and learning strategies with the six levels of the Bloom's taxonomy of cognitive processes / learning outcomes (Anderson & Krathwohl, 2001) for teaching Python programming for business students (with different computer programming background) at the University level. University lecturers could use these six proposed teaching and learning strategies as a guideline to design their course contents and materials for teaching Python in the University level.

In this paper, feedback from both instructors and students are captured. Most of the comments are positive towards the proposed teaching and learning strategies, which indicated that the teaching and learning strategies are useful for better students' learning experiences, especially for those without computer programming background.

For the future research direction, empirical studies with large sample size and more robust measurement are suggested for examining the effectiveness of the six proposed teaching and learning strategies of teaching Python programming for students of different majors and computer programming backgrounds.

6. REFERENCE

Anderson, L. W., Krathwohl, D. R., & Bloom, B. S. (2001). *A taxonomy for learning, teaching, and assessing: a revision of Bloom's Taxonomy of educational objectives (Abridged ed.)*. New York: Longman.

Bloom, B. S., & Krathwohl, D. R. (1956). Taxonomy of Educational Objectives: The Classification of Educational Goals. *Handbook I: Cognitive Domain*. NY, NY: Longmans, Green.

Chao, P. Y. (2016). Exploring Students' Computational Practice, Design and Performance of Problem-solving through a Visual Programming Environment. *Computers & Education, 95*, 202-215.

Garner, S. (2002). COLORS for Programming: A System to Support the Learning of Programming. *Proceedings of Informing Science*, 533-542.

Guibert, N., Girard, P., & Guittet, L. (2004). Example-based Programming: A Pertinent Visual Approach for Learning to Program. *Proceedings of the Working Conference on Advanced Visual Interfaces, Gallipoli, Italy*. ACM, 358-361.

Lee, M. J. (2014). Gidget: An Online Debugging Game for Learning and Engagement in Computing Education. *Proceedings of 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Melbourne, VIC, Australia*. IEEE, 193-194.

Mitamura, T., Suzuki, Y., & Oohori, T. (2012). Serious Games for Learning Programming Languages. *Proceedings of 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Seoul*. IEEE, 1812-1817.

Saeed, T., Gill, H., Fei, Q., Zhang, Z., & Loo, B. T. (2011). An Open-Source and Declarative Approach Towards Teaching Large-Scale Networked Systems Programming. *ACM SIGCOMM Education Workshop*, 1-5. ACM.

Schneider, D. (2016). *Introduction to programming using Python, 1st edition.* Boston: Pearson.

Wang, Y., Hill, K. J., & Foley, E. C. (2017). Computer Programming with Python for Industrial and Systems Engineers: Perspectives from an Instructor and Students. *Computer Applications in Engineering Education, 25*(5), 800-811.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Teaching Computational Thinking to Applied Science Majors: What and How

Li XU
College of Applied Science and Technology
University of Arizona, USA
lxu@u.arizona.edu

ABSTRACT

Since Jeannette Wing proposed Computational Thinking (CT) as a fundamental skill to everyone (Wing, 2006), CT has become a phenomenon. In addition, it has been verified by program accreditation and employer requirements that undergraduate students in STEM need to develop higher-order thinking and metacognitive skills in problem solving. Thus, in our institution we intended to teach CT to students in Applied Science majors and support them to master the CT skill. While developing a CT course, we noticed that there was little agreement on what and how to teach CT. In this paper, we examine the CT course and provide a review that addresses two questions: 1) What to teach CT and 2) how to teach CT effectively. More specifically, we present the course topics covered in the CT course and describe six teaching strategies we utilized to engage students in learning and doing CT. While analyzing the course development reflectively, we become informed to continually improve the course in order to teach CT effectively in future.

KEYWORDS

computational thinking, course development, Applied Science majors, problem solving, student-centered learning

1. INTRODUCTION

Undergraduate students in STEM need to develop higher-order thinking and metacognitive skills in problem solving, which is verified by program accreditation documents and employer requirements. In addition, since Jeannette Wing proposed Computational Thinking (CT) as a fundamental skill to everyone (Wing, 2006), CT has become a phenomenon. According to Hu (2011), CT is present not only because of the nature of computation but also because of the way how people think critically—people gain different kinds of critical thinking capabilities through variety of means in CT. In our institution, we intend to teach and promote CT explicitly, and believe that every student in Applied Science disciplines such as Informatics, Cyber Operations, and Network Operations must master the CT skill. In the Applied Science programs, students can use the CT course as a critical thinking course to meet their Bachelor degree requirement.

How to support students to develop the CT skill? Research works done on thinking processes convinced that thinking skills were most effectively taught when teaching them directly and deliberately (Bono, 1992). Guzdial (2008) also pointed out "the metaphors and ways of thinking about computing must be explicitly taught." To exploit the idea to teach CT explicitly, we developed a CT course and offered it to students in the Applied Science programs. By viewing CT as a skill in general, we intend to teach CT by supporting students to acquire CT as competencies over time with practice but not facts or information compiled during the student learning process.

While developing the course, we found that even though CT had drawn a lot of attentions and become a popular subject, there was little agreement on what should be taught and how to teach CT effectively. For our CT course development, we designed the course by investigating literatures and resources on CT as well as the prior skills and knowledge of students who we intended to teach and support. Especially, in our approach we used Kansanen's didactic triangle (Kansanen, 1999) as a framework to design and evaluate the course content, considering what and how students would learn, what instructor's roles would be, and how students, instructor, and course content should work together using a student-centered approach to deliver the course.

In particular, to engage students into the teaching/learning process, we applied the *preference matrix* focusing on the two key dimensions including "make sense" and "get involved" to develop the CT course. The preference matrix method is based on an observation (Paxton, 2006): If an individual can "make sense" of and "get involved" in the course learning environment, the individual prefers the environment and then it is likely that the person will spend time within the environment; As a side effect of "make sense" and "get involved", learning will take place, which leads the individual to function effectively and have a productive learning. Moreover, we strongly believe that students are able to acquire the CT skill through hands-on projects. Therefore, we utilized problem-based learning (PBL) to engage students with hands-on projects, and students actively involved in doing CT practice persistently during the course delivery terms.

To summarize what and how we did, we centered our teaching on engaging and supporting students so that students conducted their learning by solving problems in multiple projects throughout every course delivery term. In parallel with the problem-solving activities, the course supported students to direct a self-regulated learning that refers to "the process whereby learners personally activate and sustain cognitions, affects, and behaviors that are systematically oriented toward the attainment of personal goals" (Zimmerman & Dschunk, 2011). Additionally, the course utilized writing, which provided one of the best ways to help learn the active, dialogic thinking skills according to Bean and Weimer (2011).

In this paper, we examine the course development and focus on addressing two questions: 1) What should be taught in order to support students to develop the CT skill, and 2) what are the effective teaching strategies, i.e., how we can teach and promote CT to the Applied Science majors effectively during the learning process. While developing the course,

we have persistently and reflectively touched on the two questions. Section *Course Topics* presents the covered CT topics when delivering the course in our institution. Section *Teaching Strategies* focuses on two aspects: 1) Practice CT skills by solving problems; and 2) explicitly guide learners to promote meta-cognitive awareness and conduct guided learning on CT. Section *Findings* presents four course deliveries by an instructor and discusses the impacts of course topics and teaching strategies on student learning. Finally, Section *Conclusions* concludes the study.

2. COURSE TOPICS

Among the CT literatures, we couldn't find a clear-cut definition of CT. In this paper, we highlight a definition Wing presented in a later paper (Wing, 2011), where she defined CT as "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be carried out by an information-processing agent." The CT course development was based on the fact that CT uses a set of concepts drawn from Computer Science (CS) to solve problems and design systems. To help students to understand and practice CT, we designed the course for students to develop a foundation of CT concepts and techniques, practice the various CT tools, and eventually synthesize them in critical thinking and problem solving.

When developing the course, we didn't intend to come up with an ultimate definition of CT to students. Instead, we explored various definitions and guided students to identify recognizable CT concepts such as abstraction, simulation, and algorithm design. While introducing multiple CT definitions, we highlighted Wing's arguments and definitions on CT so that students could see how the definitions, concepts, techniques and tools are related and put together. More specifically, we proposed a list of course topics including introduction to computational thinking, algorithm design, programming languages, data abstraction, programming in Python, thinking Object Oriented (OO), abstraction, simulation, shell programming in UNIX, and theory of computation.

First, we started the class with the instruction topic to allow students develop insight on what is CT, what are available CT definitions by researching CT literatures in ACM digital library and other online resources addressing CT. Students compared, analyzed, and identified the concepts and skills between the CT definitions and from what aspects people think about CT. After the introduction topic, student learned algorithm representation and creation in pseudo code that was written in Python style. (Python was used as the primary programming language in class.) After the algorithm topic, students learned variables and expressions, control structures, programming paradigms, and data structures in Python and bash. While students were acquiring knowledge on the essential concepts and techniques in programming languages, they also utilized and practiced programming to explore meanings of the CT concepts as well as problems such as Caesar cypher coding and random walking. Later, students further studied how to think in terms of objects, form communities by putting the objects/agents to act together, and design systems based on system behaviors and agent responsibilities. While exploring OO programming in Python, students used Python code to conduct simulation, and analyzed the steps of a simulation study. In addition, student studied theory of computation to understand what computers can do and what they cannot do in practice.

Through the course, we intended to support students to define and identify CS terms and concepts in CT; analyze and estimate what and how computers do; program operations in at least two programming languages (Python and bash); and apply CT to solve problems and design systems in practical applications. Among the topics, we emphasized concepts including algorithm, programming, and abstraction in a problem-solving context. When approaching problems, students needed to apply abstractions and make transitions among the different levels of abstractions. Students learned to use, analyze, and create algorithms by applying tools such as decomposition and generalization along with others such as planning and evaluations.

We introduced programming quite early in the course so that students were able to use programming as a vehicle to practice CT rigorously. Through programming, students were able to realize the power of computing by bridging the gap between informally expressed problems and formal solutions. They learned to invent formalisms by coming up with operations they designed and implemented. While programming, students approached to write procedures and functions in imperative program modules and later moved to program objects and classes using OO programming paradigm.

Note that in the course development we viewed CT as a skill rather than a set of knowledge facts. Such view was remarkable to guide our course development when we were deciding how to assess student learning while addressing the various concepts, techniques, and tools. We believe the course topics must be relevant and make sense to students regarding CT, and the CT skill must be acquired and constructed while students are doing CT and deeply involved in the learning process. Therefore, we carefully designed the learning assessment focusing on skill acquisition and CT development among students. To accomplish the learning goals, we used quizzes, online discussion, and programming/writing assignments. In particular, we included a final project where students needed to solve a problem.

3. TEACHING STRATEGIES

To effectively teach CT, we employed multiple teaching strategies to build a student-centered learning environment focusing on problem solving and guided learning with student self-awareness.

3.1. Problem Solving and Skill Construction

According to Lu and Fltscher (2009), CT provides a conceptual way to "systematically, correctly, and efficiently process information and tasks" to solve problems. We argue that CT is a skill that students acquire so that they can think like computer scientists to approach problem solving. Even though problem-solving skills are not specific to CT, as John

Dewey (1916) rooted critical thinking in the students' engagement with a problem, we recognized that problem solving was relevant to engaging and promoting CT, and intensively employed problems to stimulate thoughts and inspire learning while developing the CT course.

3.1.1. Strategy 1: Scaffold with Progression Model

During the learning process, we guided students to learn using a progression model composed of three steps: *use*, *modify* and *create*. We intended to use the model as a pattern of engagement to support student learning and maintain a level of challenge while avoiding too much learning anxiety. To practice a tool such as data abstraction, students used data structures such as arrays, lists, and dictionaries, to approach pre-defined tasks including file processing and behavior simulation. Then, we provided code that approached a problem with an incomplete solution. Students needed to modify the given code, trace execution steps, and empirically explore data structures being practiced in order to approach a complete solution. For the last step of the progression model, students needed to create customized data structures while approaching a problem. When designing the course content, we carefully conducted scaffolding the course materials to support student learning using the three-step progression model.

3.1.2. Strategy 2: Break Down and Synthesize

To align with the root of CT in problems, while introducing CT to students during the first topic, we referenced and shared the operational definition of CT introduced by International Society for Technology in Education (ISTE). The definition defines CT as a problem-solving process with characteristics including: formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking; identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; and generalizing and transferring this problem-solving process to a wide variety of problems. The operational definition provides a breakdown of CT skills for both the instructor and the students to identify and connect the key concepts and means in CT. Our objectives to teach CT consist of the acquisition of the ability to apply the CS concepts and techniques flexibly and creatively in a variety of contexts and situations. The course intentionally introduced the means and tools in CT such as algorithms, data structures, abstractions, thinking Object Oriented, and programming so that students were equipped with tools when they were approaching problems designed in the course assignments and the final project. The set of assignment problems was well structured and designed to promote learning in purposeful and engaging activities. The final project was to support students to synthesize their learning on CT and transfer the CT skill to problem solving.

3.1.3. Strategy 3: Abstract to Solve Problems

While referencing the ISTE operational definition of CT, the course development focused on the core CT skills identified by Selby and Woollard (2013), including abstraction, algorithmic thinking, decomposition, evaluation, and generalization. According to Kramer (2007), abstraction is the key to computing. In the CT course, we guided students to explore how to use abstraction to model problems and create solutions. To highlight the concept, we explicitly taught abstraction as a topic after introducing procedural and algorithmic thinking. In addition, when exploring CT from multiple aspects, students experienced practicing multiple-level abstractions with other tools such as programming and simulation. We also followed what Hazzan (2008) suggested that we should educate students to move between abstractions consciously. In particular, in our course development, we applied instructional scaffolding strategies to teach the various levels of abstractions involved in CT including data representation, procedures, objects, and problem solving.

3.2. Guided Learning and Self-Awareness

According to Kaplan & Kaplan (1983), the single most effective step one can take in improving the process of sharing knowledge is understanding and respecting the cognitive requirements of the intended recipient. The CT course development supported learners to promote metacognitive awareness, and built multiple channels for students and the instructor to interact and facilitate the student-centered learning process.

3.2.1. Strategy 4: Set Up Learning Goals and Objectives

While designing the CT course, we were aware that students needed to be coached to become self-regulated learners. The CT course development carefully presented the learning goals for students to accomplish from the beginning and throughout the course term. For each learning topic such as algorithm or programming, there were learning objectives and activities explicitly instructed to students. During the learning process, we used the course goals and module learning objectives to support students to monitor and assess their learning persistently. At the beginning of each course term, we informed every student and expected him or her to be proactive and reflective. While the student was progressing the learning process, he or she needed to constantly evaluate instructor/peer feedbacks and comments as well as learning performances, and gradually the student adjusted his or her learning approaches to master the CT skill, and developed self-regulated learning skills on thinking computationally.

3.2.2. Strategy 5: Engage to Read Critically

To effectively approach each subject covered in the CT course, students needed to read critically to gain essential conceptual knowledge and comprehension. Additionally, we aimed to support students become engaged readers on the CT topics. Our reading-engagement models emphasized on students' motivational beliefs such self-efficacy, interest, and value (Guthrie & Humenick, 2004). First, the course had a required textbook to cover algorithm, programming languages, data structures, and theory of computation. For more practical subjects such as programming in Python and bash, we provided hands-on notes and an online interactive book to guide reading and practice programming. In addition, we provided optional reading materials including podcasts, videos, and Voice Thread slides available online. To make sure students get involved in reading, we utilized

reading-quizzes, practice assignments, and online discussions.

3.2.3. Strategy 6: Write Reflectively and Persistently

Hazzan (2008) suggested conducting reflections and stated that reflection "increases one's awareness of the objects with which one thinks, and may therefore systematically and consciously lead one to think …" We exploited writing and reflections as two primary means to guide students to deepen their understanding on the CT concepts and develop the CT skill iteratively. To implement the writing strategy to teach CT, we deliberately required students to conduct weekly reflective writing to recognize, evaluate, and refine their learning on CT as well as stages of problem solving. To guide the weekly reflective discussion, we designed a set of scaffolding online-discussion questions with the expectation that student would write and unfold the computational concepts that form the foundation of CT. Moreover, in their reflective writing, students described their learning state and provided details for the instructor to monitor student learning.

4. FINDINGS

The CT course developments aimed to support a student-centered, participatory approach to teach and learn CT skills. We present what we found in the below subsections.

4.1 Course Deliveries

The course was initiated in 2013. Since then, we offered the course annually and in 2017 we started to offer it two times each year. In this paper, we would like to discuss the most recent four course deliveries offered by one single instructor. Table 1 presents the overview of the four course deliveries. In spring 2016 and spring 2017, we offered the course using 16 weeks. In spring 2016, 26 enrolled the class, one dropped, and one failed to pass it. In spring 2017, 31 enrolled, two dropped and two failed the course. In fall 2017, we offered the same course within 7.5 weeks. There were 31 students enrolled, one dropped, and two failed. In summer 2019, the course was offered within 7.5 weeks. There were 15 students enrolled and one student failed. Based on the Teacher-Course-Evaluation (TCE) reports collected by the end of each term, the teaching effectiveness is 4.65 over 5 in spring 2016, 4.32 in spring 2017, 4.65 over 5 in fall 2017, and 4.57 in summer 2019. The TCE numbers are positive to indicate that our teaching on CT has been effective.

Table 1. Overview of Course Deliveries

	Enrollment/ Dropped/Failed	No. Of Weeks	TCE
SP 2016	26/1/1	16	4.65
SP 2017	31/2/2	16	4.32
FA 2017	31/1/2	7.5	4.65
SU 2019	15/0/1	7.5	4.57

Since we employed programming as the primary means to carry out abstraction and automation while students were practicing the CT skill, we asked student input at the beginning of each term so that we were aware of their prior knowledge and experiences on programming. Due to a new and quickly growing Cyber Operations program developed in our institution, we've learned that more students enrolled in the course with little CS or programming experience. In 2016, about 40% of the students who enrolled the course had very little programming experience prior to the class. In spring and fall 2017, the numbers were about 60% and 75%. In summer 2019, only one of the 15 students had prior programming experience.

By monitoring student performance data and how students conducted their learning process, we observed that usually students were able to identify the CT concepts rapidly. For the reading quizzes, which we designed to assess how students understood the CT concepts, all 25 students who completed the course had passing grades (C or better) in spring 2016, one of the 29 students in spring 2017, two of the 30 students in fall 2017, and two of the 15 students in summer 2019 failed to pass the reading quizzes. For the online discussion component, which we employed to assess how students explained and applied the CS concepts in writing, only two of the 23 students didn't pass the online discussion component in spring 2016, four of the 29 students didn't pass in spring 2017, and four of the 30 students failed the online discussion in fall 2017. In summer 2019, one student failed online discussion.

In addition to analyzing student learning performance on reading quizzes and online discussions, we also investigated how students conducted CT to solve problems. Based on student learning performances on the assignment questions, which required students to apply and synthesize the means and tools in CT to address, we found three students in spring 2016, seven students in spring 2017, three students in fall 2017 and two students in summer 2019 failed to pass the assignments. In fall 2017, we started to provide a few more problem-solving hints on the coding assignments based on student questions and feedback comments we collected from students enrolled in spring 2017. The revision certainly helped students in fall 2017 to succeed their assignments. In summer 2019, we tried adding more programming components to support students practice Object Oriented (OO) programming and simulation, which followed a suggestion from the Cyber Operation program. The new added programming activities to employ OO programming paradigm certainly provided more practice for students to think OO and program simulation more rigorously. However, we also observed that the additional OO programming paradigm introduced in the short summer term generated more confusion between procedural and OO programming. And two students failed their programming assignments in the past summer.

Although students reported that practicing CT in the assignments and the final project increased their professional skills, it was obvious that students had difficulty on synthesizing the CT means and tools into their final project. In 2016, three of the 25 students who completed the course failed the final project even though two of the three students still completed proposing their projects and reported their progress on project development. In spring 2017, seven students didn't complete the final project but six of them completed their proposals and progress reports. In fall 2017, we delivered the course using 7.5 weeks, half of the time that we spent to deliver the course in the previous two spring terms. We found six students were not able to propose their projects and another six

students didn't submit their project posters. In summer 2019, based on our collected 7.5-week teaching experiences, we updated the final project by asking students to solve a single problem. In the revised final project, the program statement was provided and students needed to model the problem and implement their solution in Python. However, based on the final project submissions, the revision didn't improve student learning performance: only two thirds of the students created and implemented solutions to the problem, and the other five students failed to approach the final-project problem. Note that since the final project was the designated final exam, which contributed 15% of the overall grade, some students chose not to complete their final projects due to their busy schedules during the final exam period, especially if they felt satisfied with their accumulated grades. Thus, the performance data on the final project might be depressed to represent how students learned to employ CT to solve problems. Nevertheless, by analyzing the learning data and student comments in fall 2017 and summer 2019, we think that the shorter terms didn't work well as the longer ones for students to transfer their CT skill into problem solving while approaching the final project.

4.2 Discussion

Based on TCE reports and comments at the online discussion forums, students reported that they enjoyed and engaged in reading the course materials, and they liked how the course used the online discussions in conjunction with the assignments and reading quizzes to make all work together, and the online reflective writing contributed to establish a safe environment where students felt like they could be open and not get criticized. As they built the supportive, inclusive learning community, most students were willing to put more efforts to deal with the learning challenges even though they admitted several of the course topics could be overwhelming.

The course topics covered programming in Python and bash, which we essentially intended to provide two problem-solving contexts to tackle abstractions and automate execution of algorithms. Programming was a focal point in the CT course development to carry out important concepts and skills in authentic contexts of use. Even though students perceived programming as the most challenging subject, we observed that programming was engaging for students, especially for students who had little or none programming experience, to master as a means to express algorithms and accomplish abstractions and automations. However, we were also aware and let students well informed that programming and CT are not equivalent and programming is but one context for the practice of CT (Voogt, Fisser, Good, Mishra, & Yadav, 2015).

Note that we utilized programming in our course rightly after CT was introduced and students finished the topic algorithm. We delayed programming later than introducing CT so that students could acquire a bare model of CT first instead of being overwhelmed with programming and programming languages since the beginning. Considering the students in the Applied Science disciplines had various programming experiences and some of them had none, we were concerned that premature attempts to introduce programming with CT simultaneously could lead to confusions on understanding CT and failures to see the relevancy of the other course topics to CT. We believe such arrangement was fruitful---the writing reflections affirmed that students were able to understand CT and connect the CT skill to the various topics we practiced during each delivery term.

One critical learning component in the course development was the programming assignment part, which was designed based on Problem Based Learning (PBL). In PBL, it is common to give students a large ill-defined problem and let students figure out how to resolve it. Such practice is useful for students to practice tolerating ambiguities, to identify and formally define problems. However, to avoid overwhelming students, we carefully provided well-defined problems in each programming assignment so that the assignment problems were able to promote learning with purposes and challenges. Student learning performance was mostly positive while students were practicing CT in the assignments. The learning reflections and TCE comments also indicated that students were challenged and deeply engaged in resolving the problems computationally.

Nevertheless, for the 16-week deliveries and the first shorter-term delivery, we asked students to propose problems and create solutions in their final projects. We found that students had hard time to transfer the topics including programming into problem solving in their project development, especially when they were in charge of modeling their own problems of interest. In the most recent summer-term delivery, even with the provided problem statement describing a task to extract networking frame data, student performance data indicated that students were challenged significantly when they needed to synthesize various tools in CT as well as programming to create the problem solutions.

On the other hand, it was obvious, based on the student reflections and TCE comments, that the use-modify-create model helped student to make progress and acquire the CT skill and competencies gradually. Even for the shorter terms including fall 2017 and summer 2019, most students commented that the course delivery paced well. For students who had no prior programming experience or just returned back to school, their input including midterm surveys regarding learning progression was positive. However, since programming has been used as the primary means to express problem models and implement solutions, for students who hadn't done any rigorous programming before, creating a sound and complete solution to a problem was an intimidating challenge. Especially, based on the learning performances on the final project in the four course deliveries, we found that the complexity to move around the multiple-level abstractions when solving a complex problem and/or conducting a self-regulated project learning on the final project required time and practice for students to move around and gradually generalize and transfer the CT skill between problem contexts.

We found that the learning reflections we conducted were definitely helpful for students to retain their knowledge cognitively and ensure the whole learning make sense to students. During the learning process, the writing provided reliable, persistent learning traces for the instructor to

support student learning. The student reflections revealed the various backgrounds of the Applied Science majors. While reflecting what they were learning, students also brought up different knowledge and skill frameworks, which led to acquire the CT skill in different manners. Additionally, the learning reflections effectively involved both students and the instructor to be aware of the learning obstacles as well as critical issues to address. It was not unusual to see some students described that they couldn't continue due to certain programming bugs they confronted or they had no clue on how to approach a problem. In response to such reflections/questions, the instructor would guide their thinking, point out learning materials to refresh a review, and set up meetings to discuss the issues if necessary. Moreover, their peers often recommended problem-solving approaches or external materials/tips they found helpful. Last but not the least, since the course development required persistent learning reflections, writing became part of the systematic process for students to regulate and monitor their learning. Students became better communicators by transmitting and receiving messages clearly and reading the input from their peers and the instructor.

5. CONCLUSIONS

To draw our conclusions, we present and provide a review on a course development that intends to promote and teach CT to students in Applied Science disciplines. In particular, we address the questions including what and how to teach CT by identifying six effective teaching strategies. Our investigation focuses on the course content, students, and the instructor as well as relationships among the various learning components. Based on student learning outcomes and performances, we conclude that the course development is promising to engage and teach students to acquire CT as a skill to solve problems computationally. While we become more informed by analyzing and reflecting on the course development, we hope the course design and teaching strategies could be useful for our colleagues when they teach similar courses in their institutions.

6. REFERENCES

Bean, J. C, & Weimer M. (2011). *Engaging ideas: The professor's guide to integrating writing, critical thinking, and active learning in the classroom, 2nd Edition.* Jossey-Bass.

Bono, E. D. (1992). *Six thinking hats for schools*. Hawker Brownlow.

Dewey, J. (1916). *Democracy and Education*. New York: Macmillan.

Guthrie, J. T., & Humenick, N. (2004). Motivating students to read: Evidence for classroom practices that increase reading motivation and achievement. *The voice of evidence in reading research.* Baltimore: Brookes, 329-354.

Guzdial, M. (2008). Education: Paving the Way for Computational Thinking. *Communications of the ACM, 51*(8), 25-27.

Hazzan, O. (2008). Reflections on Teaching Abstraction and Other Soft Ideas. *SIGCSE Bull, 40*(2), 40-43.

Hu, C. (2011). Computational Thinking: What It Might Mean and What We Might Do About It. *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education: ITiCSE '11*. New York, NY, USA: ACM, 223-227.

Kansanen, P., & Meri, M. (1999). The Didactic Relation in the Teaching-studying-learning Process. *TNTEE Publications, 2*, 255-275.

Kaplan, S., & Kaplan, R. (1983). *Cognition and environment: Functioning in an uncertain world*. AnnArbor, MI: Ulrich's.

Kramer, K. (2007). Is Abstraction the Key to Computing? *Communications of the ACM, 50*(4), 36-42.

Lu, J.J., & Fletscher, G.H.L. (2009). Thinking About Computational Thinking. *ACM SIGCSE Bulletin, 41*(1), 260-264.

Paxton, J. (2006). The Preference Matrix as a Course Design Tool. *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*. New York, NY, USA: ACM, 124–127.

Selby, C., & Woollard, J. (2013). *Computational thinking: the developing definition.*

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational Thinking in Compulsory Education: Towards an Agenda for Research and Practice. *Education and Information Technologies, 20*(4), 715-728.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. (2011). *Computational thinking: What and Why.* Retrieved December 10, 2019, from http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why.

Zimmerman, B.J., & Schunk, D.H. (2011). Self-regulated learning and performance: An introduction and overview. In B.J. Zimmerman and D.H. Schunk, editors, *Handbook of Self-Regulation of Learning and Performance*. Taylor and Francis, New York, 1-12.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Developing Computational Thinking Through Tinkering in Engineering Design

Ashutosh RAINA[1*], Sridhar IYER[2], Sahana MURTHY[3]

[1,2,3] Indian Institute of Technology Bombay, India

raina.ashu@iitb.ac.in, sri@iitb.ac.in, sahanamurthy@iitb.ac.in

ABSTRACT

Artefact creation as part of constructionist approaches towards learning has seen an increase pertaining to the growth and ease of availability of design tools. Projects that involve artefact creation allows the learner to experience the problem solving process while being situated in a real-life context. Tinkering is one such approach to problem-solving. In this paper, we present a design of our tinkering intervention for teaching and learning of computational thinking. The intervention is a composition of four major components, namely the Pedagogy, Problem, Resources and Mentor. The proposed Explore-Solve-Evolve pedagogy incorporates aspects of constructionism, progressive formalisation, learning situated in a real-life context and immediate feedback for reflection. Lego Mindstorm is provided as a building resource, and an app seamlessly provides information about the resources. The mentor encourages the learners towards exploration and play with the resources in the problem space and scaffolds them with strategies to overcome challenges. A proposed study has been discussed to further understand the development of CT with tinkering. The paper is concluded with presenting the mapping between the phases of our intervention and the three dimensions of the CT framework.

KEYWORDS

computational thinking, tinkering, intervention, robotics

1. INTRODUCTION

Computational thinking has been defined as "The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Brennan & Resnik, 2012). Computational thinking has been taught not only through programming but also through activities like playing games, building a robot to solve challenges, creating e-textiles and range of activities that involve concepts of computational thinking. The idea is to be able to express yourself using computational artefacts which have been identified as an essential aspect of computational literacy. While developing artefacts, learners also deal with failure in physical components and compatibility issues that can be frustrating. However, they are an essential part of solving problems where one is often required to use of computational thinking, not limited to just writing code (which has been termed as the material aspect of CT). In addition to the material aspects of CT (which is the *how*), learning-environments that include artefact building as a part of the problem-solving process also focus on the social (which is the *where* and *whom*) and extends it to the cognitive aspects (which describe the *why*). Building artefacts to solve a given problem situates the problem-solving process in a physical context that is closer to an authentic scenario.

One such practice that includes artefact creation with problem-solving is tinkering. It has been considered as a novice and expert practice which sets it apart from most of the classroom practices (Danielak, 2014). It does not make tinkering better or worse but it does make it an authentic professional practice (Berland, 2016). Tinkering provides the opportunity to work in a realtime environment with immediate feedback on actions taken hence making it a potential means for developing computational thinking. We believe that tinkering with robotics kits like Lego Mindstorm provide a medium and opportunities for the development of computational thinking. We are interested in the ways that tinkering activities with programmable tangible robotics kits, like the Lego Mindstorm, can support the development of computational thinking in students in higher education which is highly dependent on learning of programming languages (Brennan & Resnick, 2012).

2. THEORETICAL BASIS

2.1. CT Framework

Computational-Thinking has further been classified into *CT Concepts* that learners develop while learning to program like loops, conditionals, sequences, parallelism, data structures, operators, event handling, procedures and initialisation. *CT Practices* that learners repeatedly demonstrate in the programming process like problem formulation, problem decomposition, abstracting and modularising, algorithmic thinking, reusing and remixing, being iterative and incremental, testing and debugging. *CT Perspective's* talk about the Learners' understanding of themselves and their relationships with others and the world of technology, also termed as Computational Identity (Brennan & Resnik, 2012). It also includes programming empowerment as well as provides a perspective of expressing, connecting and questioning with programming. The elements of CT as mentioned earlier in its three dimensions have also been included in the operational definition of CT for K-12 education by the International Society for Technology in Education and Computer Science Teachers Association (ISTE & CSTA, 2011).

2.2. Tinkering Practice

The growing availability design tools have led to a commitment to learning through design activities in a constructionist approach (Harel & Papert) to a level of learning that highlights the importance of young people engaging in the development of external artefacts (Kafai & Resnick, 1996). Besides, progressive formalisation (Bransford, Brown & Cocking, 2000) requires teaching to be designed to encourage students to build on their informal ideas in a gradual, structured manner that enables them to acquire the concepts and procedures of the discipline. Moreover Learning situated in a real-life context (Bransford, Sherwood, Hasselbring, Kinzer, & Williams, 1990) enables

a better understanding of abstract concepts by establishing there need in a real-life context using everyday examples. In addition to situated learning, play becomes an essential tool for learning in real-life context as it allows experimentation with the available resources and one's ideas in the actual problem space with just in time feedback that enables reflection. It also allows one to take multiple perspectives on an action and its impact, which is an essential social skill for the development of the mind (Bailey, 2002). Tinkering has been addressed to be at the intersection of all the above (Roque, Rusk & Blanton, 2013). A definition of tinkering calls it as a playful, experimental, iterative style of engagement, in which people are continually reassessing their goals, exploring new paths, and imagining new possibilities (Honey & Kanter, 2013). Here play has been referred to as experimental play. Tinkering provides a multitude of possible paths taken progressively while situated in problem space working with immediate feedback.

2.3. Explore-Solve-Evolve Pedagogy

Based on our synthesis from the literature on tinkering for problem-solving, we have identified a few operational aspects of tinkering as Exploration, Play and Reflection. Exploration is used to determine the affordances or *can do's* of the available resources and possible solution or *want to do* for the problem at hand. Play is used to determine if a solution could emerge by mapping the *can do's* and *want to do's.* Reflection is used to overcome states of *stuck* and *fixation* that arise due to unexpected contingencies (exception violation (Schank, 1983)) or failure. Using strategies like questioning, repurposing, reflective strategies on productive failure (Kapur, 2008) provide the means to overcome such challenges. Reflection on the tinkering trajectories to enable modification of understanding and learning about the problem space.

We used the above operational aspects along with tinkering frameworks like Spark, Sustain and Deepen (Honey & Kanter, 2013), and Think, Make and Improve (Martinez & Stager, 2013) to derive a three-phase pedagogy named Explore, Solve and Evolve for taking a tinkering approach to computational thinking. The features of free exploration to capture intrinsic motivation have been incorporated in the explore phase. Progressive formalisation has been implemented in all the three phases of explore, solve and evolve. In explore learners start with small problems situated in context robotics, which requires them to interact with the physical space using the components of the robotics kits to solve the problem. In the problem given in the solve phase allows the learners to build their solutions with small component problems solved in the previous phase. This method also allows the reuse and iteration of previous solutions. Finally, in evolve, the learners frame and solve a problem to advance the solution they develop in the solve phase. The learning environment comprises of building resources and some pre-build solution of similar problems.

We believe the features of the pedagogical design and the element of the learning environment based on tinkering which has been aligned to the operational elements of CT aided with an explicit reflection on the action will lead to the development of CT among the students. The problems that have been chosen align to the High school curriculum of various educational boards in India.

Table 1. Summary of the pedagogy with its mapping to available resources and activities to be performed.

Pedagogy	Problem	Resources		Activities	
		Building	**Information**	**Learner**	**Mentor**
Explore	Small problems that are a part of the challenge for the next phase. *E.g. build a chases with wheels.*	With the focus on use of basic individual resources and their affordances. *E.g. Connecting motors and the EV 3 brick.*	Using the AR component view from the app for affordances of the individual resources.	Interaction with resources while solving problems to understand their affordances.	Encourage exploration and play with resources
Solve	One open-ended challenge that is derives from problems of "Explore" phase with opportunities for reuse. *E.g. build a wheeled bot that can move and turn.*	With the focus on combined use of the resources and their interactions with each other. *E.g. Mounting the EV3 on the chassis and building the turning mechanism.*	Additionally, information about the interaction of different resources and available use cases. Scaffolds for techniques for getting unstuck	Determining the sub problems and primary functional modules. Use pre-built solutions from previous phase	Additionally, provide prompts and scaffolds for techniques like reflection and productive failure.
Evolve	Additional challenge to increase the complexity of the previous challenge requiring the need of abstraction modularization and iteration. *E.g. Make the bot avoid obstacles*	Use of additional complex resources to enhance capability of the current build. *E.g. Adding IR, Ultrasonic sensors and building a parallel process of obstacle detection.*	Similar as above	Frame the new problem, choose the sub problems and address the sub problems while using techniques to overcome challenges	Indirect guidance using instances from the previous phases.

3. INTERVENTION DESIGN

The Tinkering environment for learning with CT comprises of the problem whose potential solutions derives from CT. Available resources allow free exploration, have a low floor and high ceiling and align to the constructs of computational thinking. Both the problem and the resources ensure the requirement of tinkerability (Resnick & Robinson, 2017). The pedagogy encompasses features like progressive formalisation, alignment to intrinsic motivation, guided reflection. Finally, a mentor provides scaffolds for the use of strategies like re-purposing, question-posing and reflection for working with expectation violation and productive failure. A summary of the entire intervention is as presented in table 1.

3.1. Problems

Though any problem with its corresponding resources could be provided in a tinkering based learning environment, we choose Lego Mindstorm Robotics kit and design a maze that would have to be solved as a part of the activity. This activity provides enough freedom to the learners for designing the robot as per their choice to solve a given maze. Keeping progressive formalisation in mind the problems are divided into two categories. The first category of problems is toy problems that help the learners to explore the resources available in Lego Mindstorm and get used to them. E.g. one of the problems requires the learners to determine the volume of the room given the Lego Mindstorm EV3 brick and the ultrasonic sensor. The objective of this problem is for the learners to understand the usage of ultrasonic sensors and also to be able to build a quick prototype and use the data representation features of the EV3 brick. Additionally, they are being exposed to the concept of input and output of data using physical sensors, or what we call they are getting a sense of the kind of output the sensor can provide. Though this question requires them to work with the ultrasonic sensor, the mentor encourages them to use all possible actuators and sensors to get a sense of the devices. Similarly, one of the problems requires the learners to build a two-wheel powered bot and a four-wheel powered bot to determine the use cases of each configuration. These problems are candidate sub problems to the bigger problem that the participants will solve in the next phase.

In the second phase "Solve" we provide them with a maze that their bot has to navigate. The maze is an NxN matrix where obstacles have been places, and the bot must follow the unblocked edges and reach the destination. The learners are given the maze along with the edges that will be blocked. This problem becomes a standard path traversal problem where the learner must sequence a set of instructions, and the sequence would determine the path that is traversed by the bot. The length of the edges are standard; hence the learners must determine the distance the bot would move and code it accordingly. Though the length is the same distance would vary based on the bot they have built or the motor parameters they are using. Thought a hard-coded solution is not the ideal solution for this problem, the problem the idea is to take the learners through this journey to understand the different solutions and challenges they pose and evolve them towards building using constructs to build better / dynamic/efficient solutions.

In the third phase named "evolve," they are given a new challenge where they are to program and modify the robot in such a way that it could traverse the maze even if the obstacle locations have not been determined initially. They could add markers on the obstacles for the bot to identify and take action accordingly. The objective here is to allow the learners to understand the concept of functions and modularisation so their bot can take decisions based on the maker. This problem evolves the learners to thinking in terms of higher-order CT concepts while providing them with the freedom of incorporating their idea of how to implement them.

3.2. Resources

Resources in the learning environment refer to the components of the learning environment. These are divided into building resources and information resources. Building resources refer to raw building materials, fabricated building materials and electronic components. As an example, in our case, the building resources would consist of the Lego Mindstorm kit and a few other resources like tape cardboard etc. Further classification of the components could be done based on their nature of use and other characteristics.

Figure 1. Building Resources and Mobile Application

The information-seeking resource consists of repositories of information on a mobile application. The mobile app also has an interface to interact with the learning environment using Augmented Reality. The learners work in the problem space with the available tools and resources to find solutions to the problem at hand. Prior knowledge of affordances of tools and resources available for tinkering through a problem or ability to acquire such information in the time of need is a challenge for learners who intend to take a tinkering approach. Gathering this information from manuals and online resources frequently requires switch context, which inhibits or discourages explorations with the unknown components. Hence this app will enable the learners to seek information about problem statements, help them track their session, provide information about components. The app will have a different section for the different phases of the pedagogue. The app will also act as a platform where prompts and scaffolds will be presented. The apps also enable delivering just in time information by presenting information in an augmented manner to ensure seamlessness, as seen in Figure 2 below.

3.3 Pedagogy

The pedagogy has evolved from our explorations with tinkering and literature (Honey & Kanter, 2013) (Martinez & Stager, 2013). The initial motive is building curiosity into the mind of the learners by exposing them to various complex solutions and stories about solving them. The learners are guided to explore and play with the available

solutions to build their understanding of the environment. One of the intended ways of doing it is by starting with candidate subproblems of the main problem that they will be solving in the second phase. These subproblems are introduced as primary problems for exploration with simple resources to interact with and gradually increase the complexity of the problems and the use of resources.

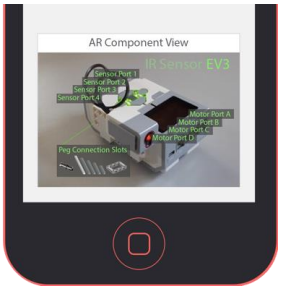


Figure 2. AR Component view of the Lego EV3 brick.

The motive is to encourage exploration of the resources for indented use. At the end of this activity, the students should have an understanding of the different components of the robot, their function, and how they can be arranged/combined to achieve a more significant function.

In the second phase, they are given a problem to solve within the same environment. Initially, the learners are mentored to find a starting point and then are left free to begin working on the starting point of their choice. Here the learners interact with the building resources based on their understanding from the previous phase of the problem. The disposition that learner should take is of experiencing what would happen than predict the outcome by observing or thinking about challenges. With practice, one may be able to predict the outcome by mentally experimenting with the problem space. This experience will later give rise to the needs of the solution or what kind of function/behaviour would be required by the solution. Another challenge they might face is of being stuck. Stuck is to be interpreted as the state when the participants are not able to ask the right questions. Being in one state but still being able to ask the right questions is still considered the state of flow. In the stuck state, the role of the mentor is to guide the participants to ask the correct questions. The app acts as the repositories of information about building resources and examples. Some necessary information maybe even augmented using the app on the resources for a quick understanding. The learns may record their progress on their app as a medium intended for logging. This can also be used by the learners to reflect and make decisions. The final part of this phase of the workshop is to enable reflection on the solutions the learners have built from the problem they were provided. The reflection would be triggered by posing questions regarding the requirements of the problem. The type of question to be posed. The learners will also be encouraged to use their logs to aid this reflection process. These reflections will be recorded by learns in the app. The objective of reflection is to make the explicit realisation of the CT elements and connect them to the activities performed by this. It ensures the development of an understanding of the use of CT as per the three domains.

This stage, the learners will evolve their solution to either enhance their capabilities or refine its function or performance. One of the objectives is to introduce them towards abstraction of primary function and their modularisation. Also, expose them to parallelism. In this phase, the mentor will be available only on demand as the mentor does not take an active role in the solution process. The purpose of the mentor would be to observe learners actions to monitor their approaching. The mentor may choose to intervene in some situations mentioned in the mentor's guide. The intervention would be limited to directing the learner's approach by asking broad questions. The objective of this phase is to determine if the learner can initiate explorations, make observations and ask questions about it. The mentor may choose to allow the learns to exit without building the refined solution if enough evidence of the objective is available. These will be available as guidelines to the mentor. In the final stage of the workshop, the mentor will trigger reflections among the learners on the entire actions to develop an understanding and use of the elements of CT.

3.4. Mentor

The Mentor is more of a facilitator to observe the activities and the process the students are following. The motive of doing this is to help them reflect on their actions. Guide them towards exploration and play. Guide them the overcome challenges by identifying the reasons. The reasons could vary from not being able to construct the intended the solutions, not being able to use the resources at hand, not being able to identify resources and/or the corresponding affordances or unpredicted behaviour. To direct the students to the flow state, the mentor themselves must become a genuine participant of the activity. They should try to figure out what is the problem. The mentor can probe using questions like what seems to be the challenge? What seems to be your approach? If the learner can answer mapping to solving a problem and changing the design, then the learner is actually in the flow state. To probe further, the instructors could explicitly ask "Which questions are you trying to answer?" if the participant shows signs of frustration or seems to have given up. These would be responses like I do not know what to do next, I have tried many things. It will not work. This cannot be solved. The Mentors could guide them by asking questions as stated above that would help them proceed with the approach. The participants could respond with answers that talk about the loss of interest or boredom like I am getting bored, and I do not feel interested in doing the same. I am not able to think more. The mentors could guide them towards skipping the current challenge and work on a different aspect or just ask them to take a break. If the mentor feels the participant is struggling due to lack of information, they may guide them towards the information. The objective is to make them realise that such information can be looked at.

The mentor should be able to take a multi-level view to weight between the more significant problem and the problem they are stuck. The criticality of the current problem for the more significant problem can help determine to solution approach. If it is critical, we need to find a way to work it out or if not, can we manage to solve the bigger problem without the problematic component at hand. The mentor should guide the students via open-ended prompts describing the behaviour of the component at hand. The

prompts should target misconception or refer to some other project and explaining the function of the component and have them try it. Another way of doing this is by posing questions starting with What are they trying to achieve? Why are they doing it this way? How will it achieve what they intend to achieve? When and where does this help to solve the bigger picture?

3.5 Proposed Study

The study is targeted at High school students who have just started using programming languages and do not have exposure to Lego Mindstorm robotics kits. This version of the study will be done with one individual per kit. We plan to introduce elements of collaboration in later studies. The objective of the study is to explore the use of tinkering as a strategy for learning elements of CT. Will the tinkering learning environment designed with an alignment of CT elements lead to explicit learning of CT in its three dimensions? This will us with a deeper understanding of the alignment and the features that may or may not work as intended.

The study is based on the Explore, Solve and Evolve pedagogy and distributed over three days. On the first day, the learners will be introduced to Lego kit using the candidate sub-problems. They start with problems to introduce them to the EV3 brick along with the sensor and the motor functions. Similarly, they will be given problems that lead them to explore the construction blocks and beams. The learners are allowed to dismantle a few prebuilt bots. In the final part of the day, they would be given problems that would require them to code, either the prebuilt bots or the bots they have built. The day would end with the mentor asking the learners about the kind of bots they would want to build and making them reflect on their observations and understanding of the building resources.
On the second day in the "Solve" phase, the learners would be provided with the challenge of solving a static maze. They could reuse the bots from the previous day or build new ones. At the start of this session, the learners will try to find out the essential requirement of traversing the maze. The bot will have to perform two functions which are moving on clear lines and turning to avoid obstacles. The mentors may lead the participants to play in the problem space to physically experience the problem by manually navigating the maze using a non-motorised bot. Once the participants have realised the essential functions, the instructor will facilitate the participants in realising the needs from the previous exercise and then try to translate them into functions and behaviour for their solutions. Once the desired behaviours have been achieved, learners can move forward to the next essential objective. Learners may perform as many numbers of trials on the maze and only when they determine or the time is done, they would require to demonstrate their solution. The learner determines the final demonstration beforehand. If the learner finishes before the time the mentor asks them to improve the efficiency in terms of time taken by the bot to complete the maze. The day ends with the mentors making the learners reflect on the solution trajectories. The reflection will be carried out through activities where the learner would be told about the CT elements, and they would map it to their solution strategies and later determine one use-case for their application.

On the third and final day called "Evolve," the learners are required to solve a similar maze, but they would not know where the obstacles would be placed. In this case, the obstacles would have a provision to place markers. The learner could use these markers to make the bot respond with a specific action like turning left or right. In this phase, the mentors will gradually reduce the scaffolds and prompt limited to making them recall things they did on the previous day, so they can make associations from what they learned. To increase the complexity of the problem, the standard length between the nodes may vary. The mentor facilitates reflection by having the learners talk about their experience and pointing out key actions they performed and having them articulate what they exactly did and what did they achieve. The mentor may ask learners to demonstrate the use of CT concepts that could be implemented if a given behaviour was to be achieved? Once the reflection session is over the learner are given scenario-based MCQ.

4. CT IN TINKERING

In this paper, we present the design of an intervention and a proposed study to explore the use of tinkering as a means for developing an operational level understanding of the different dimensions of CT. In the explore phase, activities that emphasise the interaction with the sensors and the EV3 brick help the learners to understand with programming is and empowers them with the opportunities of being able to program physical objects. Constructing small artefacts exposes them to CT concepts of operators, procedures etc. In the solve phase, the learners are introduced to sub-problem generation and encouraged to reuse and remix solutions from the previous phase adding a few more CT concepts. In the evolve phase, the learners are made to reflect on the iterative and incremental way of solving problems. The slight increase in complexity of the problem introduces them concepts of abstraction modularisation of the turning function. They also learn about parallel processing to achieve the motion and obstacle detection function. Table 1 below provides a summary of the mapping between the activities performed in the tinkering environment and operational elements of CT from the CT Framework. Table 2 also presents the distribution based on the three dimensional CT framework aligned to the essential phases of our tinkering pedagogy. We believe that by such an alignment of dimensions of CT with our tinkering pedagogue, the learners will be able to develop an operational understanding of using CT for solving problems.

Table 2. Activities done in different phases of the pedagogy and their mapping to dimensions of CT.

Phases	Activity	CT Concepts	CT Practices	CT Perspectives
Explore	Interaction of sensors with the environment Finding their affordances Making moving bots, right left turns Stopping and moving on obstacle	Operators, Procedures, Data structures	Problem Formulation, Questing	Programing empowerment, Perspective of expressing.
Solve	Use pre-built solutions from previous phase Determining the subproblems and primary functional modules	Sequencing, Event handling	Problem Decomposition, Algorithmic Thinking, Reusing Remixing	Connecting Questioning
Evolve	Using the learning from explore about sensors functions and bot motion Evolving the solution to a modular approach. Achieving obstacle detection while moving	Loops, Conditionals Parallelism	Iterative Incremental, Abstracting Modularising	Connecting Questioning

5. CONCLUSION

As present above, we proposed intervention for teaching computational thinking (CT) as a part of the high school curriculum. The first component of the intervention is problems that provide learners with opportunities to use CT. We have used problems with robotics. The second component of our intervention are resources to work with. We have chosen Lego Mindstorm and a few everyday materials for construction. Our application provides information about the resources textually, visually seamlessly using augmented reality. The third aspect of our intervention is that the Explore-Solve-Evolve pedagogy ensures a rich, authentic problem-solving experience for the learners. Reflections after each phase introduce the learners to the concepts, practices and perspectives of computational thinking. The mentor assumes the role of a noncontributing companion by scaffolding the learner towards exploration and play using strategies like question posing. They mentor learners with strategies to overcome challenges and reflection to ensure an explicit understanding of learns action.

The question that we pose to ourselves is that "Will the tinkering learning environment designed with an alignment of CT elements lead to the development of such an understanding of CT in its three dimensions?" Before we could aim at answering this question, this study will provide us with a deeper understanding of the alignment and the features that may or may not work as intended. With an evolved Tinkering enabled learning environment we plan to conduct more studies using techniques to evaluate the learning of CT as reported in the literature (Kong & Abelson, 2019) to be able to determine the impact of using a tinkering approach towards developing computational thinking.

6. REFERENCES

Bailey, R. (2002). Playing social chess: Children's play and social intelligence. *Early Years: An International Journal of Research and Development*, *22*(2), 163-173.

Berland, M. (2016). Making, tinkering, and computational literacy. *Makeology: Makers as learners*, *2*, 196-205.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn (Vol. 11)*. Washington, DC: National academy press.

Bransford, J. D., Sherwood, R. D., Hasselbring, T. S., Kinzer, C. K., & Williams, S. M. (1990). Anchored Instruction: Why we need it and how technology can help. *Cognition, education, and multimedia: Exploring ideas in high technology*, 129-156.

Brennan, K., & Resnick, M. (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 (1)*, 25.

Danielak, B. A., Gupta, A., & Elby, A. (2014). Marginalized identities of sense‐makers: Reframing engineering student retention. *Journal of Engineering Education*, *103*(1), 8-44.

Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Honey, M., & Kanter, D. E. (Eds.). (2013). *Design, make, play: Growing the next generation of STEM innovators*. Routledge.

ISTE, I., & CSTA, C. (2011). Operational Definition of Computational Thinking for K–12 Education. *National Science Foundation*.

Kafai, Y., & Resnick, M. (1996). *Constructionism in practice: Designing, thinking and learning in a digital world.* Routledge.

Kapur, M. (2008). Productive failure. *Cognition and instruction*, *26*(3), 379-424.

Kong, S. C., & Abelson, H. (Eds.). (2019). *Computational Thinking Education*. Springer.

Martinez, S. L., & Stager, G. (2013). Invent to learn: Making. *Tinkering, and Engineering in the Classroom*. *Torrance, Canada: Construting Modern Knowledge.*

Resnick, M., & Robinson, K. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*. MIT press.

Roque, R., Rusk, N., & Blanton, A. (2013). Youth Roles and Leadership in an Online Creative Community. *CSCL, (1),* 399-405.

Schank, R. C. (1983). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge University Press.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

A Comparison of Computational Thinking Approaches in HCI-SEO Design: Implications to Teaching and Learning STE(A)M

Chien-Sing LEE
Sunway University, Malaysia.
chiensingl@sunway.edu.my

ABSTRACT

Search engine optimization has often been through tagging (metadata descriptions) and appropriate placement of these metadata in inherent document structures e.g. XML. This paper presents a complement whereby the structure and information design based on design thinking and computational thinking results in more effective scoping of user requirements and leaner, agile design. This form of human-computer interaction-search engine optimization is much used in successful e-commerce websites due to Data Science. Comparison between the standard 4 CT aspects approach and Brennan and Resnick's 3 CT aspects approach and implications to STE(A)M teaching and learning are investigated through a meta-analysis of two Project Management course assignments. Significance of the paper is direct link and greater specificity between design thinking, computational thinking, human-computer interaction, Project Management and search engine optimization within an entrepreneurial project management framework.

KEYWORDS

design thinking, computational thinking approaches, design optimization, STE(A)M, higher education

1. INTRODUCTION

The trends in project management (PMI, 2017) highlight the need for first, entrepreneurial project managers who are able to think and decide not only quickly but also analytically and judiciously, by utilizing and managing frameworks and diverse decision support tools. This leads to judicious application of agile project management as well as hybridization of project management methodologies from different industries to promote different ways to build things and enhance processes and outcomes.

Analytical, judicious thinking and the ability to synthesize are characteristic of creative thinking (Arnold, 1959). Arnold's (1959) Theory of the Creative Process regards the creative process as:

a) applicable to several domains to a certain extent;
b) dependent on the processes a person follows;
c) a search and problem-solving process aimed at better meeting basic human needs;
d) influenced by meta-cognitive processes, which identifies and regulates creative progress.

Another two trends which are increasingly gaining attention are man-machine collaboration and gamification. If designed well, these can sustain e-commerce, supply chain and growth. Hence, there is a need to train students to design through modelling and computational thinking. The question is how to scaffold generative deep thinking?

1.1 Objective

Computational thinking (CT) commonly emphasizes four aspects (Figure 1a). A critical CT concern is also to link with real-life applications, scenarios. A real-life example of decomposition and algorithmic thinking (Olaf can rearrange parts of himself) in CT is in Figure 1b.

Figure 1a. Four key aspects *Figure 1b.* Example of CT

For this paper, implementations of the popularly accepted four CT aspects and Brennan & Resnick's (2012) 3 CT aspects: *computational concepts, computational practice and computational perspectives* are juxtaposed and the implications to teaching and learning are compared. For both case scenarios, project management knowledge areas are integrated within an entrepreneurial framework.

Both studies/systems aim to increase Search Engine Optimization (SEO), sustainability and interactivity. For the standard 4 CT aspects, we choose to focus on an e-commerce website that sells furniture, *Furnitize* (Chew, Chee, Wong, Hiew, 2017). Patterns (templates), with decomposition (parts of objects), abstraction (different levels of details) and algorithmic thinking (processes to create the simulated desired interior) using the software. For Brennan and Resnick's CT aspects, we choose an e-commerce-crowdsourcing recycling website, *The Enchantress* (Yew, Lim & Sugumar, 2017), which questions how we define fashion, diverse perspectives of fashion design as well as entrepreneurial possibilities.

2. RELATED WORK

In this section, we present the design factors considered. To scaffold goal-based contextual thinking, goal-based scenarios (GBS) proposed by Schank, Fano, Bell and Jona, (1993) recommends the use of mission as overriding goal. The mission can be reflected in themes and these can be adapted into different cover stories with variations in situations, roles and challenges. These cover stories consequently, result in interrelated smaller missions. This is necessary to mediate from easy to difficult situations, roles and challenges.

Schank, Fano, Bell and Jona's (1994) GBS finds support in design thinking. Design thinking focuses on context, empathy and user experience as starting points. As a Human-Centered Design methodology, design thinking incorporates

consumer insights as the first design space (Dym & Little, 2003). Apple is a representative example of systemic solutions, partly emotional and partly cognitive, within knowledge-based ecosystems.

Brennan and Resnick's (2012) CT aspects are concepts, practice and perspective. Examples of concepts are events, conditions, sequence, and loops. These are similar to information system's conceptual schema, conditions, data flow. Practice in incremental improvement and testing are akin to pilot, alpha-beta testing, reusing and remixing strategies/assets. Practice in abstracting and modularizing is pattern-based. Perspectives are expressive, connecting and questioning, to encourage meaningful iterations. These researches point out that more needs to be understood in terms of how design and computational thinking helps to develop creativity among designers, in higher education.

3. METHODOLODGY

The students are not Computer Science students. Hence, Brennan and Resnick's (2012) perspectives is first utilized. Students are asked to identify which current trends and issues in project management they find interesting from the PM Institute's Pulse of the Profession (2017) report. Project Management and HCI concepts are integrated with information systems analysis and design (ISAD) constrained by impact on society and sustainability of products/services. ISAD provides the computational thinking aspects, e.g. patterns (templates), decomposition, abstraction (different levels of details) algorithmic thinking (processes/data flow), prototyping and user testing.

4. SYSTEM DESIGN & DEVELOPMENT

Project Management considerations are first applied for systems analysis and design. This is followed by Waterfall/agile methodology for systems development.

4.1. Furnitize's design factors

For *Furnitize* (Chew, Chee, Wong, Hiew, 2017), the first Project Management consideration is *Project Integration Management.* Their design factors for *Furnitize* are extracted as follows:

a) User satisfaction, behaviors

Fayad and Paper's (2015) Technology Acceptance Model (TAM): perceived usefulness, perceived ease of use, and intentions; add four predictor variables to the original TAM: expectations, process satisfaction, outcome satisfaction, and e-commerce use – to extend TAM from measuring intentions to measuring actual behavior. Expectations (ease of use, usefulness), customer satisfaction (process and outcome satisfaction) and intention (e-commerce use) as design guidelines are thus utilized.

b) Cross-sell and Up-sell

Choosing which products to offer to which customers to maximize the marketing return on investment and to work around business constraints is complex but necessary to retain customers (Salazar, Harrison & Ansell, 2007).

i) Market segmentation analysis, purchase acquisition trees and survival analysis can be applied in many contexts;

ii) Lim and Lee's (2010) study on online analytics using classification and association rule mining.

c) Social Media and influencers

d) Gamification

Gamification, is transforming business models. It integrates game mechanics into non-game environments to motivate participation, engagement, and loyalty. Gamification works because it leverages on our motivations and desires for community, feedback, achievement, reward (Yang, Asaad & Dwivedi, 2017).

The derived system requirements are in Table 1.

Table 1. System requirements

Company strategies	**Company services**
Increase customers' satisfaction, confidence, and loyalty	• Customer relationship management system (live chat, social media, forum, subscribe, membership, news) • Customize: allow customers to have their own experiments with concepts (design their own floor plan, own decoration using templates, tutorials) • Google analytics, Gamification (future)
Increase variety of products, cost, time	• Joint venture with other companies • Delivery system (Supply Chain Delivery System) • Agent, to save cost and time • Installation and renovation services

The outcome of *Project Integration* is in Figure 2a.

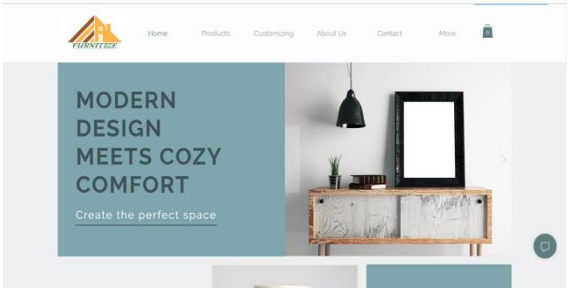


Figure 2a. Wing's 4 CT aspects: *Furnitize*
https://pailekchew963.wixsite.com/mysite

The second round of considerations are Project Scope Management, Project Time management, Project Cost Management and Project Quality Management. The outcome from this second round of considerations is illustrated in the choice of floor plans, and customization of interior design and furniture selection. Examples shown (Figures 2b, c, d) are customized screenshots, using the open source *RoomSketcher* software.

Figure 2b. Customizable System

Figure 2c. Interior Design and Furniture Selection

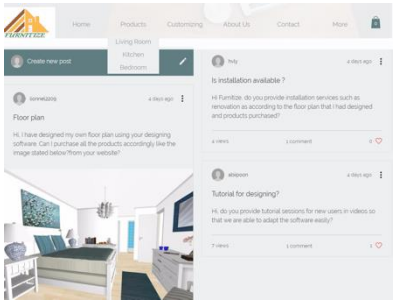


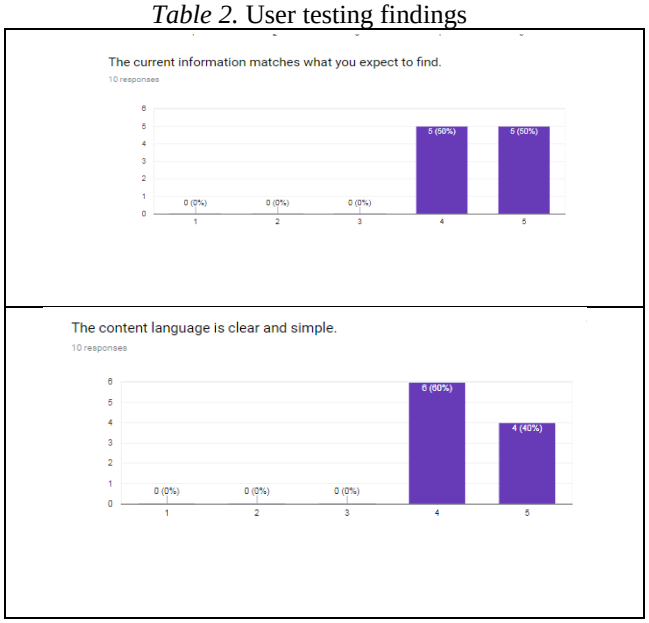
Figure2d. Forum

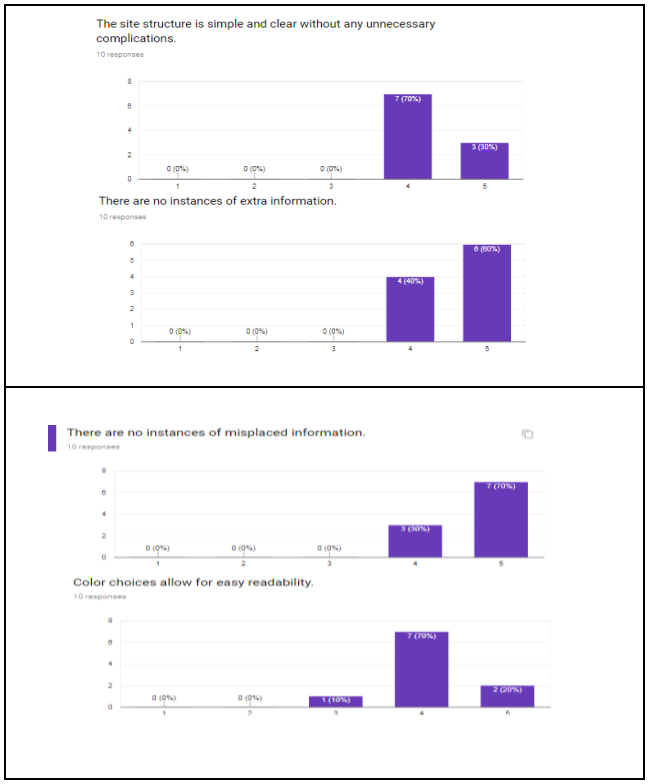
To confirm feasibility, a third round of considerations are factored in. These are Human Resource Management, Communication Management, Risk Management, Procurement Management and Stakeholder Management. This layered-iterative methodology reflects agile principles.

4.2 Evaluation (user perception)

The user testing questionnaire is designed based on generic human-computer interaction (HCI)/TAM principles. HCI/TAM principles, optimize search. Findings extracted from the report are presented in Table 2 below.

Table 2. User testing findings





4.3. The Enchantress' design factors

Inspired by Starbucks's gamified crowdsourcing ideation system, *The Enchantress* (Yew, Lim & Sugumar, 2017) is a crowdsourcing platform. Their proposition is to encourage the community to *develop a new habit* i.e., to recycle. To encourage and to sustain such new habits, would require not only time scheduling and task load considerations, but also, development of new perspectives through new value propositions. *The Enchantress* (Figure 3) piques imagination to the highest of what fashion is or can be. Hence, it's like a nested loop of perspectives.

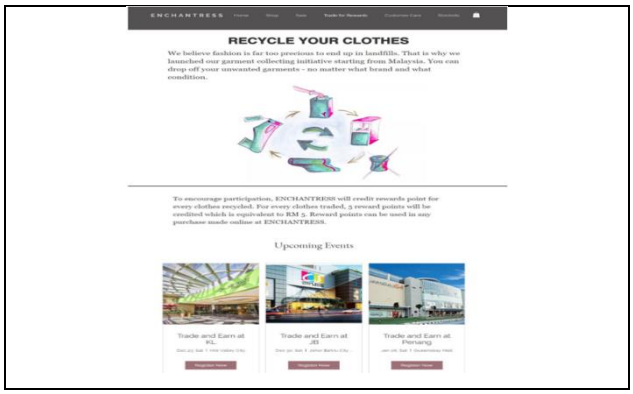


Figure 3. Brennan & Resnick's 3 CT aspects: *The Enchantress*

4.4. Evaluation (user perception)

Technology acceptance by users has also been promising though there are challenges as not everyone is interested in design. Nevertheless, due to its social innovation orientation of conserving the environment by encouraging product innovation and entrepreneurship, it is still worth a try. To sustain, a knowledge management framework has been

investigated (Yew & Lee, 2019). Findings are promising but indicates the need for smart partnerships.

5. IMPLICATIONS AND CONCLUSION

Prior research is aimed at investigating how we can scaffold generative/deep processing, i.e., how we can design deep reflective questions, which would contribute towards pattern recognition, theorizing, knowledge construction, and subsequently, creativity and transfer of learning along with the development of epistemic agency.

Both systems indicate that inter-disciplinarity in realistic ecosystems aimed at meeting real needs are the most effective motivators, confirming the efficacy of goal-based scenarios. Interestingly, design factors are similar and the two most important are the supply chain and cross-sell and up-sell; and the ultimate goal: sustainability.

These findings confirm success factors identified in Lee and Wong (2014; 2015; 2017; 2018):

a) design thinking (viability and sustainability of innovations) and computational thinking; [2015]
b) design as search/SEO/navigational structure (Interaction Design Institute); [2014]
c) Project Management (PM) grounded in Information Systems Analysis and Design and correspondingly, the Technology Acceptance Model (TAM), PMI; [2017]
d) marrying PM-TAM concepts with human-computer interaction metrics enhances design
e) marrying the above within a knowledge management framework ensures cycles of innovation. [2018]

The implications to teaching and learning are, first, the four key CT aspects are more oriented towards Computer Science projects in diverse contexts, with heavier research and Data Science underpinnings. Brennan and Resnick's (2012) 3 key CT aspects naturally have research and Data Science underpinnings, but are more easily understood and do-able for the masses, given Resnick's years of creativity research e.g. Scratch. Reducing entry level/cognitive access, fun, community engagement, overlay Computer Science/Data Science underpinnings. It is also easier for the masses to develop and transform value propositions.

Furthermore, adaptations are based on different centralities in design. Interestingly, *Furnitize* leans more towards structure, behavior, and function first whereas *The Enchantress* leans towards function, behavior, structure first. Hence, juxtaposing the 4 CT aspects against the 3 CT aspects highlight their complementarity based on goal-based scenarios, HCI and TAM principles to different contexts and spectrum of abilities in education.

6. ACKNOWLEDGEMENT

The two Project Management course assignments (August-December 2017) are by her Project Management students, Pai-Lek CHEW, Yew-Keong CHEE, Jing-Pynn WONG, Yit-Thang HIEW, Lee-Yin YEW, Kwan-Sheng LIM, Priscilla SUGUMAR. Much thanks for their inspiring work. Thanks also to Georgia Tech for introducing GBS during her Fulbright Fellowship in 2008/2009 while with Multimedia University, Dr. K. Daniel Wong for introducing design and computational thinking in the past. Thanks to CTE for propelling CT to greater heights.

7. REFERENCES

Arnold, John E. (1959). *Creative engineering*. Stanford.

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* AERA.

Dym, C.L., & Little, L. (2003). *Engineering Design: A Project-Based Introduction (2nd ed).* NY, John Wiley.

Fayad, R. & Paper, D. (2015). The Technology Acceptance Model E-Commerce Extension: A Conceptual Framework. *Procedia Economics and Finance, 26*, 1000 – 1006.

Lee, C. S. & Wong, K. D. (2014). Designing framing and reflective scaffolds to develop design thinking and transfer of learning: Theorizing for pre-school. *Proceedings of IEEE International Conference on Advanced Learning Technologies*. IEEE, 80-81.

Lee, C. S**.** & Wong, K. D. (2015). Developing a disposition for social innovations: an affective-socio-cognitive co-design model. *Proceedings of International Conference on Cognition and Exploratory Learning in Digital Age*, 180-186.

Lee, C. S. & Wong, K. D. (2017). An entrepreneurial narrative media-model framework for knowledge building and open co-design. *SAI Computing*, 1169 - 1175. IEEE.

Lee, C. S. & Wong, K. D. (2018). Design - computational thinking, transfer and flavors of reuse: Scaffolds to Information and Data Science for sustainable systems in Smart Cities. *Proceedings of IEEE International Conference on Information Reuse and Integration*, 225-228. IEEE.

Lim, A. H. L. & Lee, C. S. (2010). Processing online analytics with classification and association rule mining. *Knowledge-based Systems*, *23*(3), 248-255.

Project Management Institute (2017). *Pulse of the Profession.* Retrieved July 5, 2017, from http://www.pmi.org

Salazar, M., Harrison, T. and Ansell, J. (2007). An approach for the identification of cross-sell and up-sell opportunities using a financial services customer database. *Journal of Financial Services Marketing, 12*(2), 115-131.

Schank, R. C., Fano, A., Bell, B. & Jona, M. (1994). The Design of Goal-Based Scenarios. *Journal of the Learning Sciences, 3*(4), 305-345.

Yang, Y., Asaad, Y., & Dwivedi, Y. (2017). Examining the impact of gamification on intention of engagement and brand attitude in the marketing context. *Computers in Human Behavior*, *73*, 459-469.

Yew, L. Y. & Lee, C. S. (2019). Resource-knowledge-mixed Knowledge Management approaches to enhancing e-Commerce sustainability: A comparative case study. *Proceedings of International Conference on Engineering Technology.*

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Development of Programming Self-efficacy Scale for University Students in the Information Domain

Hsien-Sheng HSIAO[1], Jun-Wei LAI[2*], I-Ning WU[3], Chung-Pu CHANG[4]
[1,2, 3,4]Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taiwan
[1] Chinese Language and Technology Center, Institute for Research Excellence in Learning Sciences, National Taiwan Normal University, Taipei, Taiwan.
etlab.paper@gmail.com, polo24072407@gmail.com, nancy851224@gmail.com, enzoapu@gmail.com

ABSTRACT
The purpose of this study was to develop a self-efficacy scale for students in the field of information studentss about programming thinking procedures. The questionnaire consisted of 21 questions which divided into three dimensions such as "importance", "confidence" and "anxiety". The research object is the college students who have taken programming courses in Taiwan. There were 208 participants from northern, central, and southern Taiwan. The statistical methods used in this research include descriptive statistics, item analysis, exploratory factor analysis and Cronbach α internal consistency analysis. The internal consistency coefficient of the scale is between .950 and .957 and the validity of the construction is verified by factor analysis. On the whole, the scale has good reliability and validity.

KEYWORDS
information domain, programming, self-efficacy, programming self-efficacy scale

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

資訊領域大學學生程式設計思考程序自我效能量表發展之研究

蕭顯勝[1]，賴俊維[2*]，吳翊寧[3]，張仲樸[4]
[1,2,3,4]科技應用與人力資源發展學系，臺灣師範大學，台灣
[1]臺灣師範大學學習科學跨國頂尖研究中心，臺灣師範大學華語文與科技研究中心，台灣
etlab.paper@gmail.com，polo24072407@gmail.com，nancy851224@gmail.com，enzoapu@gmail.com

摘要
本研究旨在發展與編制資訊領域大學生對於程式設計思考程序自我效量表，量表包括「重要性」、「信心」、「焦慮」三個構念共21題，作答方式採用Likert十一點量尺。研究對象為臺灣修習過程式設計課程的大學生，以立意抽樣臺灣北、中、南部的208位大學生為預試樣本。本研究所使用的資料分析方法包括描述性統計、項目分析、探索性因素分析、Cronbach α內部一致性分析。量表的內部一致性係數介於.950至.957之間，並以因素分析來驗證建構效度，整體而言，本量表具有良好的信度與效度。

關鍵字
程式設計；資訊領域；程式設計；自我效能；程式設計自我效能量表

1. 前言
程式設計教育近年來在全球掀起一股浪潮，世界各國皆致力於推動程式設計教育，將程式設計納入課綱當中，這不僅是為了大量需求的科技人才，更是為了培養學生問題解決、創造性思考、勇於犯錯等能力，以及因應未來的數位生活（王令宜，2017）。美國前總統歐巴馬在 2016 年時提出「全民電腦科學教育」（Computer Science for All），讓全美的學生都能享有完整的電腦科學教育，具備基本的程式編寫能力，以因應新科技下急遽加速的未來，確保每一位學童都能夠站在公平競爭的起跑點。

程式設計教育可以讓學生根據程式語言的語法、語言結構與設計技巧來解決問題（Schollmeyer, 1996）。然而，學生在學習程式語言上時常會遭遇到許多困難，首先要將問題的描述轉換為邏輯，再由邏輯轉換為程式碼，這個過程對初學者來說是困難的（Sengupta, 2009；Saeli, Perrenet, & Jochems, 2011），此外，程式設計的初學者還需要記得許多抽象、不易理解的語法及命令，容易導致學習上的困難（Kelleher & Pausch, 2005；Lahtinen, Ala-Mutka, & Jarvinen, 2005）。學生的自我效能會影響其面對困難時的態度，相信自己能力的學生會勇於面對困難的任務，並將其視為需要解決的挑戰（Bandura, 1994）。自我效能是一種心理概念，它可以評估個人的心理狀態，自我效能會影響學生的活動選擇，包括學生將花費多少精力或時間來解決特定的任務和情況（Bandura, 1997）。Moos 與 Azevedo 也在 2009 年指出在程式設計課程中，學生的自我效能與學習表現有著密切關係（Tsai, Wang, & Hsu, 2019）。過往有相關研究曾開發出用於評估程式設計自我效能的量表。如 Ramalingam 與 Wiedenbeck（1998）發展由四個構面組成的調查問卷，來評量初新手學習 C++程式語言的自我效能；Askar 與 Davenport（2009）以及 Govender 與 Basak（2015）皆基於 Ramalingam 與 Wiedenbeck（1998）的量表來改編，以評量學生學習 Java 程式語言的自我效能。從上述研究可以發現，多數都是針對特定的程式語言的學習感受進行測量，鮮少針對程式設計的解決問題流程、運算思維概念的自我效能量表（Tsai, Wang, & Hsu, 2018）

因此在新科技不斷推陳出新，變化快速的當代，本研究將改善前述相關研究缺乏，發展一個不限特定程式語言環境的自我效能量表，並參考 Tritrakan、Kidrakarn 與 Asanok（2017）所提出程式設計的程序，編撰出學習程式設計的七個步驟，來測量學生在學習程式設計七個步驟的自我效能。本研究參考 Carberry、Lee 與 Ohland （2010）所開發的「工程設計自我效能問卷」進行改編，發展「程式設計思考程序自我效能量表」，未來希望透過此一量表來探究大學生在程式設計學習過程中態度之影響。

2. 研究方法
2.1. 研究工具
2.1.1. 預試量表之設計與架構
本研究蒐集學習程式設計時，學習態度、自我效能、態度傾向等相關文獻，根據文獻資料歸因出學習程式設計的影響自我效能的因素（Carberry, Lee, & Ohland, 2010），逐步形成完整之「程式設計學習態度」之核心構面，包括「重要性」、「信心」、「焦慮」三構念及 21 題項，並參考 Tritrakan、Kidrakarn 和 Asanok（2017）專家程式設計思考程序（Expert programming design thinking and procedure），發展出程式設計 7 步驟，包含（1）定義問題、（2）思考架構、（3）釐清資料、（4）設計演算法、（5）程式撰寫、（6）測試與除錯、（7）完成發表。經歷本研究團隊 3 次討論與字句修正，完成此量表，本量表採用李克特式十一點量表分別以「0」至「10」代表「低」至「高」，各構面之平均分數愈高代表學習者對此構面之態度愈高。

2.1.2. 預試量表編制
完成測驗題目初稿編制後，本研究邀請 3 位教授程式設計領域的專家學者對量表進行專家審查，審查與討論的重點為：（1）審查初始量表題項之歸類與重要性；（2）審查題項的題意與文字內容，提供題項之文字修改意見。

經專家審查完後，本研究將程式設計思考程序自我效能量表進行預試。在回收預試資料後，為了檢視檢驗測量題項是否適切，透過項目分析，以瞭解各題項之鑑別力與同質性，根據分析結果將不適合的題項刪除或修改；透過探索性因素分析，以瞭解各題項的聚斂情形。

預試結果經項目分析與探索式因素分析後，本研究正式問卷按照自我效能之三項核心構面與程式設計 7 步驟發展出 21 題項與個人基本資料 5 題（性別、年齡、學校、學系、年級），如表 1 所示。

表 1 設計思考程序自我效能量表因素名稱與題目分布

量表名稱	因素名稱	題號	題數
設計思考程序自我效能量表	重要性	1、4、7、10、13、16、19	7
	信心	2、5、8、11、14、17、20	7
	焦慮	3、6、9、12、15、18、21	7

2.2. 研究對象

本研究預適量表發放對象為資訊領域且在修習過程式設計課程的大學生。本研究以立意抽樣的方式進行，在臺灣的北、中、南十一所大學中進行預試施測。預試使用電子問卷發放，施測 208 份，回收 208 份，回收率為 100%。

2.3. 探索性因素分析

本研究經過專家審閱量表後進行預試數據收集後，項目分析結果，21 題項 CR 值均顯著水準（p <.001）；同質性檢定結果均為高度相關，故此 21 項均保留。進行因素分析前，為了瞭解取樣的適切性，首先使用 KMO 和 Bartlett's 球形檢定來判定是否做因素分析。結果 KMO 值=.889> .8，且 Bartlett's 球形檢定的 p 值達顯著（p=.000<.001），根據 Kaiser（1974）指出的判斷標準，KMO 值大於.70，且 Bartlett 球形檢定達顯著水準，適合作因素分析。採用正交轉軸之最大變異法萃取出三個構面。結果如表 3，每一題項的因素負荷量皆大於.5，無須刪題。各構面的特徵值皆大於 1，且總解釋變異量（%）達 79.24，顯示具足夠之效度。

表 2 設計思考程序自我效能量表項目分析表

題號	因素負荷量			共同性
	重要性	信心	焦慮	
IM1	.77			.65
IM2	.89			.84
IM3	.82			.73
IM4	.86			.80
IM5	.86			.79
IM6	.91			.84
IM7	.86			.77
CO1		.80		.76
CO2		.89		.81
CO3		.83		.77
CO4		.91		.85
CO5		.90		.83
CO6		.87		.83
CO7		.85		.75
AN1			.85	.74
AN2			.88	.80
AN3			.90	.80
AN4			.93	.87
AN5			.92	.86
AN6			.87	.77
AN7			.88	.78
特徵值(λ)	5.62	5.54	5.48	
解釋變異量(%)	26.78	26.38	26.08	
累積解釋變異量(%)	26.78	53..17	79.24	
各構面 Cronbach's α 值	.950	.955	.957	
總量表 Cronbach's α 值	.898			

3. 結論與建議

本研究建立一套「程式設計思考程序自我效能量表」，根據項目分析、探索性因素分析結果顯示，此一量表具有良好信效度。後續本研究將擴大規模發放正式問卷，期望本量表可以幫助大學資訊領域教師在程式設計課程規劃時，能夠了解學生在思考流程中是否遭遇困難、也能了解學生經由學習後對於程式設計思考的自我效能是否有所提升。

4. 參考文獻

Askar, P., & Davenport, D. (2009). An Investigation of Factors related to Self-efficacy for Java Programming among Engineering Students. *Online Submission, 8*(1).

Bandura, A. (1997). *Self-efficacy: The Exercise of Control.* New York: W.H. Freeman and Company.

Carberry, A. R., Lee, H. S., & Ohland, M. W. (2010). Measuring Engineering Design Self-efficacy. *Journal of Engineering Education, 99*(1), 71-79.

Govender, D. W., & Basak, S. K. (2015). An Investigation of Factors related to Self-efficacy for Java Programming among Computer Science Education Students. *Journal of Governance and Regulation, 4*(4), 612-619.

Kelleher, C., & Pausch, R. (2005). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys (CSUR), 37*(2), 83-137.

Lahtinen, E., Ala-Mutka, K., & Jarvinen, H.-M. (2005). A Study of the Difficulties of Novice Programmers. *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, 14-18.

Maltese, A. V., & Tai, R. H. (2011). Pipeline Persistence: Examining the Association of Educational Experiences with Earned Degrees in STEM among US Students. *Science education, 95*(5), 877-907.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-efficacy Scale and Group Analyses of Novice Programmer Self-efficacy. *Journal of Educational Computing Research, 19*(4), 367-381.

Saeli, M., Perrenet, J., Jochems, W.M.G., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education, 10*(1), 73-88.

Schollmeyer, M. (1996). Computer Programming in High School vs. College. *ACM SIGCSE Bulletin, 28*(1), 378-382.

Sengupta, A. (2009). CFC (Comment-First-Coding) – A Simple yet Effective Method for Teaching Programming to Information Systems Students. *Journal of Information Systems Education, 20*(4), 393-399.

Tritrakan, K., Kidrakarn, P., & Asanok, M. (2017). Development and Study the Usage of Blended Learning Environment Model Using Engineering Design Concept Learning Activities to Computer Programming Courses for Undergraduate Students of Rajabhat Universities. *Journal of Education, 11*(1), 46-59.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2019). Developing the Computer Programming Self-efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research, 56*(8), 1345-1360.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Evaluation

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Using Eye-Tracking to Evaluate Program Comprehension

Fabian DEITELHOFF[1*], Andreas HARRER[2*], Benedikt SCHRÖDER[3*], H. Ulrich HOPPE[4*], Andrea KIENLE[5*]
[1,2,3,5] University of Applied Sciences and Arts, Dortmund, Germany
[4] University of Duisburg-Essen, Germany
[1,2,5]firstname.lastname@fh-dortmund.de, benedikt.schroeder020@stud.fh-dortmund.de, hoppe@collide.info

ABSTRACT
Computational thinking has been identified as an essential problem-solving skill in the information age. Although more specialized, programming is an essential manifestation of computational thinking, and in turn, source code comprehension is a vital subskill of programming. The study reported here compares the effects of different source code examples on source code comprehension and different learning hints as a starting point for a dynamic learner support system. Our analysis relies heavily on using eye tracking data in combination with specific data models and visualizations. This form of behavioral analytics is complemented with answers to comprehension questions to assess the effects of these hints with different code examples. Our findings indicate that syntax highlighting is of limited benefit for better comprehension, and a dynamic highlighting of the scope of code blocks and variables is less used than expected.

KEYWORDS
eye tracking, program comprehension, computational thinking, learning analytics

1. INTRODUCTION
In a society permeated by digital representations and tools in professional and everyday life, the desirable general knowledge of science, technology, engineering, and mathematics (STEM) must be combined with more meta-level skills like critical thinking, adaptive problem solving, and creativity. As argued by Wing, "computational thinking" (CT) is an important ingredient in this context (Wing, 2006). Although CT cannot be reduced to programming, programming is an activity that both builds on CT and can support the development of CT. Accordingly, it has been argued that there is an overall value in learning basic concepts and skills of programming. However, programming is a complex cognitive activity (Pea & Kurland, 1984). When learning to program, comprehending source code is the priority.

Eye tracking is more and more integrated into the process of analyzing learners and creating better support systems (Njeru & Paracha, 2017). Additionally, it is a powerful ingredient in the context of learning analytics (Greller & Hoppe, 2017).

In this paper, we describe the analysis of comprehension problems participants encounter while reading source code and answering comprehension questions. Especially, the detection of common reading patterns may reveal differences in computational thinking and understanding among participants.

2. ANALYSIS APPROACH
The basis of every eye tracking analysis is fixation hits on specific regions. In the source code examples, AOIs are placed around every code line (*line model*) or every important workspace area (*workspace model*). (Deitelhoff, Harrer & Kienle, 2019b) These are marked with letters, in the line model from top to bottom, with additional AOIs for the question and answer areas, and in the workspace area model with *A = answer*, *C = code*, and *Q = question*. Additionally, we label non-hits with "_", to identify gaps in, e.g., transitions.

We used the recorded eye tracking data to calculate fixations based on the raw data. The fixation calculation is done with an I-VT filter (*Velocity-Threshold Identification*) with a maximum radius of *60 pixels*, a minimum fixation duration of *60 ms*, and a maximum of *55* missing gaze samples to count as a fixation.

For analyzing the reading behavior of participants, we are using a *top-down* approach with predefined patterns. Two global patterns are, e.g., the *Linear Scan* and *Jump Control*, also known as *Story Order Reading (SOR)* and Execution *Order Reading (EOR)* (Busjahn et al., 2015). SOR is a reading pattern from top to bottom, like a story in a normal text, while EOR follows the program execution. Besides, we use one visualization to show the fixation order of AOIs (Deitelhoff, Harrer & Kienle, 2019a). Furthermore, we use our analysis tool CodeSight, which provides the feature to search for eye movement patterns (fuzzy search).

3. RESEARCH QUESTION
Previous research has shown the effects of different source code examples on reading patterns, and of syntax highlighting as a form of learning hint. Some studies found effects for novices or in general (Asenov, Hilliges & Müller, 2016); some do not (Hannebauer, Hesenius & Gruhn, 2018). The highlighting is used as a visual cue for programmers to decrease the time required for mental execution. Novices tend not to use/ignore the highlighting or misinterpret the meaning completely. The objective of our study was to investigate the effect of learning hints on the outcome of source code comprehension processes. Additionally, we analyzed how learners use and perceive the source code examples. In summary, we tested the following research questions and hypotheses.

$H_{\text{Answer-Quality}}$ We examined how the answer quality differs between the various source code examples and learning hints. We assume that more complex code examples have less correct answers overall and that learning hints influence the answer quality.

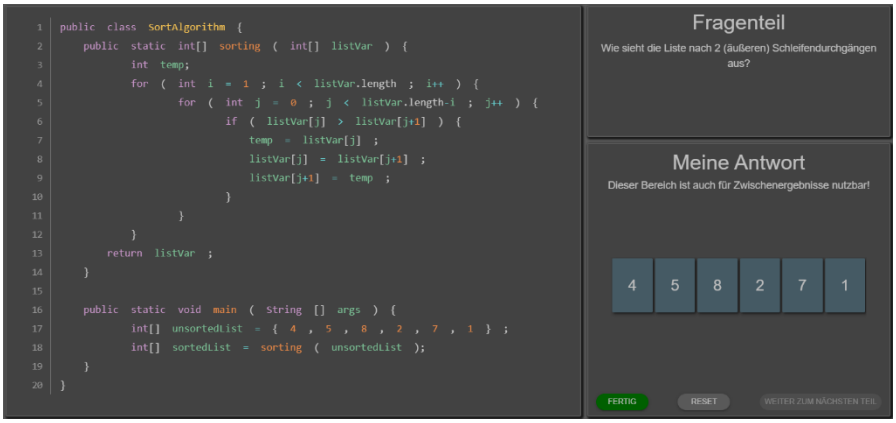


Figure 5. The study prototype with the code example "Bubble" and with syntax highlighting.

$H_{Answer\text{-}Quality}$ We examined how the answer quality differs between the various source code examples and learning hints. We assume that more complex code examples have less correct answers overall and that learning hints influence the answer quality.

$H_{Patterns\text{-}Answer\text{-}Quality}$ We found the patterns *Story Order Reading (SOR)*, *Execution Order Reading (EOR)*, and *Flicking* in the visualized AOI-DNAs. We assume that the presence of these patterns is correlated to more correct answers.

$H_{Workspace\text{-}Area\text{-}Switches}$ We analyzed, which visual context switches between important workspace areas of the study prototype are common between learners. We propose that different context switches, and therefore comprehension strategies, are visible. A different perception of the workspace can lead to different approaches in solving the comprehension questions, which may affect CT strategies.

4. STUDY PROTOTYPE

We used three code examples *Bubble*, *GCD*, and *Vehicle* as stimuli. They correspond to the algorithms *Bubble Sort*, *Greatest Common Divisor*, and a class that represents a *Vehicle* with methods like accelerating and decelerating. The complexity of these code examples varies between *complex* (Bubble), *medium* (GCD), and *easy* (Vehicle), assessed with the help of researchers involved in education, learning analytics, and teaching. **Error! Reference source not found.** shows an example screenshot for the Bubble source code. To measure how successful participants comprehend the source code, we asked the following comprehension questions:

Bubble "What does the list look like after two runs of the outer loop?"

GCD "To which values are the variables 'number1' and 'number2 set after three runs of the loop?"

Vehicle "To which values are the objects 'vOne' and 'vTwo' set at the end of the program?"

The code examples are fixed in their order (Bubble → GCD → Vehicle), but with varying hints. We distinguish between passive and active learning hints. The first is always available, and the latter needs to be used actively by the participant. The hint *Syntax Highlighting* highlights the Java code is passive and helps to navigate the code and focusing on parts like variable assignment and logic (see figure 1). The second hint, called *Dynamic*, allows learners to focus a variable or curly bracket with the mouse to highlight the scope of either the usage of the variable or the, of a source code block. Therefore, this hint is active. The third hint, called *Plain*, is our control group without any hints.

5. DATA BASIS & ANALYSIS RESULTS

In this section, we report the results of our quantitative and qualitative analysis of every hypothesis. As the data basis, we recorded n = 24 participants from the nearby University campus, out of which seven were females and 17 males, with a mean age of 26.29 (SD = 4.28). The participants were all Computer Science students (semesters 1-10).

5.1. Answer Quality

The overall correct answers for the Bubble source code are 12, for the GCD again 12, and for the Vehicle 5. Therefore, the Vehicle code example seems to be more complex. This result is contrary to our assumption, from an algorithmic perspective, that the GCD is the most complex code example. It seems that many participants had problems with the object-oriented task. If we additionally consider the time limit of every code example, we can confirm our impression that many participants had problems with the Vehicle task. Four participants exceeded the time limit for the Bubble source code, 2 for GCD, and 9 for the Vehicle.

The data also shows that the syntax highlighting learning hint is balanced for the correct/incorrect answers. Syntax highlighting seems not to be an essential factor related to answering a comprehension question. Complex code is still complicated. However, the difference between the dynamic learning hint and the plain code examples are indecisive for the Bubble and GCD code examples. For the first code, plain has a more significant effect on correct answers than the dynamic learning hint (5 to 3 participants). This result is reversed for the GCD code examples, with a more substantial effect for the dynamic help on correct answers (2 to 6 participants). This is positive for the dynamic learning hint of the GCD code example and interesting for the Bubble code example. Overall, this needs a more in-depth analysis, how often the dynamic learning hint was used across code examples. For the Vehicle code, both conditions with the dynamic help and the plain text seems to have no positive effect on the comprehension result. Again, this needs to be analyzed further on how often the dynamic learning hint was used.

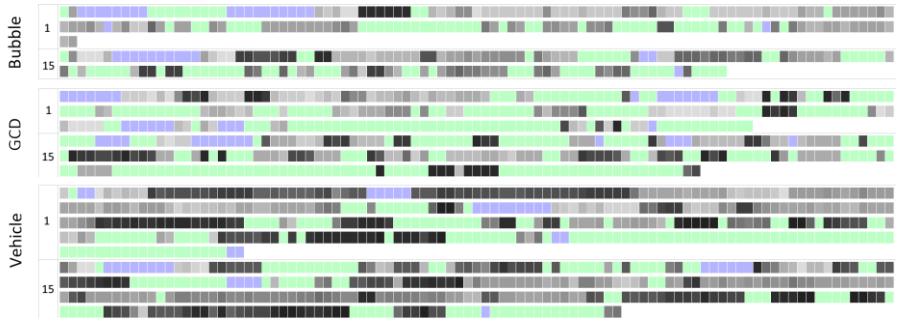


Figure 6. AOI-DNAs for the participants 1 and 15 for the code examples Bubble, GCD, and Vehicle.

5.2. Reading Patterns to Answer Quality

As a first step, we are visualizing the gaze patterns of participants in our analyzing platform *CodeSight* to reveal the reading behavior. The visualization, in the form of our AOI-DNAs, shows some similarities between the participants. We describe our findings for participants 1 and 15 as examples. **Error! Reference source not found.** shows the AOI-DNAs for both participants for all three code examples. The visualization uses a grayscale color coding for visualization of the source code lines from 1-n for the line AOI model. The parts with, e.g., loops and methods, are brighter, and the main method is darker. The question area is light purple, whereas the answer area is light green.

For the Bubble code example, both participants almost immediately start with reading the question, followed by reading the main method and a SOR phase subsequently. Afterward, the answer and code fragments are read alternately. For the GCD code example, participants again start with the question, followed by reading some parts of the code. Afterward, participants, like 1 and 15, are alternately reading the answer and sections of the code, with participant 15 reading the main method more often. For the Vehicle code example, participant 15 focuses more on the question in the beginning, while participant 1 reads the code, mostly the main method, first. Afterward, both participants read parts of the code and the answer area.

Our analysis platform *CodeSight* supports searching for eye movement patterns based on regular expressions, to find common patterns linked to the assumption that they have advantages for comprehending source code. For our analysis, we searched for the Execution Order Reading (EOR) and Flicking patterns. The SOR pattern is already visible with the grayscale visualization and, because of the length and diversity, hard to search for directly. The patterns EOR and Flicking should show advantages for the participant for answering the comprehension question correctly. Searching for patterns is dependent on an appropriate regular expression. The expressions are based on the character labels for the AOIs. Therefore, we are describing the transitions we found with these labels. For the EOR, we are searching for AOI transitions like F|G|H|I → E|D (Bubble), E|H|D → C (GCD), and T|U → C|D|E or Y|Z → M|N|O (Vehicle) whereas the vertical separators are used only to indicate the different AOIs within the patterns visually. For the Bubble code example, the pattern search revealed, that the patterns F|G|H|I → E are often present in the AOI-DNAs. These eye movement patterns are essential because they encode reading and comprehending the loop structures. The fuzzy pattern search also shows that most of these patterns, especially F → E, and G → E, are visible for participant 1 with a far better comprehension result compared to participant 15. Reading and tracking the loop structures is vital for comprehending the Bubble example. For GCD, the second code example patterns like E|H|D → C|D are important. We found multiple hits for E|D → C and H → C|D. These are important patterns for the while loop and jumps from the two branches within the if statement to the while loop. Both necessary for comprehending the structure and behavior of the GCD algorithm. As for the Bubble code, we found differences between the two participants. Reading and tracking the loop and the if structures are vital for comprehending the GCD example.

These results are also true for the Vehicle code example. We identified important patterns like T|U → C|D|E, X → G|H|I, or Y|Z → M|N|O. These are encoding (a) jumps from the main method, were, among other things, constructors are called, to the constructor definitions, and (b) jumps from the for loop in the main method with method calls to the corresponding method definitions. In contrast to the other two code examples, we found only a small amount of pattern matches for both participants.

5.3. Workspace Switches

First, we analyzed the overall fixation time for the source code examples and the distribution of these durations on the three AOIs in the workspace AOI model. The overall fixation time for the question area is the highest for the Bubble source code. This result is not a surprise, because sorting an array takes many fixations and thereby time to complete. The highest fixation time for the Vehicle source code without any help is too as expected and in line with our other analysis. The Vehicle example is the most difficult one according to the participants, and the plain condition without any additional help amplified this difficulty level. For the Bubble source code example, the most fixation time spent on the code AOI with the dynamic learning hint. For the GCD code example, the most fixation time spent again on the code AOI, but this time with the syntax learning hint available. The fixation time results for the Bubble and GCD code examples are a bit surprising. We assumed that for the syntax and dynamic learning hints, these values should drop. One reason could be the lower usage rates we see in the data. Another possible reason is, that for the syntax learning hint transitions and therefore fixations are getting higher, because participants can read the source code much better, and for the dynamic learning hint transitions and fixations between the source code and answer area are higher because participants getting help from emphasized variables which helps them answer the question.

5.4. Overall Comprehension Problems

Overall, we find more comprehension problems in the Vehicle code example than in the other two code examples. We visually analyze the fixation distributions on the stimuli. For that reason, we superimpose the fixations on top of the source code examples. In three cases, it is noticeable that these participants do not have enough fixations on the for loop in the main method. Therefore, it is explainable why these participants failed the source code comprehension questions. But overall, the fixation distribution is equally good or bad compared to the other two code examples. We assume that this difference has to do with object-oriented programming because the Vehicle example uses a class with methods, which are called in the loop within the main method.

After the participants did the comprehension tasks, we asked them in the conclusive interview the question, among others, if they can identify the source code examples. Not on a specific algorithmic level, but in a more meta-level way. The specific question in the interview was: "What was the aim of the individual program codes?". For the code examples Bubble and GCD, the participants could answer this question very specifically most of the time (70%). Whereas, the answers for the Vehicle code example were much less precise. In most of the cases (> 80%), the participants could only tell that it has something to do with "a vehicle, which can be controlled".

Besides, we analyze the duration time and fixation count of every participant on the AOIs. Therefore, we can count the overall durations and fixation counts per source code line. The results show that the duration of important areas of the source code examples is no decisive factor for a correct comprehension question. For the Bubble source code, important areas are D and E for the loop, F for the if statement, and G, H, and I for swapping the values of two array elements. Participants with a (very) high fixation duration, and these AOIs are not answering the comprehension question more correctly overall. The inner loop (AOI-E) of the bubble sort seems to be the most important one regarding the answers, but for the other AOIs, the results are inconclusive, which is a bit different for the GCD and Vehicle code examples, whereas the fixation duration on important AOIs seems to have an impact on the answer quality of the source code comprehension. In contrast to our expectation, we found that the fixation duration is not a good predictor of comprehension success. This result is quite different for the fixation counts. We can summarize that a participant, who fixates an important AOI more often, gives overall more correct answers for the comprehension questions, which is especially true for the GCD and Vehicle code examples and source code elements like loops and if statements.

6. SUMMARY & DISCUSSION

To our surprise, the Vehicle code example was the most difficult one, regarding the answers of participants. We initially assumed that the Bubble code is the most complex one, regarding the complexity of the program structure (nested loops). However, the study showed that many participants have problems with the object-oriented code, no matter which learning hint was available.

The dynamic learning hint was less used than expected. We thought that our target group, with knowledge in programming and therefore, development environments would use this hint more frequently. Overall, the hint may be useful for the GCD example but ambiguous for the other two code examples. This finding needs more in-depth analysis and a specific study if the dynamic learning hint is a candidate for the dynamic learner support system.

Regarding the reading patterns on both used AOI models, we found common patterns across all participants, code examples, and learning hints. The analysis showed that these patterns form groups. Furthermore, a first analysis showed that these patterns are not the distinguishing factor for the answer quality of participants.

7. REFERENCES

Asenov, D., Hilliges, O., & Müller, P. (2016). The Effect of Richer Visualizations on Code Comprehension. *Proceedings of the Conference on Human Factors in Computing Systems*. ACM, 5040–5045.

Busjahn, T. Bednarik, R., Begel, A., & et al. (2015). Eye Movements in Code Reading: Relaxing the Linear Order. *Proceedings of the IEEE International Conference on Program Comprehension*. IEEE, 255–265.

Deitelhoff, F., Harrer, A., & Kienle A. (2019a). An Intuitive Visualization for Rapid Data Analysis: Using the DNA Metaphor for Eye Movement Patterns. *Eye Tracking Research and Applications Symposium,* 78-83.

Deitelhoff, F., Harrer, A., & Kienle, A. (2019b). The Influence of Different AOI Models in Source Code Comprehension Analysis. *Proceedings of the 6th International Workshop on Eye Movements in Programming*. IEEE Press, 10–17.

Greller, W., & Hoppe, U. (2017). *Learning Analytics: Implications for Higher Education*. Books on Demand.

Hannebauer, C., Hesenius, M., & Gruhn, V. (2018). Does Syntax Highlighting Help Programming Novices? *Proceedings of the International Conference on Software Engineering 23*(5), 2795–2828.

Njeru, A. M., & Paracha, S. (2017). Learning Analytics: Supporting At-risk Student through Eye-Tracking and a Robust Intelligent Tutoring System. *Proceedings of the 2017 IEEE International Conference on Applied System Innovation: Applied System Innovation for Modern Technology*. IEEE, 1002–1005.

Pea, R. D., & Kurland, D. M. (1984). On the Cognitive Effects of Learning Computer Programming. *New Ideas in Psychology 2*(2), 137–168.

Wing, J. M. (2006). Viewpoint: Computational Thinking. *Communications of the ACM 49*(3), 33–35.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Learning Behaviors Analysis of the Six Grader Students Integrating Educational Robots with the Computational Thinking Board Game

Tzu-Chin ZHOU[1], Ting-Chia HSU[2*]
[1,2]National Taiwan Normal University, Department of Technology Application and Human Resource Development, Taiwan
K8k3k13@gmail.com, ckhsu@ntnu.edu.tw

ABSTRACT
This study aimed at integrating computational thinking board game with robots, so that learners put computational thinking process into practice when they completed the tasks on the board game by controlling the action of the robots. The participants were the sixth-grade students in Singapore. Two students divided into a team collaborated with each other and competed with the other team composed of two students. This study developed a table of the behavioral coding schema according to the observations of the students' behaviors. From analyzing the overall learning behaviors of the students, this study evaluated and found the learning behavioral patterns of the students in the learning circumstances.

KEYWORDS
computational thinking, board game, collaborative learning, behavior analysis

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

小學六年級學生使用教育機器人結合運算思維桌上遊戲之學習行為分析

周子敬[1]，許庭嘉[2*]
[1,2]國立臺灣師範大學，科技應用與人力資源發展學系，臺灣
K8k3k13@gmail.com, ckhsu@ntnu.edu.tw

摘要
本研究旨透過運算思維桌上遊戲與機器人，讓學習者透過控制機器人在桌遊上面完成任務，來實際練習運算思維的歷程。本研究使用手機積木程式來控制桌遊上面機器人的動作，以新加坡小學六年級學生為受測對象，並以二人為一組的合作學習模式和另外二個人的小組進行對戰。本研究根據學生的行為內容進行行為編碼表，並且觀察學習者的完整學習過程，藉此來評估即發現學習者在此學習情境下的學習行為模式。

關鍵字
運算思維；桌上遊戲；機器人；合作學習；行為分析

1. 前言
近年來，學習運算思維的概念變得非常重要，而且被認為是在數位時代不可或缺的一種技能 Kalelioglu（2016），甚至許多國家已經把運算思維的概念引入到 K-12 的課程當中（Grover & Pea, 2013），不論是在資訊科技、數學、社會研究和程式設計，都有涉及到運算思維的概念（Barr & Stephenson, 2011），運算思維在當今社會的重要性，不言而喻。

許多研究也保持著讓每個人都要學習運算思維的想法，但大多數都是使用機上程式設計來學習運算思維的邏輯，因為透過程式設計，能夠實作運算思維中的結構化、抽象化、問題拆解等能力，但這種抽象的程式設計邏輯，並不適用在每個學習者身上，如果無法引起學習者的動機，只會降低學習者的學習意願。

所以，本研究使用運算思維桌上遊戲，以遊戲式學習的教學策略，藉以提高學習者的學習態度與動機，此外，本研究還觀察學習者在學習過程中的所有行為，並且開發行為編碼表，以序列行為分析的方式，觀察學習者認知或行為的變化，希望找出學習者行為之間的關聯性，以便找到更好的教學方法或策略，提升學習者的學習成效和表現。

所以綜上所述，學習運算思維已經是全球趨勢，身處在一個資訊爆炸的時代，幾乎人人都需要一點運算思維的概念，本研究希望透過桌上遊戲式學習結合安譜機器人，搭配手機應用程式控制的教學策略，並從中把運算思維的抽象概念與程式設計的邏輯結合，藉以吸引學習者的注意力，激發學習者的動力及效率，進而讓學習者獲得更好的學習成效。

2. 文獻探討
2.1. 運算思維
運算思維，就是一種用電腦的邏輯來解決問題的思維，雖然至今對於運算思維的定義還沒有一個統一的答案（Zhao & Shute, 2019），最早的定義由學者 Jeannette Wing 在 2006 年時提出，Wing（2006）認為運算思維是利用電腦科學的基本概念進行問題解決、系統設計與人類行為理解的思維模式，而在 2010 年時 Wing 更進一步提出運算思維是提出問題及其解決方案所涉及的思維過程（Wing, 2010）。

Wing（2006）認為電腦運算思考的技巧，並不只是電腦科學家或是一些相關人員的專利，而是每一個人都應該要具備的素養和能力，因為運算思維和我們日常生活的關係越來越密切，舉凡醫療、購物、交通、社交網路等等，都包含在其中，所以具有運算思維的能力更能夠有效的解決日常生活中遇到的問題。。現在最被大家所接受的是 Google 在自己的教育網站中所提出的（Google, 2016），把運算思維分成心理的思考過程，共有 11 項定義，在本研究有涵蓋到的部分如：

抽象化：為定義主要概念去識別並萃取相關資訊，演算法設計：設計出有順序的指令以解決問題或完成任務，自動化：利用電腦或機器執行重覆性的任務，解析：將資料、過程、問題拆解成較小、較容易處理的部分，平行化：同時處理大任務時，也同時處理較小的任務以有效達到解決問題目的，

2.2. 遊戲式學習
遊戲，既可以指人的一種娛樂活動，也可以指這種活動過程。一般是以娛樂為目的，有時也有教育目的。法國社會學家 Caillois（1957）定義了遊戲是有以下特性的活動：有趣、獨立性、不確定性、虛構、無生產性和受規則的約束，遊戲的分類有很多項，例如：數位遊戲、桌上遊戲，本研究是使用桌上遊戲當作教具進行教學活動。

學者 Hogle（1996）提出遊戲對於學習有下列優點：可引發內在動機並提高興趣，遊戲中好奇與期望、控制與互動性以及故事情節的幻想性等特性，都可提高學習者的學習興趣和內在動機（Dichev & Dicheva, 2017）。在保留記憶方面，相較於傳統的課程，模擬遊戲在記憶保留方面有較好的效果（Acquah & Katz, 2020）。並且能提供練習及回饋，許多遊戲學習軟體提供練習的機會，讓學習者可以反覆的操作，並獲得即時的回饋，讓學習者可以自我評估學習成效，促進學習目標的達成。提供學習者高層次的思考，將教學內容融入遊戲當中，讓學習者不斷的在遊戲中解決問題、做決定，學習者要能夠整合自己所學，以找到解決方式。教學內容將不斷的重複進入學習者記憶中，是最好的學習形式。

遊戲式學習可以積極激發學生在課堂上的行為（Simões, Redondo, & Vilas, 2013）中，遊戲式學習將激勵他們付出更多的努力在這門科目上。遊戲式學習也可以顯著增強學生在合作學習中的參與度（Hew, Huang, Chu, & Chiu, 2016）。Lin and Davidson-Shivers

（1996）的研究中發現採用遊戲式學習後，學生的最終成績得到了顯著提高。Ebner and Holzinger（2007）的研究中表示，大多數學生，他們參與過遊戲式學習之後會去主動進行遊戲式學習。所以根據上述研究，可以通過遊戲式的學習環境來培養學生的學習成就以及課堂參與度。

2.3. 序列行為分析

序列分析，或者稱行為序列分析是將研究對象的行為資料進行編碼，依照行為出現的先後順序，找出一個行為接著另外一個行為出現的頻率，並以二項式檢定計算編碼與編碼之間的轉換是否有達到顯著性的一種方法（Sackett, 1980）。

序列行為分析的目的是為了瞭解學習者在學習過程中，依照研究者實施不同的教學策略，觀察學習者認知和行為改變當中的交互作用，並且了解哪些行為對於學習者來說是有意義的學習，以及學習者行為反應之間的關聯性。

行為序列分析也在不同的領域中使用（Bakeman, 1997），過去也早有過將教育遊戲結合行為分析的研究（Barab et al., 2009），而本研究也是探討以遊戲式學習的教學方法觀察學習者的行為過程，就從這邊可以將學習者行為轉換，分析究竟哪些是屬於比較明顯的議題，藉此了解何種學習行為促使學習動機提升，了解學習者學習的認知成果，並藉此找到更好的教學策略與方法（Chang et al., 2014）。

2.4. 合作學習

合作學習，是一種有系統性的教學策略（Slavin, 1985），讓學習者在小組與同儕之間相互學習，分享大家的觀點並共享成果（Parker, 1985），這種以學習者為主角的教學過程裡，除了個人的努力之外，每位成員也要有所貢獻，以達到共同設定的目標，而老師在其中只扮演引導者和協助者的角色（Joe Cuseo, 1992）。

合作學習大都包含下列五項要素（Johnson, Johnson, & Holubec, 1994），積極互相依賴是指學習者能知覺到自己與小組是榮辱與共的，因此組內的每一個成員都要一起努力，以完成任務（Nattiv, 1994）。面對面的助長式互動是組內學習者可以相互助長彼此學習的成效，例如鼓勵組內的其他同儕、努力完成任務、達成共同目標。個人責任是指如果個人的表現不好，小組的表現也不會好。因此，合作學習除了注重小組的整體表現外，更重要的是個人的表現。在合作學習下，學生就會察覺到個人的努力與小組相關，所以反而會更督促自己。人際與小組技能是學習者之間如果有好的協同合作，將會有高品質、高效率的學習成效。在合作學習的情境下，彼此之間互動磨合，發生衝突無法避免，所以要教導學習者：相互信任、良好溝通、相互接納、化解衝突。團體歷程是指給予學生適當的時間及去檢討小組的運作狀況，強調自我檢視的重要性，並檢視組員在過程中哪裡需要改進的一種反思過程。

所以綜合合作學習可以帶來的好處。比如：可以幫助學生提升深度學習和批判性思考的能力（Munir, Baroutian, Young, & Carter, 2018），促進學生擁有更高的學習成效、提高學生的社會競爭力，讓學生獲得更好的學程成就等等。

3. 研究方法

3.1. 實驗對象

本次實驗由 25 位來自新加坡中的一所小學六年級學生，13 名男生 12 名女生，這些學習者的平均年齡為 11-12 歲，教學課程主要是以遊戲式學習配合機器人，以手機應用程式控制機器人的行為，讓學習者學習運算思維的概念，釐清運算思維的抽象意義，進而提高學習者的學習成效。

3.2. 行為編碼表

本實驗透過遊戲式學習的教學策略，記錄學習者在實驗過程中的所有行為，並將資料進行編碼，行為編碼表如表 1 所示，分析學習者在實驗過程的所有行為，並探討行為之間的關聯性。

表 1 行為編碼表

類別	代碼	意義	範例
運算思維編碼	PP(People&People)	組內對談	同組的兩個人在對談
	PC(People Commnication)	組外對談	與別組在對談
	PR(People & Robot)	使用機器人	掃描卡牌使機器人移動
	ID(Individual Decision)	個人使用任務卡	使用石頭、砂
	CD(Cooperation Decision)	共同使用任務卡	土…等，放置任務卡上
	AT(Algorithm)	使用卡牌	排除卡牌(前進、左轉…等)
	PM(Physical Message)	姿體表達	行為左右轉、手勢左右轉…等
	AG(Abstraction General)	資料簡化或用其他方式表達	單程式方法便迴圈方式表達
	LI(Learning Interaction)	被觀察者正在練習口語互動	自己口語互動
其他	IM(Irrelevant Message)	無關課程	發呆、離開座位…等
	SP(Separate)	組內做不同的事	各做各的事

3.3. 評估工具

本次實驗使用 Robots city 桌上遊戲配合安譜機器人進行，加上手機應用程式，並以程式設計的邏輯控制機器人的行為，在本次實驗透過觀察學習者的行為並開發編碼表，研究學習者的行為資料進行編碼，並利用序列分析之殘插表的 Z 分數檢定來解釋編碼與編碼之間的轉換是否有達到顯著性的關聯。

學習者自行搭建完所需之場景後，接著教師便會指派 APP 的關卡任務，讓學習者自行設計機器人的動作如圖 1，進行通關任務。

圖 1 學習者使用手機 APP 自行設計安譜機器人的行為

4. 實驗設計與結果

本次實驗的樣本為新加坡小學六年級的學生，旨在以遊戲式學習的教學策略下配合安譜機器人，並使用手機應用程式，以程式設計的邏輯控制機器人的行為，藉以讓學習者了解運算思維的抽象概念，並觀察學習者在實驗過程中的所有行為，透過序列行為分析，了解學習者在學習過程中發生的行為順序之間的關聯性，從而瞭解學習者學習的認知成果。

實驗流程如圖 2 所示，一開始時在教師進行簡單的講解之後，學習者便可以自行搭建 Robots city 所需之場景，接著教師便會指派不同的 APP 任務讓學習者依照指示通關，並將學習者以 3~4 人一組，進行分組競賽，藉此來評估學習者在遊戲式學習的教學策略下中是否了解運算思維的概念。

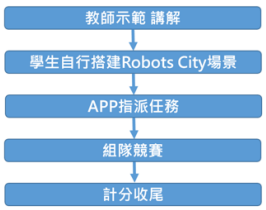


圖 2 實驗流程

透過 GSEQ 分析後的 Z 分數的結果為表 2 所示，其中若 Z 分數大於 1.96 則表示行為間有顯著關係。

表 2 Z 分數分配表

Given:	PP	PC	PR	AT	ID	CD	LT	IM	SP
PP	11.7	-0.07	-1.7	-3.8	-2.56	-0.8	-3.07	-0.06	-0.66
PC	3.01	6.73	-2.35	-2.81	-2.16	-0.68	-0.79	0.1	-0.91
PR	-4.06	0.02	6.51	5.89	-3.48	-1.33	-4.71	-3.41	-0.8
AT	-2.71	-1.94	-3.51	6.01	11.9	0.7	-3.62	-1.95	-0.58
ID	-2.54	-1	5.47	-1.49	1.07	-0.53	-2.19	-0.95	-0.37
CD	-0.69	-0.58	-1.17	-0.68	-0.45	21.05	-0.73	-0.45	-0.34
LT	-2.77	-2.28	-3.49	-3.34	-1.76	-0.85	15.32	-1.71	-0.81
IM	-1.07	-1	-1.76	-2.48	-0.98	-0.53	-0.31	12.21	-1.25
SP	-1.27	-0.14	-0.41	-1.86	-1.25	-0.39	-0.78	0.56	9.4

使用遊戲式學習搭配安譜機器人，並以手機應用程式導入程式設計的邏輯，用來控制機器人的行為，對學生的行為進行分析，圖 3 為 Z 分數比較表的視覺化呈現，在實驗過程中的行為，會發現同一組學習者在與另一組交換意見過後，會與自己同組的成員進行討論 PC→PP。

而在操作手機應用程式控制機器人時，學習者一開始在還沒有任何問題的拆解時便會先操控手機 APP，進行一些簡單的演算法步驟，爾後開始遇到問題時才會進行拆解，接著才會再次操作手機 APP，讓機器人進行正確的動作進而完成這次的指派任務 PR→AT→ID→PR。

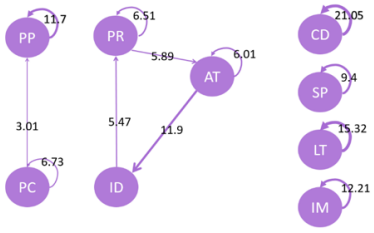


圖 3 Z 分數比較表

5. 結論與未來展望

在本研究主要探討桌上遊戲式學習結合安譜機器人，並從中導入運算思維的概念，還進一步結合了手機應用程式的教學策略，讓學習者透過本研究之系統了解運算思維的抽象概念。除此之外，本研究還開發行為編碼表，紀錄學習者在研究過程中的所有行為，分析學習者在實驗過程中的哪些行為有顯著差異，研究結果顯示，同一組學習者在與另一組交換意見過後，會與自己同組的成員進行討論 PC→PP。研究推論，因為在各組別之間進行討論時，也會想要了解其他組別討論出來的意見，所以會想要跟其他組別進行交流，彼此交換完意見後，自己組員分享其他組的觀點，讓討論的過程更加完善。

另一方面，而在操作手機應用程式進行程式設計操控機器人的行為時，學習者一開始在還沒有任何問題的拆解時便會先操控手機 APP，進行一些簡單的演算法步驟，爾後開始遇到問題時才會進行拆解，接著才會再次操作手機 APP，讓機器人進行正確的動作進而完成這次的指派任務 PR→AT→ID→PR。研究推論，因為學習者可能第一次接觸到手機、平板、機器人等裝置，所以學習者一拿到裝置時，出於好奇，會先自己使用看看，讓機器人進行一些簡單的演算法步驟，而後才會開始進行問題拆解，想好如何設計機器人的行為，在遊戲化的過程中完成這一次的指派任務，從這裡也可以發現，使用本研究的學習者在遊戲過程中確實累積了一些運算思維中的演算法設計、自動化、問題解析等概念。然而，在本研究中仍存在部分限制，本研究中的研究對象為新加坡國小六年級的學生，沒有包含所有區域之學生，未來可以將相同年級的學生納入研究，探討彼此之間是否存在差異。

綜上之結論提出以下之建議，本研究使用 Robots city 桌上遊戲，讓學習者可以透過遊戲式學習培養運算思維的概念，並搭配手機應用程式，以程式設計的邏輯控制機器人的行為，藉以分析學習者在研究過程中的行為，所以未來的研究可以嘗試使用別種科技工具，也建議可以與不同科目或導入不同的教學策略，分析學習者在過程中會不會有其他的行為達到顯著差異，還可以進一步探討學習者的學習成效，如此一來更能夠幫助學習者進行更有效的學習，也可以幫助教師找到更有效的教學策略，提高教學品質。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 108-2511-H-003 -056 -MY3 的部分補助。

7. 參考文獻

Acquah, E. O., & Katz, H. T. (2020). Digital Game-based L2 Learning Outcomes for Primary through High-School Students: A Systematic Literature Review. *Computers & Education, 143*, 103667. doi: https://doi.org/10.1016/j.compedu.2019.103667

Bakeman, R., & Gottman, J. M. (1997). *Observing Interaction: An Introduction to Sequential Analysis.* Cambridge University Press.

Barab, S. A., Scott, B., Siyahhan, S., Goldstone, R., Ingram-Goble, A., Zuiker, S. J., & Warren, S. (2009). Transformational Play as a Curricular Scaffold: Using Videogames to Support Science Education. *Journal of Science Education and Technology, 18*(4), 305. doi:10.1007/s10956-009-9171-5

Barr, V., & Stephenson, C. (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), 48-54.

Caillois, R. (1957). *Les Jeux et les Hommes.* Paris: Gallimard.

Chang, S. P., Hou, H.T., Sung, Y,T., & Chang, K. E. (2014). Applying Sequential Analysis to Teaching Methods: Case Study of a CSL Classroom. *International Journal of Research in Social Sciences, 4*(1), 1-16.

Dichev, C., & Dicheva, D. (2017). Gamifying Education: What is Known, What is Believed and What Remains Uncertain: A Critical Review. *International Journal of Educational Technology in Higher Education, 14*(1), 9. doi:10.1186/s41239-017-0042-5

Ebner, M., & Holzinger, A. (2007). Successful Implementation of User-Centered Game Based Learning in Higher Education: An Example from Civil Engineering. *Computers & Education, 49*(3), 873-890. doi: https://doi.org/10.1016/j.compedu.2005.11.026

Google. (2016). *Computational Thinking Concepts Guide.* Retrieved July 19, 2016, from http://computationalthinking.pbworks.com/w/file/fetch/108605812/ComputationalThinkingConceptsGuide.pdf

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

Hew, K. F., Huang, B., Chu, K. W. S., & Chiu, D. K. W. (2016). Engaging Asian Students through Game Mechanics: Findings from two Experiment Studies. *Computers & Education, 92-93*, 221-236. doi: https://doi.org/10.1016/j.compedu.2015.10.010

Hogle, J. G. (1996). *Considering Games as Cognitive Tools: In Search of Effective "Edutainment".* ERIC Clearinghouse.

Joe Cuseo, J. B. (1992). Cooperative Learning vs. Small-Group Discussions and Group Projects: The Critical Differences. *Cooperative Learning and College Teaching, 2*(3), 4-9.

Johnson, D. W., Johnson, R. T., & Holubec, E. J. (1994). *The New Circles of Learning: Cooperation in the Classroom and School.* ASCD.

Kalelioglu, F. G., Yasemin; Kukul, Volkan. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing, 4*(3), 583-596.

Lin, C.-H., & Davidson-Shivers, G. V. (1996). Effects of Linking Structure and Cognitive Style on Students' Performance and Attitude in a Computer-Based Hypertext Environment. *Journal of Educational Computing Research, 15*(4), 317-329. doi:10.2190/JU82-YHCA-X5DR-EHYU

Munir, M. T., Baroutian, S., Young, B. R., & Carter, S. (2018). Flipped Classroom with Cooperative Learning as a Cornerstone. *Education for Chemical Engineers, 23*, 25-33. doi: https://doi.org/10.1016/j.ece.2018.05.001

Nattiv, A. (1994). Helping Behaviors and Math Achievement Gain of Students Using Cooperative Learning. *The Elementary School Journal, 94*(3), 285-297. doi:10.1086/461767

Parker, R. E. (1985). Small-Group Cooperative Learning-Improving Academic, Social Gains in the Classroom. *Nass Bulletin, 69*, 48-57.

Sackett, G. P. (1980). Lag Sequential Analysis as a Data Reduction Technique in Social Interaction Research. Exceptional infant. *Psychosocial Risks in Infant-Environment Transactions, 4*.

Selby, & Collins, C. (2013). *Computational Thinking: The Developing Definition.* University of Southampton

Simões, J., Redondo, R. D., & Vilas, A. F. (2013). A Social Gamification Framework for a K-6 Learning Platform. *Computers in Human Behavior, 29*(2), 345-353. doi:https://doi.org/10.1016/j.chb.2012.06.007

Slavin, R. E. (1985). Cooperative Learning: Applying Contact Theory in Desegregated Schools. *Journal of Social Issues, 41*(3), 45-62. doi:10.1111/j.1540-4560.1985.tb01128.x

Swaid, S. I. (2015). Bringing Computational Thinking to STEM Education. *Procedia Manufacturing, 3*, 3657-3662. doi: https://doi.org/10.1016/j.promfg.2015.07.761

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*, 33-35.

Wing, J. M. (2010). Computational Thinking: What and Why? Retrieved November 17, 2010, from https://pdfs.semanticscholar.org/628a/da255c83abfee8693132310cba2ccfaed5a6.pdf?_ga=2.10897140.1106366303.1576479702-448492393.1576479702

Zhao, W., & Shute, V. J. (2019). Can Playing a Video Game Foster Computational Thinking Skills? *Computers & Education, 141*, 103633. doi: https://doi.org/10.1016/j.compedu.2019.103633

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking and Non-formal Learning

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Implementing a Computational Thinking Curriculum with Robotic Coding Activities through Non-formal Learning

Poh-tin LEE[1*], Chee-wah LOW[2]
[1,2] Bukit View Secondary School, Singapore
lee_poh_tin@moe.edu.sg, low_chee_wah@moe.edu.sg

ABSTRACT
This paper shares the implementation of a robotic coding curriculum for the students to develop Computational Thinking skills through non-formal learning at a secondary school in Singapore. These after-school activities are implemented for students who are members of the school's Infocomm Club. The students learn to program the robotic balls using block-based coding and apply problem solving skills in their projects using recycled materials for green environment. The projects are also designed for the students to apply Mathematics and Science concepts.

KEYWORDS
non-formal learning, coding, computational thinking, curriculum, implementation.

1. INTRODUCTION
At the Bukit View Secondary School, 38 students of the Infocomm Club are between age 12 and 17 years old. These students acquire Computational Thinking skills (Wing, 2006) through non-formal learning in the after-school activities (Lee et al., 2019).

The Infocomm Club runs various programmes for the students to learn coding such as Scratch programming (Maloney et al., 2010), Python programming (Rashed & Ahsan, 2012) and MIT's App Inventor (Wagner et al., 2013). A new Robotic Ball Coding Programme has been implemented to excite the students through coding of the Sphero balls (www.sphero.com) using block programming.

2. RATIONALE FOR USING ROBOTIC BALLS
There are various electronics platform available for the teaching of coding to infuse Computational Thinking skills. In the new programme, teachers of the school's Infocomm Club facilitate the students to code on robotic ball as it comes with built-in sensors such as accelerometer (measure motion), gyroscope (measure tilt angles), light sensor (measure luminosity), infrared sensor (measure relative distance between robotic balls) and compass sensor (measure orientation in real-world directions).

Other microprocessor boards usually require motors and wheels to be attached for movement. With robotic balls, the students can now focus on coding activities to move or rotate these balls without other hardware accessories.

3. THE ROBOTIC BALL CODING CURRICULUM
Under the Robotic Ball Coding Programme, the students learn through activities which make use of the built-in sensors of the Sphero robotic balls such as the accelerometer sensor, gyroscope sensor and control its sound and LED lights. The students also create prototypes such as maze and tractor vehicles using recycled materials for green environment including card boards, ice-cream sticks and paper cups. Table 1 shows the topics and activities of the 6-week Robotic Ball Coding Curriculum with projects on Music, Mathematics and Science.

Table 1. Robotic Ball Coding Curriculum.

Week	Topic	Activity
Week 1	Introduction and Loop Statements	Navigate the Robotic Ball through a maze.
Week 2	Variables and Conditional if-else Statements	Create games with the built-in sensors.
Week 3	More fun with if-else Statements	Create a futuristic Robotic Ball using the Accelerometer Sensor.
Week 4	Mathematics Project	Control the LED lights based on the Gyroscope Sensor's axes of rotation.
Week 5	Music Project	Synchronize the Robotic Ball dancing with a song.
Week 6	Science Project	Build a tractor vehicle and explore force and motion.

4. IMPLEMENTATION OF THE ROBOTIC BALL CODING CURRICULUM
The students learn to program the robotic balls using Sphero Edu App installed on the iPads. This app allows students to code through Draw Programming, Block Programming and Text Programming using Javascript (Sphero Edu, 2019).

As the Infocomm Club comprises of both junior and senior members, the students are taught the Block-based Programming (Kelleher & Pausch, 2005; Weintrop & Wilensky, 2017) which is easier to learn than Text Programming. A block program code using the Sphero Edu App is shown in Figure 1.

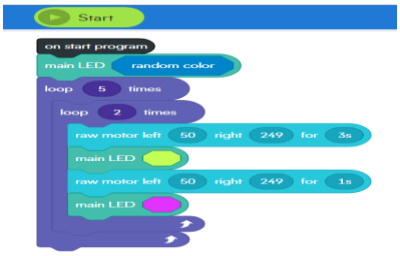


Figure 1. Block program code using Sphero Edu App.

To infuse more elements of fun, a Sphero Race Competition is held and the students are required to code their robotic balls to move through a race course where speed, inertia and obstacles have to be taken into account. After each stage, the students are allowed to improve their program code. Figure

2 shows the final stage of the race with 2 Sphero balls in the competition.

Figure 2. Final stage of the Sphero Race Competition.

5. SURVEY RESULTS

After the 6-week Robotic Ball Coding Programme, a survey was conducted for the 38 students of the Infocomm Club. 95% of the students enjoyed coding activities on robotic balls. 87% of the students have expressed that they can develop Computational Thinking skills to solve real-world problems as shown in Figure 3. Similarly, 87% of the students also expressed that they can apply Mathematics and Science concepts in the coding activities as shown in Figure 4. Some students have faced challenges in testing and debugging the errors in their programs.

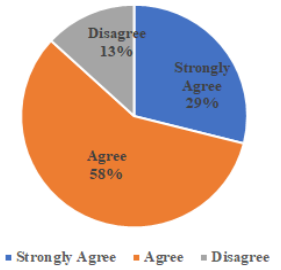


Figure 3. Survey Question 1: I can develop Computational Thinking to solve real-world problems with robotic balls.

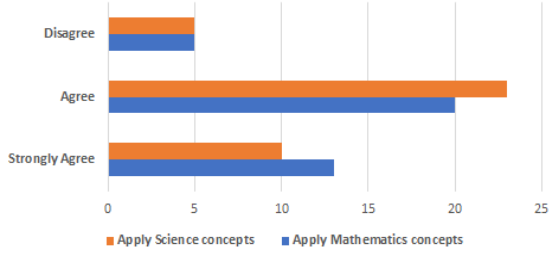


Figure 4. Survey Question 2: I can apply Mathematics and Science concepts in the coding activities.

At the end of the programme, some students gave the following feedback:

"I am able to use coding to control the ball."

"I like the coding when the balls start dancing."

"I can apply Mathematics and Science in the coding."

"I like making the ball move to a light source."

"I can use the raw motors to make the balls bounce like crazy."

6. CONCLUSION

This paper shares the rationale, curriculum and implementation of Computational Thinking with robotic coding activities on Sphero balls through non-formal learning at the school's Infocomm Club. The 6-week programme enables students to develop Computational Thinking through block-based coding with built-in sensors and create prototypes using recycled materials. The survey results show that the students are motivated as they find coding with Sphero balls to be fun and they could apply Science and Mathematics concepts in their projects. Future study will explore on coding with projects that involve integration of knowledge from various subject matters.

7. REFERENCES

Kelleher, C., & Pausch, R. (2005). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys (CSUR), 37*(2), 83-137.

Lee, P. T., Lee, X. R., Low, C. W., & Kokila, A. (2019). Implementing Computational Thinking through Non-formal Learning in after School Activities at Students Society Club. *Proceedings of the International Conference on Computational Thinking Education 2019*, 201-202.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education, 10*(4), 16.

Rashed, M. G., & Ahsan, R. (2012). Python in Computational Science: Applications and Possibilities. *International Journal of Computer Applications, 46*(20), 26-30.

Sphero Edu. (2019). *Programming*. Retrieved Nov 1, 2019, from https://support.sphero.com/article/cptfh361pk-programming-with-sphero-edu

Wagner, A., Gray, J., Corley, J., & Wolber, D. (2013). Using App Inventor in a K-12 Summer Camp. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 621-626.

Weintrop, D., & Wilensky, U. (2017). Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. *ACM Transasctions on Computing Education 18*(1), 3.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

General Submission to Computational Thinking Education

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Investigating the Effects of Gender and Scaffolding tools on the Development of Preschooler's Computational Thinking

Kyriakoula GEORGIOU[1*], Charoula ANGELI[2],
[1,2] University of Cyprus, Cyprus
georgiou.kyriakoula@ucy.ac.cy, cangeli@ucy.ac.cy

ABSTRACT
A large body of literature emphasizes the importance of effective integration of computational thinking at preschool education (Ching, Hsu, & Baldwin, 2018) as it is enlisted in the 21st century skills (Lye & Koh, 2014). Nonetheless, the factors related to the development of computational thinking are under investigation (Román-González, Pérez-González, Moreno-León, & Robles, 2018). Consequently, the study herein investigated the impact of scaffolding and gender in the development of one hundred and eighty children's computational thinking. The results indicated strong interaction ($p<0.000$) between the aforementioned factors and the advancement of computational thinking producing practical suggestions for the preschool educators and the computing community in general.

KEYWORDS
computational thinking, scaffolding tools, gender, young children, robotics

1. INTRODUCTION
Science, technology, engineering and mathematics (STEM) are the cornerstones of our society that upon them its healthy development is constructed (Chabbott & Ramirez, 2000). However, there is an oppressive shortage of human resources in the aforementioned areas and simultaneously a declining trend in the number of students choosing STEM courses (Bøe, Henriksen, Lyons, & Schreiner, 2011). In addition it is predicted that by 2020, the 50% of STEM jobs will be in computing (ACM Pathways Report, 2013). Computational thinking is a fundamental concept of computer science emerging from its basic principles and practices (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013) while at the same time improves computing education since it derives methods from different disciplines (Guzdial, 2008). In environments where computation thinking has been used as a tool for learning STEM content it has been shown to synergistically deepen learning of the STEM topics and computing concepts (Sengupta et al., 2013).

Computational thinking is being described as a key set of skills (Guzdial, 2008; Wing, 2008) involved in problem solving (Bocconi et al., 2016). Its core elements are: abstraction, generalization, decomposition, algorithmic thinking and debugging (detection and correction of errors) (Angeli et al., 2016). It is discussed in the computing community that is of great importance the development of computational thinking to be realized within school contexts and furthermore to be integrated in the curricula (Grover & Pea, 2018). Although several empirical studies have been conducted studying the development of computational thinking in elementary and high school settings however, the research area of the development of computational thinking in preprimary education is still in its infancy (Bers, Flannery, Kazakoff, & Sullivan, 2014).

The teaching and the development of computational thinking especially in the early childhood education is mainly being implemented with the use of the robotics (Bers et al., 2014). Recent studies support the introduction of robotics in preprimary education since they reported that the active manipulation of the various robotics tools can enhance the learning experience of the children. In addition the use of robotics can advance the development of cognitive skills (Papert, 1980); social skills and engineering design skills (Bers, 2008).

Programming is theorized as a teaching approach interwoven with the learning of robotics (Papert, 1980) supporting the implementation of cognitive tasks directly correlated to the development of computational thinking (Lye & Koh, 2014). in this study the design pattern of Papert (1993) "low floor and high ceiling" was embraced which is considered suitable for programming educational robots (Resnick & Silverman, 2005).

Among the contributing factors that are directly connected to the development of computational thinking is gender since there are consistent findings in the literature that support the claim that gender differences influence student learning (Duckworth & Seligman, 2006) and school achievements (Sousa & Tomlinson, 2011). More specifically neuroscience studies recite that these differences are interwoven with the fact that girls' and boys' brain have morphological variances resulting to more cortical areas devoted to verbal functioning and visual-spatial information processing respectively (Baron-Cohen, 2004). Accordingly girls are better at verbal and sensory memory and boys at visual memory (Bonomo, 2011), justifying the fact that girls are excelling in complex tasks of reading and writing whereas boys in tasks which involve mental rotation (Maeda & Yoon, 2013).

Another factor which is scrutinized in the present study is scaffolding. It is well documented in the literature that the use of scaffolding is imperative in education especially, when learning is accompanied by technological tools (Azevedo & Hadwin, 2005). Moreover scaffolding provision is essential especially for young students (Belland, 2014) since in its absence students may fail to complete the task (Van de Pol, Volman, & Beishuizen, 2010). Studies connected scaffolding with the theory of the cognitive load due to the fact that scaffolding tools support the reduction of cognitive load that is being imposed to student during learning (Myhill & Warren, 2005) while at the same they improve the acquisition of cognitive skills (Reid-Griffin & Carter, 2004).

2. RESEARCH PURPOSE

Very little research has been conducted exploring gender differences and the impact of different types of scaffolding tools in young children's robotics and programming abilities (Angeli & Valanides, 2019; Sullivan & Bers, 2013) most likely because the use of robotics and programming in early childhood classrooms is relatively new.

The research aim is two folded as it focus on investigating the effects of different scaffolding tools on children's computational thinking in preprimary education and at the same time it examines whether the two different types of scaffolding tools have a different impact on boys' and girls' performance on the scores of computational thinking.

3. THEORITICAL BACKGROUND

3.1. Participants

The participants were one hundred and eighty preschoolers, ranging in mean age from five to six years old. The researchers obtained written consent from their parents to participate in the study.

3.2. Research Materials

3.2.1. Problem-solving Tasks

Three different problem-solving tasks, corresponding to three different research phases, were designed for the children to program and direct the Bee-Bot into different paths. Children had twenty minutes at their disposal to complete each problem-solving task. The first problem-solving task engaged children into an exploration of the commands of the programming language of the Bee-Bot and it consisted of thirteen subtasks. The second task consisted of five subtasks aiming to teach children how to formulate sequences of commands in increasing levels of complexity. Finally the third task comprised of five subtasks that were used to evaluate children's computational thinking.

3.2.2. Modeling-Based Scaffolding

This scaffolding tool is a representation of the floor mat, the robotic toy Bee-Bot and the programming commands all in reduced size. The child thought about the algorithm and constructed a representation of it using the model that was used to support his/hers endeavor to guide the Bee-Bot into the task's path.

3.2.3. Code Structure-Based Scaffolding

This type of scaffolding included small laminated cards representing each of the Bee-Bot commands and a larger laminated card and was developed to simulate the way the code is being written while programming. For this reason the participants were asked to choose the cards and attached them in the larger card in the order they believed it was the correct one. With this way they formed a sequence of commands that visualize the algorithm and then tested it.

3.3. Research Procedures

Research procedures consisted of three research phases that were administered in three consecutive days. All of the research phases were conducted individually for each participant. The first day, during Phase 1, all the children became acquainted with the basic commands of the Bee-Bot and small sequences of commands.

On the following day, during Phase 2, the children were randomly divided into three equivalent groups as shown in Table 1. In the first experimental group, children used the modeling-based scaffolding tool, while in the second experimental group, they used the code structure-based scaffolding tool. The last group of children constituted the control group where they worked with no scaffolding tool. During this phase, children learned small codes that comprised sequences of commands with a minimum length of four commands and a maximum of seven. Children were evaluated for their initial attempts to solve the problem solving task. More specifically, children developed a sequence of commands and used it by pressing the corresponding buttons. Then, they observed which path Bee-Bot would follow and if the path was not correct, they had the opportunity to try again. During the last phase, Phase 3, the scaffolding tools were withdrawn and children's performance was assessed while trying to carry out the third problem-solving task.

Table 1. Participants' Distribution into the Two Experimental Groups and the Control Group According to Their Gender

Groups	Participants		
Control Group	60	Boys	Girls
		37	23
Model-Based Scaffolding	60	Boys	Girls
		35	25
Code-Based Scaffolding	60	Boys	Girls
		26	34
Total	180	96	82

3.4. Data Analysis

This study used a total of one hundred and eighty hours of video data. The entire process of the individualized instruction that resulted from children's interactions with the Bee-Bot was videotaped, transcribed and analyzed over one year period. Many researchers propose various software for coding recorded data however their use was not applicable in the present study. The reason for this is that the human interpretation process of the data was deemed necessary in this research since the robotic device (Bee-Bot) that was used in the herein study, is designed to support a playful learning process (Bers et al., 2014) and in such learning environments, children's actions are coded by researchers (Basu, Biswas, & Kinnebrew, 2017). Consequently, the researchers had to observe the videotaped videos and record the actions of the children corresponding to the command's choices. Specifically, the researcher recorded which buttons the children selected in their various attempts to solve the problem of each teaching intervention. Following, the research data were analyzed using the method of process coding (Saldaña, 2015), which is considered to be ideal when the observed actions of the participants include problem solving (Corbin & Strauss, 2008). At first, four videos from each group were coded from two researchers to ensure validity and afterwards researchers' coded videos independently.

4. Results

4.1. Computational Thinking Assessment Rubric

The researchers collected data from all the one hundred and eighty students for each problem-solving task and then identified whether students solved the tasks correctly on their first attempt or whether they required more attempts. Based on the analysis, a rubric was created that scores students' total effort along two aspects: (a) number of attempts and (b) the ability to complete the tasks step by step.

4.2. Computational Thinking

The picture emerging from the descriptive statistics shown in Table 3 indicates an advantage of male participants. In all groups, during the initial and final assessment of the computational thinking in Phase 2 and Phase 3, boys seem to outperform girls. A 2 X 3 analysis of variance was conducted to determine whether there was statistically significant difference between boys and girls on the different forms of scaffolding strategies during the assessment of computational thinking in Phase 2. The results revealed that only the use of scaffolding tool ($F (2, 179) = 49.26$, $p < 0.000$) was statistically significant for the scores of computational thinking. In order to detect the differential performance on the computational thinking regarding to scaffolding tools, the researchers performed post-hoc LSD comparisons. The results showed that both modeling-based scaffolding and code structure-based scaffolding outperformed the control group.

Table 2. Descriptive Statistics of Children's Computational Thinking in Phase 2 for each Scaffolding tool and Gender

Research Phase2			
	Mean	SD	N
Modeling-Based Scaffolding			
Girls	246,08	17,94	25
Boys	239,40	43,81	35
Total	485,48	61,75	60
Code Structure-Based Scaffolding			
Girls	226,11	39,75	34
Boys	230,13	22,92	26
Total	456,24	62,67	60
Working without Scaffolding (Control Group)			
Girls	156,60	57,11	23
Boys	180,21	44,00	37
Total	336,81	101,11	60

During the third research phase boys outperformed girls in all groups (Table 4). In addition, the children who belonged in the control group scored higher than the children who belonged in the two scaffolding groups. A 2 X 3 analysis of variance was conducted to investigate the differences between boys and girls and the different forms of scaffolding strategies used in the previous research phase. The findings showed that only gender had a significant main effect ($F (1, 179) = 12.82$, $p < 0.000$) in the computational thinking score, revealing that the intervention produced significantly higher gains for the male participants.

Table 3. Descriptive Statistics of Children's Computational Thinking in Phase 3 for Each Scaffolding tool and Gender

Research Phase2			
	Mean	SD	N
Modeling-Based Scaffolding			
Girls	164,60	41,05	25
Boys	202,09	58,91	35
Total	366,69	99,96	60
Code Structure-Based Scaffolding			
Girls	168,61	57,73	34
Boys	195,76	53,86	26
Total	364,37	111,59	60
Working without Scaffolding (Control Group)			
Girls	175,86	52,01	23
Boys	206,59	47,02	37
Total	382,45	99,03	60

5. DISCUSSION

Interventions that are being implemented with the use of robotics and contemplate the development of computational thinking have become increasingly popular within the school system (Grover & Rea, 2018). This study brings into focus a large contributor to the discussion of how to integrate the development of computational thinking in preprimary education, a notion affecting the computing community in general. In this study the authors investigated and documented gender differences in educational robotics instruction. Unlike Sullivan and Bers (2013) that reported no gender differences regarding the performance on robotics and on the development of computational thinking of young children respectively, the findings of the herein study are in line with findings of the studies of Angeli and Valanides (2019) and Román-González et al. (2018) that reported that boys outperformed girls during the assessment of the development of the computational thinking.

This result could be justified by a range of factors that are studied in the study herein. The gender disparities on the development of computational thinking might be related to the spatial ability of the participants, since the majority of this study's problem-solving tasks required the formation of sequences of commands that comprised the spatial referents "left" and "right". Researchers cited that especially in tasks that involve mentally rotation of figures (Maeda & Yoon, 2013), that the stereotype threats are often particularly noticeable for female, the task's performance may be attributed to a lack of ability. Mental rotation requires the operation of visual-spatial working memory (Hyun & Luck, 2007) which is being influenced by the cerebral cortex and is larger in boys than girls supporting the fact that boys' learning is improved through visual-motor experiences (Bonomo, 2011). Indeed, some studies have shown evidence that males, with their better visual-spatial working memory, are likely to perform better in visual-motor tasks than girls (e.g. Maeda & Yoon, 2013) resulting to this study's observed male advantage on task's performance.

Alternatively, another possible interpretation of the strong effect of gender in our data might be related to the scaffolding tools used for the development of the computational thinking. More specifically the modeling and code structure-based scaffolding tools may have contributed to a lack of engagement of the female participants resulting to their lower performance on the problem solving tasks in comparison with their male counterparts. A different type of scaffolding tool including storytelling activities might have showed different results (Kelleher, Pausch, & Kiesler, 2007). More precisely girls that used storytelling showed more evidence of engagement with programming and expressed greater interest in future use of coding than girls

who did not have storytelling support. Findings obtained by Angeli and Valanides (2019) report the importance of gender oriented scaffolding tools. In their study, higher means in the computational thinking scores were found in the male and female group that used manipulative-based and collaborative writing activity respectively.

No significant differences were found between female and male attrition from robotics activities during Phase 2. However, significant differences were found between the experimental groups and the control group: in scaffolding salient condition the performance of children was substantially better from the performance of children in the control condition. Specifically children who had been provided with scaffolding outperformed children that had no scaffolding. These results are consistent with previous work by Jonassen (1992) and more contemporary work by Angeli and Valanides (2004) that showed the necessity of scaffolding techniques, such as, external memory systems to facilitate students' learning with technological tools. In addition the results of the herein study are collinear with the results of studies that outlined that necessity of scaffolding especially when students of preschool education use technological tools (Azevedo & Hadwin, 2005) since the cognitive load that is being imposed to students during learning is reduced (Van Merriënboer, Kirschner, & Kester, 2003).

The observed low scores of the children of the control group on the measurement of computational thinking, during the second research phase, are caused from their difficulties that they encountered while visualizing the procedure needed to execute a program (Fessakis, Gouli, & Mavroudi, 2013). Children's problems with the visualization of the commands sequences can be attributed to children's misconceptions situated in the mental rotation (Sarama & Clements, 2009). More specifically children are not able to correctly discriminate their left and right body parts; use and apply the word left and right; label the directions correctly as 'left' or 'right' (Sarama & Clements, 2009). However children with appropriate scaffolding can understand and use the concept of left and right correctly (Shusterman & Spelke, 2005) while being engaged in activities that include concepts strongly correlated with the rotation such the use of commands that directed the floor robot to turn right or left. Therefore the use of the scaffolding tools materialized the visualization of the algorithm used to program the floor robot and supported the learning of computational practices (Lye & Koh, 2014) that enabled children to excel in the problem-solving learning environment.

These findings have a number of implications of both theoretical and practical significance. Regarding the theoretical significance, this study contributes to the literature addressing gender effects on computational thinking achievement by examining the contribution of scaffolding tools on children's computational thinking development during preprimary education. This study extends previous findings in gender differences in visual spatial memory providing additional data indicating that gender differences in visual spatial working memory appears also in younger ages.

Despite the fact that adults can use visualization effectively in many tasks (Wohldmann, Healy, & Bourne, 2007) it is clear that this strategy is not available for children resulting to an incensement of their cognitive load. The role of scaffolding in educational robotic settings should be communicated since scaffolding assist students to successfully complete a complex task (Belland, 2014).

It has been reported that early childhood educators lack of competence and confidence while teaching robotics therefore they need training and resources (Bers, Seddighin, & Sullivan, 2013). Therefore, in regards with the practical implications, this study provides insights integrating computational thinking with the use of robotics into teaching practices of preschool education verifying the effectiveness of scaffolding tools as an instructional design framework for the development of computational thinking. At the same time this study contributes to the resources for professional development which are considered crucial for the curricular changes.

Furthermore the herein results document that engaging children into problem solving tasks with robotics, constitutes a beneficial instructional method that advances computational thinking in early childhood settings. There is a great necessity to design environments that encourage and enhance computational thinking from a young age through meaningful playing. By introducing robotics activities that include problem solving to the early education curriculum, the play experiences of the children can be enhanced.

In conclusion, the authors in the herein study accomplished to: (a) integrate computational thinking into the learning of programming with robotics, (b) propose a set of learning activities that provide low-high ceiling problem solving tasks at preschool level and (c) advocate the use of specific scaffolding tools for supporting the development of computational thinking.

6. LIMITATIONS AND FUTURE DIRECTIONS

Our analyses provide critical insight into the association between the trend of gendered attrition with robotics activities and the development of computational thinking. Nonetheless, a number of limitations should also be considered. The findings obtained during the last research phase, when the scaffolding tools were withdrawn, reported that the differences on the scores on the assessment of the computational thinking among the experimental and control groups were not statistically significant. This result can be attributed to a number of reasons. Firstly, the duration and the number of the lessons proved to be inadequate to enable the transfer of knowledge as other researchers concur (e.g Bers et al., 2014). In regards of children's computational thinking development, it has been established by researchers that developing cognitive skills in young children requires sustained and immersive effort (Bers et al., 2014). Lastly, to trigger the augmentation of the pedagogical gains of the scaffolding is essential that the scaffolding to gradually fade out (Van de Pol et al., 2010). Therefore future research effort should focus on expanding the duration of the interventions.

While prior research has shown gender variation in computational thinking task's performance at elementary and high school level showing relative advantages for male students (e.g. Atmatzidou & Demetriadis, 2016) little is known about which cognitive strategies are directly linked with gender differences in attrition with robotics activities and most importantly, what factors contribute to female fully engaged in robotics activities. In addition gaining a better understanding and addressing the underlying causes of gender disparities to the development of young children's computational thinking will likely require focusing on different skills of computational thinking: abstraction and debugging.

Another possibility requiring further exploration is whether particular groups of children benefit more substantially from interventions that include a focus on their cognitive style. In this context, robust research that can shed further light on the relationship of young children's cognitive style is needed (Georgiou & Angeli, 2019).

A dimensional approach will be of interest in future research investigating different levels of competence - for example, whether gender-related attitudes are associated with computational thinking skills or whether a threshold effect is observed. These questions have important implications for formulating and evaluating interventions targeting to advance computational thinking. Future intervention research should also test the mechanisms through which any effect of positive computational thinking growth on learning occurs. For example whether gender disparities impact on the development of computational thinking via social pathways such as teacher-student interactions.

7. REFERENCES

Angeli, C., & Valanides, N. (2004). The Effect of Electronic Scaffolding for Technology Integration on Perceived Task Effort and Confidence of Primary Student Teachers. *Journal of Research on Technology in Education*, *37*(1), 29-43.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Journal of Educational Technology & Society*, *19*(3), 45-47.

Angeli, C., & Valanides, N. (2019). Developing Young Children's Computational Thinking with Educational Robotics: An Interaction Effect between Gender and Scaffolding Strategy. Retrieved August 10, 2019, from https://www.sciencedirect.com/science/article/pii/S0747563219301104?via%3Dihub

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.

Azevedo, R., & Hadwin, A. F. (2005). Scaffolding Self-Regulated Learning and Metacognition–Implications for the Design of Computer-based Scaffolds. *Instructional Science*, *33*(5), 367-379.

Baron-Cohen, S. (2004). *The Essential Difference*. London: Penguin.

Basu, S., Biswas, G., & Kinnebrew, J. S. (2017). Learner Modeling for Adaptive Scaffolding in a Computational Thinking-based Science Learning Environment. *User Modeling and User-Adapted Interaction*, *27*(1), 5-53.

Belland, B. R. (2014). Scaffolding: Definition, Current Debates, and Future Directions. In *Handbook of Research on Educational Communications and Technology*. NY: Springer, 505-518.

Bers, M. U. (2008). *Blocks, Robots and Computers: Learning about Technology in Early Childhood*. New York: Teacher's College Press.

Bers, M., Seddighin, S., & Sullivan, A. (2013). Ready for Robotics: Bringing Together the T and E of STEM in Early Childhood Teacher Education. *Journal of Technology and Teacher Education*, *21*(3), 355-377.

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.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education-Implications for Policy and Practice (No. JRC104188)*. Joint Research Centre.

Bøe, M. V., Henriksen, E. K., Lyons, T., & Schreiner, C. (2011). Participation in Science and Technology: Young People's Achievement related Choices in Late Modern Societies. *Studies in Science Education*, *47*(1), 37-72.

Bonomo, V. (2010). Gender Matters in Elementary Education: Research-Based Strategies to Meet the Distinctive Learning Needs of Boys and Girls. *Educational Horizons*, *88*(4), 257-264.

Chabbott, C., & Ramirez, F. O. (2000). Development and Education. In Maureen & Hallinan (Ed.), *Handbook of the Sociology of Education*. New York, NY: Springer.

Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing Computational Thinking with Educational Technologies for Young Learners. *TechTrends*, *62*(6), 563-573.

Corbin, J., & Strauss, A. (2008). Strategies for Qualitative Data Analysis. Basics of Qualitative Research. *Techniques and Procedures for Developing Grounded Theory*, *3*, 65-86.

Duckworth, A. L., & Seligman, M. E. (2006). Self-discipline Gives Girls the Edge: Gender in Self-discipline, Grades, and Achievement Test Scores. *Journal of Educational Psychology*, *98*(1), 198-208.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem Solving by 5–6 years old Kindergarten Children in a Computer Programming Environment: A Case Study. *Computers & Education*, *63*, 87-97.

Georgiou, K., & Angeli, C. (2019). Developing Preschool Children's Computational Thinking with Educational Robotics: The Role of Cognitive Differences and scaffolding. *Proceedings of the 16th International*

Conference on Cognition and Exploratory Learning in Digital Age. Cagliari, Italy: IADIS Press, 101-108.

Grover, S., & Pea, R. (2018). *Computational Thinking: A Competency Whose Time Has Come. Computer Science Education: Perspectives onTeaching and Learning in School.* London: Bloomsbury Academic.

Guzdial, M. (2008). Education: Paving the Way for Computational Thinking. *Communications of the ACM, 5*1(8), 25-27.

Hyun, J. S., & Luck, S. J. (2007). Visual Working Memory as the Substrate for Mental Rotation. *Psychonomic Bulletin & Review, 14*(1), 154-158.

Jonassen, D. H. (1992). *What are Cognitive Tools?* In D. H. Jonassen (Ed.), *Cognitive Tools for Learning*. Berlin: Springer, 1-6.

Kelleher, C., Pausch, R., Pausch, R., & Kiesler, S. (2007). Storytelling Alice Motivates Middle School Girls to Learn Computer Programming. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1455-1464.

Lye, S. Y., & Koh, J. H. L. (2014). Review on Teaching and Learning of Computational Thinking through Programming: What is Next for K-12? *Computers in Human Behavior, 41*, 51-61.

Maeda, Y., & Yoon, S. Y. (2013). A Meta-analysis on Gender Differences in Mental Rotation Ability Measured by the Purdue Spatial Visualization Tests: Visualization of Rotations (PSVT: R). *Educational Psychology Review, 25*(1), 69–94.

Myhill, D., & Warren, P. (2005). Scaffolds or Straitjackets? Critical Moments in Classroom Discourse. *Educational Review, 57*(1), 55-69.

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

Papert, S. (1993). *The Children's Machine: Rethinking School in the Age of the Computer*. New York, NY: Basic Books.

Reid-Griffin, A., & Carter, G. (2004). Technology as a Tool: Applying an Instructional Model to Teach Middle School Students to Use Technology as a Mediator of Learning. *Journal of Science Education and Technology, 13*(4), 495-504.

Resnick, M., & Silverman, B. (2005). Some Reflections on Designing Construction Kits for Kids. *Proceedings of the 2005 Conference on Interaction Design and Children*. ACM, 117-122.

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Extending the Nomological Network of Computational Thinking with Non-cognitive Factors. *Computers in Human Behavior, 80*, 441-459.

Saldaña, J. (2015). *The Coding Manual for Qualitative Researchers.* London: Sage.

Sarama, J., & Clements, D. H. (2009). *Early childhood Mathematics Education Research: Learning Trajectories for Young Children.* New York, NY: Routledge.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating Computational Thinking with K-12 Science Education Using Agent-based Computation: A Theoretical Framework. *Education and Information Technologies, 18*(2), 351-380.

Shusterman, A., & Spelke, E. S. (2005). Language and the Development of Spatial Reasoning. In P. Carruthers, S. Laurence & S. Stitch (Eds.), *The Innate Mind: Structure and Content*. New York, NY: Oxford University Press, 89 –108.

Sousa, D. A., & Tomlinson, C. A. (2011). *Differentiation and the Brain: How Neuroscience Supports theLearner-friendly Classroom.* Bloomington, IN: Solution Tree Press.

Sullivan, A., & Bers, M. U. (2013). Gender Differences in Kindergarteners' Robotics and Programming Achievement. *International Journal of Technology and Design Education, 23*(3), 691-702.

Van de Pol, J., Volman, M., & Beishuizen, J. (2010). Scaffolding in Teacher–student Interaction: A Decade of Research. *Educational Psychology Review, 22*(3), 271-296.

Van Merriënboer, J. J., Kirschner, P. A., & Kester, L. (2003). Taking the Load off a Learner's Mind: Instructional Design for Complex Learning. *Educational Psychologist, 38*(1), 5-13.

Wing, J.M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366* (1881), 3717-3725.

Wohldmann, E. L., Healy, A. F., & Bourne Jr, L. E. (2007). Pushing the Limits of Imagination: Mental Practice for Learning Sequences. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 33*(1), 254.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Integrating Computational Thinking in K-12 Education: Exploring Digital Fabrication Activities through CTPACK Framework

Megumi IWATA[1*], Jari LARU[2*], Kati MÄKITALO[3*], Kati PITKÄNEN[4*]
[1,2,3,4] Faculty of Education, University of Oulu, Finland
megumi.iwata@student.oulu.fi, jari.laru@oulu.fi, kati.makitalo@oulu.fi, kati.pitkanen@student.oulu.fi

ABSTRACT
This paper presents the preliminary study of integrating computational thinking (CT) into K-12 education. In order to successfully integrate CT into school curriculum, we need to enhance teachers' understanding on CT. In this paper, we explore the possibilities for widening teacher's CT understanding by merging CTPACK framework, which combines CT in technological pedagogical content knowledge (TPACK). Aim of the study is to understand how CT intersects with elements of TPACK in the context of ill-structured digital fabrication activities. We examined three cases where 7th-9th grade students visited a makerspace as part of school curriculum. Through interviews and observations, we found that CT was interconnected with technological knowledge and pedagogical knowledge highlighting the use of advanced technologies and pedagogical propositions of the context, learning by doing. We also found vague connections between CT and content knowledge (subject matters). The study urges further research on CTPACK framework which potentially enhance integration of CT in K-12 education.

KEYWORDS
computational thinking, ill-structured problem-solving, digital fabrication, TPACK, CTPACK

1. INTRODUCTION

1.1. Computational Thinking in K-12 Education and Emerging CTPACK Framework

Currently, there is a growing need in educational contexts to develop students' ability to deal with non-routine and abstract tasks (Kirschner, 2002). One of the important skills to confront ill-structured problems in this digitalized society is *Computational Thinking* (CT). CT refers to a way of solving complex problems by applying the set of thinking skills, practices and approaches which are fundamental to computer science (Wing, 2006). CT leads to understanding how computer works as well as possibilities and limitations of technologies, which is vital for taking advantage of technology-infused social world (Denning & Tedre, 2019).

Wing (2006) encourages to apply CT in K-12 education describing CT as "a fundamental skill for everyone, not just for computer scientists" (p.33). Previous studies have identified needs for further research to enhance integration of CT in K-12 education. Those research needs include combining CT with other subject studies, and teachers' professional development to synthesize CT with existing contents and pedagogical strategies (Howland, Good, Robertson, & Manches, 2019; Mäkitalo, Tedre, Laru, & Valtonen, 2019).

Mäkitalo and colleagues (2019) propose CTPACK framework to support integration of CT into school curriculum. CTPACK framework combines CT in the framework of technological pedagogical content knowledge (TPACK). TPACK framework, introduced by Mishra and Koehler (2006), has been used in educational contexts to integrate technologies, pedagogies and subject matters in teaching and learning. CTPACK represents skill set for teachers to guide development of CT through subject study with appropriate technologies and pedagogy in K-12 educational contexts (Mäkitalo et al., 2019). Although CTPACK is still an emerging framework, it has potential to enhance integration of CT in educational contexts.

1.2. Aim of the Study

The aim of this study is to understand how elements of CT intersects with technological pedagogical content knowledge (TPACK). CTPACK framework supports the integration of CT in K-12 education by 1) recognizing CT as part of aspects which teachers need to consider in order to position CT as an objective of learning at K-12 schools and 2) providing practical framework to combine CT with teachers' existing practices of designing and implementing learning activities. Results contribute in advancing practices of integration of CT in K-12 education and establishing applicable CTPACK framework.

1.3. Digital Fabrication as a Context to Integrate CT in K-12 Education

We use ill-structured digital fabrication activities as contexts to integrate CT in K-12 school curriculum. Previous studies showed *digital fabrication*, a process of making artefacts with digital technologies, is a potential context to develop CT (Borges, de Menezes, & da Cruz Fagundes, 2017; Iwata, Pitkänen, Laru, & Mäkitalo, 2019). In K-12 education, digital fabrication can be used to learn different subjects, such as mathematics, physics, art, and history (e.g., Blikstein, 2013; Pitkänen & Iwata, 2019).

The theory underlies digital fabrication in educational contexts is *constructionism* (Blikstein, 2013). Constructionism emphasizes individuals learn effectively in interactions with the physical and social environment, such as making personally meaningful artefacts and publicly sharing objects (Papert & Harel, 1991). Pitkänen, Iwata, and Laru (2019) emphasize teachers' significant roles and the needs of pedagogical views in designing and implementing ill-structured digital fabrication activities. Although digital fabrication activities tend to be student-centered, effort to support students' learning based on pedagogical understanding is necessary. There are less studies which utilize TPACK framework in digital fabrication in formal education. However, Smith (2013) applied TPACK

framework to examine afterschool digital fabrication activities. She analyzed instructional strategies related to each element of TPACK as well as in pairs and a combination of three. Results showed encouraging technical resourcefulness as technological knowledge, utilizing constructionism approach as pedagogical knowledge, and developing multiple modes of literacy as content knowledge (Smith, 2013). Thus, this study shows the importance of developing all areas. Integrating CT into TPACK framework provide the tool for teachers to better understand the holistic perspective of CT.

2. RESEARCH METHODS

2.1. Research Context and Cases

The context of the study is a makerspace in Finland. The makerspace offers digital fabrication facilities, such as 3D printers, laser cutters, vinyl cutters and programmable microcontrollers. The makerspace arranges digital fabrication activities for school visitors. We focus on three cases of school visits where 7th-9th grade students engaged in digital fabrication activities at the makerspace in 2016. Student groups from three different schools visited the makerspace as part of multidisciplinary learning module, which emphasizes integrating multiple subject domains (Finnish National Agency for Education, 2016). Overview of the cases and differences were as follows:

Case I (School A): 12 students (9th grade) accompanied by a teacher worked on digital fabrication projects for five days. The projects were, for example, electronic controlled lock, jukebox game, and music car. Students had autonomy of what to make with only a few requirements, such as using a microcontroller.

Case II (School B): 20 students (7th-8th grade) and two teachers visited the makerspace for three days. Students developed project ideas, such as Finland 100 years calendar, Finland 100 years history wheel, and Finland flag day clock, based on the theme provided by teachers and requirement of using a microcontroller.

Case III (School C): 9 students (9th grade) with two teachers visited the makerspace for five days. Students had initial project ideas as visiting the makerspace was a part of the ongoing project: designing a playhouse for the school community.

Table 2. Summary of Technologies Used in the Activities.

Technologies	School A	School B	School C
Design tool	Inkscape, Tinkercad	Inkscape	Inkscape, SketchUp
Electronics	Arduino Uno, servos, buttons, piezoelectric buzzer	Arduino Uno, servos	
Programming	Arduino	Arduino	
Machines	Laser cutter, 3D printer	Laser cutter	Laser cutter, vinyl cutter, sewing machine

Students used different technologies during the activities (see Table 2). All the projects were implemented as collaborative projects, where students worked together on one project as a group. Activities were run by two facilitators who work at the makerspace. The facilitators' main role was to provide instructions of basic operations of facilities and digital tools and to help students when they had problems in the processes. Teachers' role at the makerspace was mainly observing activities and general time management.

2.2. Data Collection and Analysis

Data was collected through 1) observation, 2) semi-structured informal interviews with teachers, students and facilitators during or after the activities, and 3) two semi-structured focus group interviews with teachers (focus group interview I) and facilitators (focus group interview II). During the observation, we took notes and photos focusing on overall structure, contents and instructions of the activities. In the semi-structured informal interviews, we asked about their perspectives on the digital fabrication activity. The interviews were recorded in video and audio.

In data analysis we focused on how CT was seen and described in relation to each element of TPACK framework. The main data for this study was focus group interviews. Observation data was used to deepen understanding of the contexts and to refine the research design and questions. Data was analyzed through theory-driven approach. We coded the data based on definitions of CT (Barr, Harrison, & Conery, 2011), which have been used in K-12 contexts, as well as each element of TPACK framework (Mishra & Koehler, 2006). We performed matrix coding analysis to see how CT and each element of TPACK framework are interconnected. NVivo software was used to support data analysis process.

3. RESULTS

Table 3 shows CTPACK elements, which represent connections of CT and TPACK, identified in focus group interviews. CT was mainly discussed in relation to each TPACK element: technological, pedagogical and content knowledge separately. Also, CT was discussed together with technological pedagogical knowledge as a pair.

3.1. CT and Technological Knowledge: Advanced Technologies and Mechanics for Developing CT

Teachers and facilitators mentioned that students' CT was developed through the following processes: 1) programming of microcontrollers, 2) machining, including preparing files in a certain format and operating machines correctly, and 3) making artefacts which have mechanical function. These results are in line with our previous study (Iwata et al., 2019), yet provide new insights of how CT intersects with technologies together with other elements of TPACK (see later sections).

3.2. CT and Pedagogical Knowledge: Solving Complex Problems through Learning by Doing

Students used CT in the processes of learning by doing. Constructionism, which underlies digital fabrication, encouraged solving complex problems while they were working on the projects. In the iterative processes of complex problem-solving, students analyzed the possible solutions to improve the next design cycle.

School B had a unique division of roles among groups. Two groups worked collaboratively on one project by dividing the tasks: one group was responsible for outer design and another group for inside mechanics of the product. Although communication load between design and mechanics groups increased, in this way, students were able to focus on specific aspects of complex ill-structured digital fabrication activity. A student from School B explained as follows:

There was two groups working for the same product, but both had own tasks. We had to decide all those dimensions together, between two groups, that the product will be right size. It wasn't hard, we get along well, and we managed to do right everything. (Student, informal interview)

Table 3. CTPACK Elements in Focus Group Interviews.

	Focus group interview I n(total)[a]=8,387 n(CT)[b] = 944		Focus group interview II n(total)=6,328 n(CT) = 826	
CTPACK elements	CC[c]	n[d]	CC	n
CT Technological knowledge	35.8%	187	64.6%	268
CT Pedagogical knowledge	34.7%	181	11.3%	47
CT Content knowledge	0.0%		24.1%	100
CT Technological Pedagogical knowledge	29.5%	154	0.0%	
CT Technological Content knowledge	0.0%		0.0%	
CT Pedagogical Content knowledge	0.0%		0.0%	
CT Technological Pedagogical Content knowledge	0.0%		0.0%	
Total	100%	522	100%	415

a Total number of words in the focus group interview; b Number of words regarding CT; c Coding coverage: percentage of the number of words coded at the node; d Number of words at the node.

3.3. CT and Content Knowledge: Applying Multiple Subjects in Complex Problem-Solving

The activities included multiple school subjects, such as math, physics, art, craft and English, as well as programming and coding (cf. Pitkänen & Iwata, 2019). One facilitator highlighted applying CT and school subjects in the context of digital fabrication as follows:

Computational thinking it's best applied to a little bit larger design problems, really have to divide your work into pieces that you have to solve piece by piece. But maybe at schools the curriculum is just their subjects, they are not linked together. But in [the makerspace] when we make a device, we have several subjects we have to combine into one device. (Facilitator, focus group interview)

In complex problem-solving in digital fabrication, which requires using knowledge of multiple school subjects, CT can be effectively developed.

3.4. CT and Technological Pedagogical Knowledge: supporting development of CT with technologies and pedagogy

In the case of School A, facilitators arranged a short lecture where they explained how logic ports on microcontroller work. Having lecture to theoretically understand logic port function effectively supported students in learning CT. Using microcontroller enhanced students' learning by enabling to apply theoretical knowledge of logical port functions into practices.

Teachers from School C explained that they used a digital mind map tool to support the students in ideation process. It helped logically organize and analyze their ideas. Teacher from School C reflected as follows:

In a start point…. the students made that mind map very quickly, just some words, and after two days, they have to make second mind map, and they just know that, "now I have so much more ideas to go through in this week". Also, they recognized the whole process and the whole project, what to do, and what we need, and how to solve the different kind of problems and so on. (Teacher, focus group interview)

In different phases of the project, the mind map tool helped students to generate ideas, to understand whole processes of the project, and to organize small steps required to complete the project.

4. DISCUSSION

CT and pedagogical knowledge were highlighted by two means: 1) Pedagogical approach of learning by doing enhanced developing CT. Students faced complex problems in the processes of making artefacts. Smith (2013) describes constructionism and learning by doing as the core of pedagogical knowledge in digital fabrication. 2) Dividing responsibilities may support dealing with complex problems. Digital fabrication project in few day activity tends to give heavy workload for K-12 students (Pitkänen & Iwata, 2019). Distribution of responsibility allows focusing on a small part of the whole project. Activities can be designed considering complexness which contributes to the development of CT, as well as students' limited capacity. Dividing responsibility may be effective in providing balanced workload.

We found two factors in which CT intersects with technological pedagogical knowledge: 1) Advanced technologies enhance feedback process of learning by doing, which contributes to developing CT. By using technological tools and machines, students can get feedback of their trial quickly, which resulted in encouraging trial and errors (Pitkänen & Iwata, 2019). As Papert (1980) described computer as an "object-to-thing-with" (p. 23), students develop CT through interacting with technological tools. 2) Technologies helped the process of supporting students' thinking process during ill-structured activities.

Results show that neither teachers nor facilitators discussed intensively how subject matters directly relate to CT. One of the potential reasons is that three cases of makerspace visit were implemented as part of schools' multidisciplinary learning module. Thus, teachers did not intend to let students learn specific aspects of subject matter. Based on the results, it is a challenge to widen teachers' understanding about CT,

because of the lack of long-term design and discussion about the skills and competencies of CT – from the holistic perspective (Denning & Tedre, 2019) and it's relation to technology, pedagogy and content. Another possible reason is that cognitive demand of the activity was high, thus participants had only limited room to focus on subject matters during the activities. Pitkänen et al. (2019) argue potential challenges which students face during ill-structured digital fabrication activities due to high cognitive demand. Cognitive demand in digital fabrication activities can be increased by digital tools and machines with which students and teachers are not familiar. In addition, ill-structured activity design with minimal instructions might contribute to increasing cognitive demand.

Limitations of the study are in data collection and analysis processes. In the interviews, we did not ask questions focusing on learning of subject matters. It might affect to results of vague connection between CT and content knowledge. In data analysis, we used operational definition of CT introduced by Barr et al. (2011). However, aspects of CT in the definition are not directly related to processes of digital fabrication. Using a definition of CT which takes the research context (digital fabrication) into account, such as Borges et al. (2017), may increase reliability of results.

5. CONCLUSION

This paper presented preliminary study examining the current practices of digital fabrication activities for K-12 students to understand how CT and elements of TPACK are interconnected. We found connections of CT and part of TPACK elements. Results provide the basis for understanding the role of CT in ill-structured digital fabrication activities. Further, CTPACK framework provide practical solutions to connect CT in subject matter with appropriate technologies and pedagogy in order to widen teachers' understanding about CT. In future study, CTPACK framework can be used as a tool to develop digital fabrication activities to integrate CT in school curriculum. To examine applicability of CTPACK, students with broader grade levels can be chosen as participants, and data can be analyzed considering different level of subject studies, students' age, sex, and background.

6. REFERENCES

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Blikstein, P. (2013). Digital Fabrication and 'Making' in Education: The Democratization of Invention. *FabLabs: Of Machines, Makers and Inventors*, *4*, 1-21.

Borges, K. S., de Menezes, C. S., & da Cruz Fagundes, L. (2017). The Use of Computational Thinking in Digital Fabrication Projects a Case Study from the Cognitive Perspective. *Proceedings of the 2017 IEEE Frontiers in Education Conference (FIE)*. IEEE, 1-6.

Denning, P. J., & Tedre, M. (2019). *Computational thinking*. MIT Press.

Finnish National Board of Education. (2016). *National core curriculum for basic education 2014*. Helsinki: Finnish National Board of Education.

Howland, K., Good J., Robertson, J., & Manches, A. (2019). Special Issue on Computational Thinking and Coding in Childhood. *International Journal of Child-Computer Interaction*, *19*, 93-95.

Iwata, M., Pitkänen K., Laru, J., & Mäkitalo, K. (2019). Developing Computational Thinking Practices through Digital Fabrication Activities. *Proceedings of International Conference on Computational Thinking Education 2019*. Hong Kong: The Education University of Hong Kong, 223-228.

Kirschner, P. A. (2002). Cognitive Load Theory: Implications of Cognitive Load Theory on the Design of Learning. *Learning and Instruction*, *12*(1), 1–10.

Mäkitalo, K, Tedre, M., Laru, J., & Valtonen, T. (2019). Computational Thinking in Finnish Pre-Service Teacher Education. *Proceedings of International Conference on Computational Thinking Education 2019*. Hong Kong: The Education University of Hong Kong, 105-107.

Mishra, P., & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record*, *108*(6), 1017-1054.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..

Papert, S., & Harel, I. (1991). *Situating Constructionism.* Retrieved December 12, 2019, from http://www.papert.org/articles/SituatingConstructionism.html

Pitkänen, K., & Iwata, M. (2019). What are the premises and the essential elements of activity design for applying digital fabrication in formal education? *Unpublished Master's Thesis.* University of Oulu, Oulu, Finland.

Pitkänen, K., Iwata, M., & Laru, J. (2019). Supporting Fab Lab Facilitators to Develop Pedagogical Practices to Improve Learning in Digital Fabrication Activities. *Proceedings of Fablearn Europe 2019 Conference.* ACM, 6.

Smith, S. (2013). Through the Teacher's eyes: Unpacking the TPACK of Digital Fabrication Integration in Middle School Language Arts. *Journal of Research on Technology in Education*, *46*(2), 207-227.

Wing, J. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33-35.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

Analysis of Research Status and Trends of Computational Thinking in China Based on Knowledge Graph

Hanrui GAO[1], Yi ZHANG[2*], Wei MO[3], Xing LI[4]
[1, 2] School of Educational Technology, Central China Normal University, China
[3] School of Education Science, Hunan Institute of Science and Technology, China
[4] School of Education，Jiang Han University, China
1053111801@qq.com, zhangyi@mail.ccnu.edu.cn, mowei0201@gmail.com, lxosu@qq.com

ABSTRACT
The study analyzed the literature of computational thinking （CT） in CNKI by using the Knowledge Graph, and analyzed the main characteristics and the level of the research field of CT in China by using the keyword word frequency co-occurrence analysis method. The results show that the research on CT in China gradually returns to the rational state. Drag-and-drop programming for children provides an opportunity for the development of computational thinking in primary and secondary schools. The training of computational thinking mainly relies on programming, information technology, mathematics and other science and engineering courses. This paper sorts out the development of computational thinking, and puts forward enlightenment for future research development: Introduction of standardized quantitative or qualitative assessment method of CT; Exploration of the development of learners' CT from multiple perspectives; Enhancement of the awareness of in-service teachers' CT, and investigation of the teaching approach and pedagogy of the in service teachers' CT.

KEYWORDS
knowledge graph, computational thinking, evolution of computational thinking research

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

基于知识图谱的我国计算思维研究现状与研究趋势探析

高晗蕊[1]，张屹[2*]，莫尉[3]，李幸[4]
[1,2]华中师范大学教育信息技术学院，中国
[3]湖南理工学院教育科学学院，中国
[4]江汉大学教育学院，中国
1053111801@qq.com，zhangyi@mail.ccnu.edu.cn，mowei0201@gmail.com，lxosu@qq.com

摘要
本研究利用知识图谱对中国知网中计算思维文献分析，通过关键词词频共现分析法探析我国计算思维研究领域的突出特性和层次深度。研究发现：我国计算思维相关研究逐步回归理性状态；适合儿童的拖拽式编程为中小学计算思维的培养带了契机；计算思维培养主要依托编程、信息技术、数学等理工科课程，研究主题多为课程模式的构建。本文梳理了计算思维发展脉络，为未来的研究发展提出了启示：引入标准化的定量或定性计算思维评价方法；多角度探究学习者计算思维的培养；增强一线教师对计算思维的认知，探究一线教师计算思维培养模式与方法。

关键词
知识图谱，计算思维，计算思维研究演变

1. 引言
2014年3月，教育部发布的《关于全面深化课程改革落实立德树人根本任务的意见》（以下简称《意见》）以及 2017 版《普通高中信息技术课程标准》（以下简称《课标》）等文件为计算思维相关研究在我国的快速发展提供了契机。为了解我国计算思维研究现状和趋势，研究者运用 CiteSpace 文献计量工具对中国知网中计算思维相关文献进行分析，探析计算思维领域研究发展脉络，梳理现有研究及现有研究的不足为计算思维的进一步研究提供支点。

2. 问题提出
周以真教授认为计算思维并非计算机专业人员的特有，计算思维等同于听、说、读、写，是每个人都需要掌握的日常生活技能之一，是运用计算机概念抽象问题模型，形成解决方案，再由信息处理代理自动化有效执行的过程（Wing, 2006；Wing, 2008）。目前计算思维的定义可分为两个方面，第一：计算思维是一种必备的思维能力，指的是形成问题及其结果方案时所设计的思维过程，使得解决方案能够快速有效的执行（Sysło & Kwiatkowska, 2013）；计算思维包括计算科学中的概念和思维过程，这些思维和过程将有助于学习者面对不同领域的问题时形成相应的解决方案（Mannila, Dagiene, Demo et al, 2014）；第二：计算思维是问题解决、系统设计的过程或方法，关注的重点是利用计算机学科的基本概念来理解人类的行为（Korkmaz, Çakir & Özden, 2017）；还有学者认为计算思维是用计算机实现问题解决的方法，能够使用抽象、迭递归等计算机学科概念来处理和分析数据，可以自动化的将相关概念和技能跨领域的应用（Barr & Stephenson, 2011）。计算思维最后指向学习者在信息社会中解决问题的一种普适的基本能力，强调运用计算机科学的基础概念进行问题求解、系统设计的思维过程和行动，在问题解决的过程中，学生不仅要能够熟练的运用可供选择的工具，还要掌握计算思维，运用计算思维，成为问题解决方案的思考着和设计者、成为新型问题解决工具的开发者与迁移者。基于此，本研究探究以下问题，以期为接下来的研究提供支点：（1）我国计算思维的研究现状如何？（2）如何随时间变化？（3）研究热点、研究领域以及层次深度如何？（4）结合国外研究前沿为我们带来什么样的启示？

3. 研究设计
3.1. 研究方法
CiteSapce 可以用于寻找某一学科领域的研究进展和当前的研究前沿，能够将一个知识领域的演变历程呈现在一张图上。CiteSpace 中关键词或特征词图谱配合突现词功能使用可以帮助人们研究热点及热点的演变（陈悦、陈超美和刘则渊等，2015）。本研究利用了聚类视图和时间线视图结合的功能，构建计算思维关键词图谱和计算思维关键词时间线图谱，分析计算思维研究热点及随时间研究热点的转变，并对关键词结果进行聚类，分析计算思维研究领域及对应的研究深度。

3.2. 数据来源
计算思维又称为“运算思维”，因此，以“计算思维”或“运算思维”为主题词，以中国知网中的期刊文献为研究对象，对时间不进行限制的情况下进行精确检索，得到记录为 3231 条，经初步分析发现 2010 年及以前文献较少，近 5 年文献数量较多，主题相关度高，依据本文研究目的为考察计算思维领域目前研究现状与趋势，将时间节点设置成 2014 年 1 月 1 号到 2019 年 6 月 1 号，对近 5-6 年内关于“计算思维”或“运算思维”的文献进行检索，共检索出 2572 条结果，经过人工筛选，剔除与主题无关、关键词混淆、会议通知等无效记录，剩余 2354 条记录。

4. 研究结果与分析
4.1. 计算思维研究热点分析
4.1.1. 计算思维研究文献时间分布图
计算思维的研究与我国重要文件和报告的提出呈显明显的相关性。如图 1 所示，11-14 年文献数量增加幅度

较大，14-17 年文献数量无显著性变化，17 年之后文献数量又有了小幅度增加，从 19 年上半年的形势来看应该跟 18 年大致持平。其中文献数量急剧增加的年份为 12 年、13 年和 17 年，结合文献发表周期等现实情况，２０１０年发布的《九校联盟（Ｃ９）计算机基础教学发展战略联合声明》强调了高等教育中计算机基础教育培养学习者计算思维的重要性，并提出了相应的课程体系建设和课程目标（董荣胜，2010），《课标》将计算思维培养列入了课程培养目标范围之内等文件内容的发布推动了相关研究的发展。

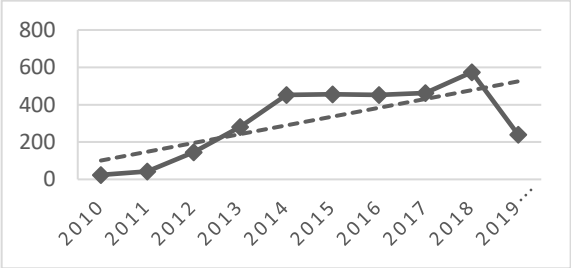


图 1 计算思维研究文献时间分布图

4.1.2. 计算思维研究热点分析

关键词是了解文献的主题、内容等关键内容的重要线索，关键词的中心度和出现的频次，代表了一段时间内该领域研究者的关注热点。本研究运用 CiteSpace 对知网中 2014 年到 2019 年 6 月 1 号的 2354 条数据进行可视化分析。Time Slicing 设定为“2014-2019”；Years Per Slice 设定为 1 年；Node Types 设置为关键词；Selection Criteria 设定为 Top N=50，运行 CiteSpace，高频关键词节选如表 1 所示。中心度最高的为“计算思维”，为 0.51；其次分别是信息技术、计算思维能力、核心素养、信息技术课程，分别为 0.09、0.08、0.08、0.08。由此可见，在该网络中，计算思维能力、信息技术课程、核心素养培养、编程语言是计算思维发展过程的主要关注点。

表 1 计算思维研究领域高中心度词汇节选

序号	词频	中心度	年份	关键词
1	1929	0.51	2014	计算思维
2	110	0.09	2014	信息技术
3	83	0.08	2014	计算思维能力
4	69	0.08	2017	核心素养
5	55	0.08	2014	信息技术课程

CiteSpace 关键词的聚类功能可以显示具体研究领域的热点与发展趋势。如图 2 所示，计算思维关键词聚类图谱网络节点共有 138 个， 598 条网络连线，网络密度为 0.0633。相关领域从研究主题上可以划分为计算思维、信息技术、程序设计、计算机基础教学、教学模式以及教学改革等；从研究层次上主要集中在高等教育研究、基础教育研究领域，其中基础教育研究中大多以信息技术课程为依托，计算思维在高等教育中的研究主要以大学计算机基础课程教学为依托。近五年关于计算思维的研究在课程方面有关于课程体系改革，课程模式探索以及教学方法等方面的研究，依托的课堂多为计算机或编程等相关课程，注重学习者 21 世纪核心素养、高阶思维能力等方面的培养。

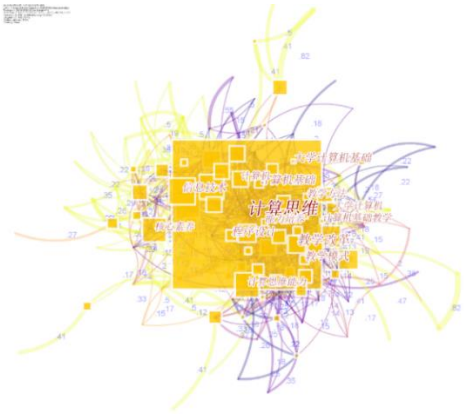


图 2 计算思维关键词频次聚类图普

4.2. 计算思维研究热点、领域等随时间发展的转变

时间线视图从时间维度对关键词进行聚类，分析聚类之间的关系和某个聚类中文献的历史跨度。设置时间切片为 1 年，构建关键词实时间线视图聚类，如图 3 所示。

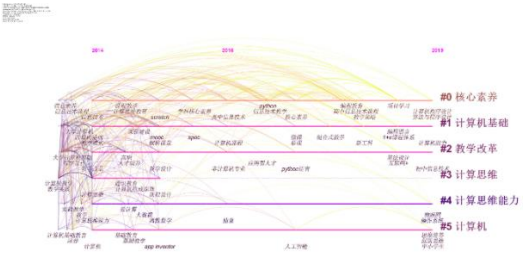


图 3 计算思维关键词时间线聚类视图

由图 3 可以清晰的看出计算思维各个研究热点聚类的发展脉络，从中我们可以看到，计算思维相关研究大致聚为 6 类，分别是核心素养、计算机基础、教学改革、计算思维、计算思维能力、计算机。

4.2.1. 类#0 核心素养

《意见》中第一次提出加快“核心素养体系建设”的指导意见，将核心素养体系放在了深化课程改革、落实立德树人目标的基础地位。有上图聚类可知，在计算思维研究领域研究者们对学习者“信息素养”持续关注，并在 15 年左右“学科核心素养”出现。期间编程教育、项目学习等学习方式也一直强调核心素养的培养，计算思维的培养与核心素养的培养持续关联。

4.2.2. 聚类#1 和聚类#5 计算机基础和计算机

计算思维自提出之日，研究者们便希望能够运用计算机学科的概念和原理去理解问题和解决问题。计算机基础教学的核心任务是计算思维能力的培养（何钦铭、陆汉权和冯博琴，2010）。在此聚类中可看出，从基础教育的信息技术到高等教育的大学计算机均为研究者们的关注点。其中随时间发展，中小学阶段以及高中阶段的信息技术课程、scratch 编程、人工智能等逐渐被研究者关注，计算思维的培养也逐步深入基础教育中去，逐渐关注中小学学生创造力等高阶思维的培养。

4.2.3. 聚类#2 教学改革

由聚类图可知，教学改革这一聚类从学科角度看主要集中在在信息技术课程、编程教育、跨学科教育等方面；从改革内容和形式来看主要集中在教学模式的探索、课程体系的架构、课程设计与建设等方面。逐步引入新工科、互联网等学科领域，目的指向应用型人才以及学习者计算思维等高阶能力的培养。

4.2.4. 聚类#3 和聚类#4 计算思维和计算思维能力

研究者们对计算思维定义及其培养的关注一直持续不下。“抽象”这一单独的概念维度在 2016 年出现被研究者们关注，计算思维的培养开始出现根据不同年龄阶段学习者的特征开展计算思维培养，不同年龄阶段学习者所接受的计算思维培养内容和层次不同。

5. 研究结论与启示

5.1. 研究结论

本研究得到的主要结论如下：

（1）计算思维研究领域受权威机构的指导意见影响。研究热点、研究情况受国家层面或权威机构发布的相关政策与文件影响较大，政府层面或权威机构发布的多个指导性文献有效推动了我国对计算思维的培养向各教育阶段的过渡，教育领域的研究者们针对计算思维从基础教育到高等教育均展开了大量的相关研究。计算思维目前研究热度处于平稳状态，相关文献数量不再急剧增长，相关研究从萌芽时期经历奠基时期和混沌时期逐步回归理性状态。

（2）计算思维的培养主要依托理工科课程。高等教育中主要依托计算机基础课程及程序设计课程，基础教育中主要为信息技术课程、数学课程以及编程兴趣班。高等教育中以计算机课程为依托，计算思维最主要的培养方式是通过代码的编写以及代码逻辑的学习来实现计算思维的培养。随着教育理念和教育技术的发展，人工智能、创客教育、拖拽式编程为中小学计算思维的培养带了契机（孙立会和周丹华，2019）。

（3）计算思维的研究主题多围绕培养模式的构建。计算思维研究主题近五年来大多围绕各阶段的信息技术课程进行，研究主题涉及计算思思维概念的介绍与界定、计算思维发展的教学模式的构建和教学活动的设计、相关课程案例的探讨等，也多处提及课程改革，关于哪些活动环节、资源形式影响学习者计算思维能力发展及如何评价学习者计算思维发展的相关研究较少。

5.2. 研究启示

计算思维的发展与习得将会帮助学习者运用计算机科学的理念和知识去理解世界，为其工作和学习带来便利，是未来人才培养的重要目标之一，结合研究结论及国际前沿演技，本研究提出以下反思建议：

（1）引入量表、测试题、理论模型等标准化的定量或定性计算思维评价方法。正确有效的评价反馈才能促进更好的教学，那么计算思维如何评价？评价什么？怎么评价？仍是相关学者需要思考的问题。有效的计算思维评价可以科学合理的评估学习者的起点水平、认知结构、学习态度以及学习进步等，以便实施个性化教学干预（Román-González Marcos，Pérez-González Juan-Carlos et al, 2018）。

（2）多角度探究学习者计算思维的培养。我国研究者对于计算思维培养的研究大多依托于中小学信息技术课程以及大学的计算机课程，主要的方式是通过程序设计来实现计算思维的培养，计算思维是一种每个人都要具备的思维模式，是一种运用计算机科学概念理解世界、解决问题的方式或方法，计算思维的培养不仅可以依托于程序设计。计算思维的培养更要从多个角度探究学习活动方式、教学资源形式对学习者计算思维发展的影响（Zhao & Shute, 2019）。

（3）增强教师对计算思维的认知，开展主题为计算思维理念与培养方法的教师培训。教师是培养学习者计算思维能力培养的直接执行人，教师素质能力的高低是影响学习者的关键因素（Liying Xia & Baichang Zhong, 2019）。构建面向学习者计算思维培养的教师发展培训模式，开发指导教师设计并实施计算思维培养课程的资源体系等是落实计算思维培养教学目标的前提与根本。

6. 基金项目

国家自然科学基金 2018 面上项目 促进小学生计算思维培养的跨学科 STEM+C 教学理论与实证研究 71874066；

7. 参考文献

陈悦、陈超美、刘则渊、胡志刚和王贤文（2015）。CiteSpace 知识图谱的方法论功能。**科学学研究**，**33（2）**，242-253。

董荣胜（2010）。《九校联盟（C9）计算机基础教学发展战略联合声明》呼唤教育的转型。**中国大学教学**，**10**，14-15。

何钦铭、陆汉权和冯博琴（2010）。计算机基础教学的核心任务是计算思维能力的培养——《九校联盟（C9）计算机基础教学发展战略联合声明》解读。**中国大学教学**，**9**，7-11。

孙立会和周丹华（2019）。国际儿童编程教育研究现状与行动路径。**开放教育研究**，**2**，23-34。

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *Inroads, 2*(1), 48-54.

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.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. ACM, 1-29.

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Can Computational Talent be Detected? Predictive Validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction, 18*, 47-58.

Sysło, M. M., & Kwiatkowska, A. B. (2013). Informatics for all High School Students. *Proceedings of International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*. Berlin, Heidelberg: Springer, 43-56.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1811), 3717-3725.

Xia, L., & Zhong, B. (2018). A Systematic Review on Teaching and Learning Robotics Content Knowledge in K-12. *Computers & Education, 127*, 267-282.

Zhao, W., & Shute, V. J. (2019). Can Playing a Video Game Foster Computational Thinking Skills? *Computers & Education, 141*, 103633.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

The Impact of Using Mobile Block-based Programming to Control Robots on the Performance of the Fifth Grader Students Learning Computational Thinking in Singapore

Tien-hsiu JEN[1], Ting-chia HSU[2*]
[1,2]National Taiwan Normal University, Taiwan
tt40621t@gmail.com, ckhsu@ntnu.edu.tw

ABSTRACT
This study attempted to cultivate the students to apply computational thinking process to solving the problems when the students play the interactive game with the educational robots. The instructional experiment participants were the fifth grader students in Singapore. The educational robots were controlled to interact in Chinese with the block-based programming. The results found that the students made significant progress both in the competence of computational thinking and the proficiency of conditional sentences in Chinses through the game-based learning tasks with robots.

KEYWORDS
computational thinking (CT), game-based learning (GBL), robot

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020.* Hong Kong: The Education University of Hong Kong.

使用手機積木程式工具操控機器人對新加坡五年級學生運算思維表現之影響

任天秀 [1]，許庭嘉 [2*]
[1,2] 國立臺灣師範大學科技應用與人力資源發展學系，台灣
tt40621t@gmail.com, ckhsu@ntnu.edu.tw

摘要
本研究藉由透過手機或平板和教育機器人的程式編輯器連線，透過編寫積木程式來操控機器人的互動內容，培養學生透過運算思維歷程來解決教育機器人互動遊戲中所遇到的問題。實驗對象為新加坡五年級的小學生，利用積木程式控制機器人使用華語進行互動，研究結果發現透過機器人的遊戲式學習任務，可以提升學生在學習運算思維上的成效，同時提升學生的華語文條件複句能力。

關鍵字
運算思維；機器人；遊戲式學習

1. 前言
在科技快速變遷的資訊化社會中，運算思維逐漸成為每個人必備的基本技能（Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014），運算思維與其相關的概念，例如：編寫程式碼、電腦程式設計、運算法思維等等，在教育領域受到越來越多的關注（Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016），培養學生運算思維的能力，成為教育領域的熱門課題。

近年來，世界各國為了因應此趨勢，並培養出在數位時代具備重要能力的人才，相繼提出新的資訊教育政策，將科技融入教育，而台灣於 2014 年頒佈的《十二年國民基本教育課程綱要總綱》也正式於 2019 年開始實行，其中資訊科技領域之課程即是以培養學生運算思維之素養為主軸，重視跨領域統整、溝通與團隊合作之能力，強調學習內涵須注重與生活的連結，而不宜再局限於單純的學科知識及技能（王佳琪, 2017）。

儘管教育隨著時代不斷地變化，但遊戲一直是教育不可或缺的一部分，教育遊戲的概念已在當今的教育界中得到應用（Donmus, 2010），根據眾多研究支持遊戲對學習的積極影響，越來越多的研究人員致力於開發教育遊戲（Qian & Clark, 2016），學者 Reinders 和 Wattana（2015）表示遊戲可以激勵人們，降低學習中的情感障礙，並鼓勵外語或第二語言（L2）的互動。

因此，本研究將探究使用手機應用程式與教育機器人相互配合，讓華語作為第二語言學習的國小生，利用手機應用程式中模組化的程式設計工具操作教育機器人，學習華語邏輯與文法規則，並從遊戲中培養運算思維與分析的能力，使得學生在遊戲過程中，能夠運用運算思維解決所遇到的問題，並且釐清華語文法上基本觀念。

2. 文獻探討
2.1. 運算思維
運算思維（Computational Thinking, CT）是使用電腦和訊息科學必不可少的概念，經常被用來解決問題、設計和評估複雜的系統，並理解人類的推理和行為（Buitrago Flórez et al., 2017）。Korkmaz、Çakir 和 Özden（2017）認為可以將「運算思維」簡單地定義為具有能夠使用電腦解決生產中的生活問題所必需的知識、技能和態度，這種思維方式對電腦科學以及幾乎所有其他領域都具有重要意義（Buitrago Flórez et al., 2017）。（Wing, 2006）表示一旦學生掌握了運算思維的概念，便可以將其應用於電腦科學以外的領域。

在過去的十年中，運算思維和相關的概念例如：編寫程式碼、電腦程式設計、運算法思維，在教育領域受到越來越多的關注（Bocconi et al., 2016），隨著世界各國政府在學校課程中引入這些技能，通過電腦程式設計來發展運算思維技能是教育的一個主要焦點（Moreno-León, Robles, & Román-González, 2016），運算思維被認為是一種普遍的能力，應將其添加到每位孩子的分析能力中，作為他們學校學習的重要組成部分（Voogt, Fisser, Good, Mishra, & Yadav, 2015）。

然而，運算思維需要透過訓練和指導，不是自然而然產生的（Sanford & Naidu, 2016），儘管程式設計對年輕學生非常有吸引力，且具有很好的實踐或經驗，但是在程式設計方法或運算思維過程中發展學生的邏輯思維能力和解決問題的能力可能更有趣（García-Peñalvo, 2018）。

2.2. 機器人
機器人一直受到越來越多的關注，並且機器人在教育中的許多方面被認為是有前途的教學手段（Cheng, Sun, & Chen, 2017）。機器人程式可以吸引人的學習環境，以獲取核心的運算思維能力（Witherspoon, Higashi, Schunn, Baehr, & Shoop, 2017）。在世界範圍內，經濟和技術要求等因素都在積極促進程式設計教育（Noh & Lee, 2019）。教育機器人程式已在大多數發達國家中流行，並且在發展中國家也越來越流行（Miller & Nourbakhsh, 2016）。

如今，教育機器人已經開始走進校園和家庭，改變了傳統的教學方式（Jin, Xie, Ma, & Ye, 2019）。機器人技術的發展具有與教育系統整合的巨大潛力，機器人技術在中小學生中變得越來越普遍（Besari et al., 2016）。機器人技術用於在各個教育階段的學生中教授問題解決、程式設計、設計、物理、數學甚至音樂和藝術（Miller & Nourbakhsh, 2016）。

2.3. 電腦程式自我效能
自我效能與人們本身對完成任務或目標能力的信念有關（Bandura, 2006）。目標設定理論表明，困難的目標

可以提高許多任務的績效，但是當目標難以實現時，目標可能會產生挫折感和動力不足，結果也會降低績效（Baron, Mueller, & Wolfe, 2016）。

隨著對電腦系統的日益依賴以及新技術的引入日漸迅速，用戶對技術的接受度仍然是一個重要的問題（Mun & Hwang, 2003）。現代技術的發展及其對當今日常生活的擴展已是不爭的事實，電腦的廣泛使用使得有必要對這些技術進行培訓，例如：電腦自我效能、自我概念、態度、動力和需求（Paraskeva, Bouta, & Papagianni, 2008）。Karsten 與 Roth（1998）的研究結果顯示，電腦自我效能感的測量提供了有用的證據，表明通過培訓的過程，學生能夠更有效地提升使用電腦的能力。

2.4. 遊戲式學習

近年來，對基於數位遊戲式學習（Digital Game-Based Learning, DGBL）有效性的系統評估越來越引起人們的興趣（All, Castellar, & Van Looy, 2015），有鑑於眾多研究支持遊戲對學習的積極影響，越來越多的研究人員致力於開發教育遊戲，以促進學生在學校 21 世紀技能的發展（Qian & Clark, 2016）。電腦遊戲已向許多方向發展，許多研究和系統涉及遊戲結構中的“樂趣”和“愉快”等不同的元素，以提高學習者在教育學習領域的動力（Al-Azawi, Al-Faliti, & Al-Blushi, 2016）。技術的進步導致教學方法的不斷創新，例如：在課堂教學中使用平板電腦（TPC）已被證明可以有效地吸引和激發學生的興趣，並提高他們參與學習活動的意願（Hung, Sun, & Yu, 2015）。遊戲可以激勵人們，降低學習中的情感障礙，並鼓勵外語或第二語言（L2）的互動（Reinders & Wattana, 2015）。

2.5. 合作學習

在過去的幾年中，有關課堂合作學習技術的研究一直在增加，在這種學習中，學生以小組形式工作，並根據小組的表現獲得獎勵或認可（Slavin, 1980）。在精心組織的小組中合作工作的學生可以最大限度地利用自己和彼此的學習（Smith, 1996）。通過鼓勵學習者共同努力解決問題，了解他人的觀點並合作尋找創造性和關鍵性的解決方案，這些經歷可以幫助認知和協作技能的發展（Lee et al., 2016）。通過合作組織努力，有大量證據表明學生將取得更高的成就，能夠學習更多，使用更高層次的推理策略，建立更完整和複雜的概念結構以及更準確地保留學習的訊息，建立更多的支持性和積極關係，其中也包括人際關係，並以更健康的方式發展，心理健康、自尊、應對壓力和逆境的能力皆會有所提升（Smith, 1996）。

3. 研究方法

3.1. 實驗對象

本次實驗對象為 52 位將華語作為第二語言學習的新加坡某國小五年級學生，性別分布為男性 30 位（58%），女性 22 位（42%），主要是透過手機應用程式與教育機器人相互配合使用，使學習者能夠運用運算思維並釐清華語文法上基本觀念，同時提升學習者的學習成效。

3.2. 研究工具

本研究使用機器人華語文句子學習單與電腦程式自我效能量表進行學習成效測量：

3.2.1. 機器人華語文句子學習單

本研究使用的機器人華語文句子學習單測驗學習者的華語能力，學習單內容取自與課文內容程度相同之華語教材，總共分為四大題，第一部份以詞語組成為主，第二部份選擇出正確的拼音，第三部份找出最適合的詞語填入句子中，最後第四部份偏重圖片識別部份。

3.2.2. 電腦程式自我效能表現量表

此量表用以個人對於自己電腦能力的自我判斷。採用 Tsai, Wang 與 Hsu（2019）所編製之「電腦程式自我效能表現量表」上的得分來決定，得分越高，表示其所具有的電腦程式自我效能表現越高，反之則越低。此量表包含三個構面分別為「邏輯思考」、「控制」與「除錯」，「邏輯思考」構面 4 題，「控制」與「除錯」構面各 3 題，合計 10 題。作答形式採用李克特的五等選項，「1」表示強烈反對，「5」表示堅決同意；各分量表加總取平均值即為各分量表分數，並分別進行前後測驗的比較。

3.3. 研究程序

本次實驗的實驗對象為 52 位將華語作為第二語言學習的新加坡某國小五年級學生，施測地點為班級教室進行施測，採團體施測的方式。使用手機應用程式與教育機器人相互配合使用，主要的目的是希望讓華語非母語的國小生利用模組化程式設計工具學習華語邏輯與文法規則，並透過相互合作學習，共同努力解決在學習過程中所面臨的問題，使學生能夠運用並釐清華語文法上基本觀念。

如圖 1 表示，在實驗開始之前，使用機器人華語文句子學習單與電腦程式自我效能表現量表，先對學生施行前測，評估基本的華語及運算思維能力，接著進行小組施測介紹並教授課程內容，經教學課程後，開始分組完成實驗內容，每組使用手機應用程式（操作介面如圖 2 所示）與教育機器人達成目標並完成實驗。待所有組別皆完成實驗，開始進行全班的團體競賽的施測介紹，藉由團體競爭的競爭方式提高學生的學習成效。全班競賽施測結束後，要求學生填寫與前測相同難易度的機器人華語文句子學習單與電腦程式自我效能表作為後測，了解實驗結果與學習成效是否有進步。

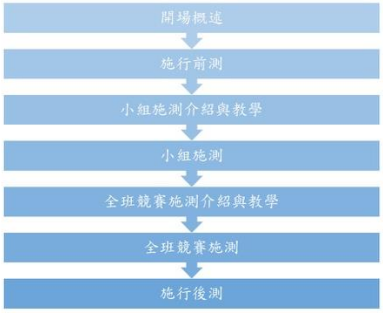


圖 1 實驗流程

圖 2 使用手機積木程式工具操控機器人之介面

4. 研究結果

本研究欲探究學生之成績，使用機器人華語文句子學習單進行前後測的成績測驗，並且將前後測的成績以相依樣本 t 檢定分析發現，由表 1 得知，學生之後測與前測之平均值有顯著差異，t（51）= -6.203，p<.05。後測成績（M=76.35，SD=13.64）顯著地大於前測成績（M=60.13，SD=23.26），由此可見學生透過手機應用程式與教育機器人相互配合學習華語，對於學習成效是有顯著增加的。

表 1 機器人華語文句子學習單之相依樣本 *t* 檢定

項目	平均數（標準差）		自由度	*t*
	前測	後測		
成績	60.13（23.26）	76.35（13.64）	51	-6.203

****p<0.001*

本研究欲探究學生在電腦程式自我效能之表現，將問卷分為「邏輯思考」、「控制」與「除錯」三個構面，將前後測以相依樣本 t 檢定進行分析發現，學生三構面後測與前測之平均值皆有顯著差異。

表 2 電腦程式自我效能表現量表之相依樣本 t 檢定

項目	平均數（標準差）		自由度	*t*
	前測	後測		
邏輯思考	3.07（1.09）	3.99（0.81）	51	-6.12
控制	2.96（1.41）	3.95（0.94）	51	-4.97
除錯	2.79（1.10）	3.92（0.83）	51	-6.94

****p<0.001*

由表 2 得知，在「邏輯思考」構面 t（51） = -6.12，p<.05，後測成績（M=3.99，SD=0.81）顯著地大於前測成績（M=3.07，SD=1.09）；在「控制」構面 t（51）= -4.97，p<.05，後測成績（M=3.95，SD=0.94）顯著地大於前測成績（M=2.96，SD=1.41）；在「除錯」構面 t（51） = -6.94，p<.05，後測成績（M=3.92，SD=0.83）顯著地大於前測成績（M=2.79，SD=1.10），由此可見學生透過手機應用程式與教育機器人相互配合學習華語，能夠顯著提升運算思維的能力。

5. 結論與未來展望

在過去的十年中，運算思維和相關的概念例如：編寫程式碼、電腦程式設計、運算法思維，在教育領域受到越來越多的關注（Bocconi et al., 2016），隨著世界各國政府在學校課程中引入這些技能，通過電腦程式設計來發展運算思維技能是教育的一個主要焦點（Moreno-León, Robles, & Román-González, 2016），而使用手機應用程式中模組化的程式設計工具，是為了培養學習者使用電腦邏輯來解決問題的運算思維（Buitrago Flórez et al., 2017），強化資訊科技能力。

在本項研究中，使用手機應用程式中模組化的程式設計工具操作教育機器人，學習運算思維與華語文法規則。研究結果顯示，學習者在經教學課程並分組完成實驗內容後，將前後測以相依樣本 t 檢定進行分析發現，使用機器人華語文句子學習單進行前後測的成績測驗之平均值有顯著差異，後測成績顯著地大於前測成績，表示學習者透過手機應用程式與教育機器人相互配合學習，確實能夠增加學習者學習華語的學習成效。並且，學習者在「邏輯思考」、「控制」與「除錯」三構面後測與前測之平均值皆有顯著差異，由此可見學習者透過手機應用程式與教育機器人相互配合學習華語，對於運算思維是確實有顯著提升的。

雖然本研究結果顯著，研究者認為華語的文法規則與語法多變性遠遠大於目前手機應用程式中模組化的程式設計工具所設計的內容，故希望未來可以朝向擴增系統的資料庫、增加不同課文內容以及語法規則等邁進，用以充實學習內容的深度與豐富度。

本研究之所以設計手機應用程式與教育機器人相互配合，是為了透過遊戲式學習激勵學習者，降低學習者在學習過程中的情感障礙（Reinders & Wattana, 2015），並鼓勵學習者增加在華語學習上的互動，使得學習者在遊戲過程中，能夠運用運算思維解決所遇到的問題，並且釐清華語文法上基本觀念。研究者認為，未來在研究上，建議也能透過教學增加學習者的學習動機以及提升整個課程的滿意度，使得未來研究方向可以更加完整。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 108-2511-H-003 -056 -MY3 的部分補助。

7. 參考文獻

王佳琪。（2017）。十二年國民基本教育課程綱要總綱之核心素養課程：評量的觀點。**臺灣教育評論月刊，6**（3）, 35-42.

Al-Azawi, R., Al-Faliti, F., & Al-Blushi, M. (2016). Educational Gamification VS. Game Based Learning: Comparative Study. *International Journal of Innovation, Management and Technology, 7*(4), 132-136.

All, A., Castellar, E. P. N., & Van Looy, J. (2015). Towards a Conceptual Framework for Assessing the Effectiveness of Digital Game-based Learning. *Computers & Education, 88*, 29-37.

Bandura, A. (2006). Guide for Constructing Self-efficacy Scales. *Self-efficacy beliefs of adolescents, 5*(1), 307-337.

Baron, R. A., Mueller, B. A., & Wolfe, M. T. (2016). Self-Efficacy and Entrepreneurs' Adoption of Unattainable Goals: The Restraining Effects of Self-control. *Journal of Business Venturing, 31*(1), 55-71.

Besari, A. R. A., Sukaridhoto, S., Wibowo, I. K., Berlian, M. H., Akbar, M. W., Yohanie, F. Y., & Bayu, K. A. (2016). Preliminary Design of Interactive Visual Mobile Programming on Educational Robot ADROIT V1. *Proceedings of the 2016 International Electronics Symposium (IES)*.IEEE, 499-503.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education-Implications for Policy and Practice.* Retrieved December 24, 2016 from https://econpapers.repec.org/paper/iptiptwpa/jrc104188.htm

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation's Way of Thinking: Teaching Computational Thinking through Programming. *Review of Educational Research*, *87*(4), 834-860.

Cheng, Y. W., Sun, P. C., & Chen, N. S. (2017). An Investigation of the Needs on Educational Robots. *Proceedings of 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT).* IEEE, 536-538.

Donmus, V. (2010). The Use of Social Networks in Educational Computer-game Based Foreign Language Learning. *Procedia-Social and Behavioral Sciences*, *9*, 1497-1503.

García-Peñalvo, F. J. (2018). Editorial Computational Thinking. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, *13*(1), 17-19.

Hung, C. Y., Sun, J. C. Y., & Yu, P. T. (2015). The Benefits of a Challenge: Student Motivation and Flow Experience in Tablet-PC-game-based Learning. *Interactive Learning Environments*, *23*(2), 172-190.

Jin, T., Xie, W., Ma, J., & Ye, K. (2019). Design Method and Example of a Simple Educational Robot. *Proceedings of the 2018 International Conference on Mathematics, Modeling, Simulation and Statistics Application (MMSSA 2018).*

Karsten, R., & Roth, R. M. (1998). Computer Self-efficacy: A Practical Indicator of Student Computer Competency in Introductory IS Courses. *Informing Science*, *1*(3), 61-68.

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.

Lee, H., Parsons, D., Kwon, G., Kim, J., Petrova, K., Jeong, E., & Ryu, H. (2016). Cooperation Begins: Encouraging Critical Thinking Skills Through Cooperative Reciprocity Using a Mobile Learning Game. *Computers & Education*, *97*, 97-115.

Miller, D. P., & Nourbakhsh, I. (2016). Robotics for Education. *Springer handbook of robotics.* Springer: 2115-2134.

Moreno-León, J., Robles, G., & Román-González, M. (2016). Comparing Computational Thinking Development Assessment Scores with Software Complexity Metrics. *Proceedings of the 2016 IEEE Global Engineering Education Conference (EDUCON)*, 1040-1045.

Mun, Y. Y., & Hwang, Y. (2003). Predicting the Use of Web-based Information Systems: Self-efficacy, Enjoyment, Learning Goal Orientation, and the Technology Acceptance Model. *International journal of human-computer studies*, *59*(4), 431-449.

Noh, J., & Lee, J. (2019). Effects of Robotics Programming on the Computational Thinking and Creativity of Elementary School Students. *Educational Technology Research and Development*, 1-22.

Paraskeva, F., Bouta, H., & Papagianni, A. (2008). Individual Characteristics and Computer Self-efficacy in Secondary Education Teachers to Integrate Technology in Educational Practice. *Computers & Education*, *50*(3), 1084-1091.

Qian, M., & Clark, K. R. (2016). Game-based Learning and 21st Century Skills: A Review of Recent Research. *Computers in Human Behavior*, *63*, 50-58.

Reinders, H., & Wattana, S. (2015). Affect and Willingness to Communicate in Digital Game-based Learning. *ReCALL*, *27*(1), 38-57.

Sanford, J. F., & Naidu, J. T. (2016). Computational Thinking Concepts for Grade School. *Contemporary Issues in Education Research*, *9*(1), 23-32.

Slavin, R. E. (1980). Cooperative Learning. *Review of Educational Research*, *50*(2), 315-342.

Smith, K. A. (1996). Cooperative learning: Making "Groupwork" Work. *New directions for teaching and learning*, *1996*(67), 71-82.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational Thinking in Compulsory Education: Towards an Agenda for Research and Practice. *Education and Information Technologies*, *20*(4), 715-728.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 33-35.

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing Computational Thinking through a Virtual Robotics Programming Curriculum. *ACM Transactions on Computing Education (TOCE)*, *18*(1), 4.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education (TOCE)*, *14*(1), 5.

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

Computational Thinking Implemented in Five Sets of High School Information Technology Textbooks in Mainland China: Comparative Study of Methods and Strategies

Ya- jing GENG[1]*, Feng LI[2]
[1]Department of Education Information Technology, East China Normal University, China
[2]School of Open Learning and Education, East China Normal University, China
51194108027@stu.ecnu.edu.cn, fli@srcc.ecnu.edu.cn

ABSTRACT
The rapid development and popularization of information technology has changed people's behavior and thinking characteristics. Among them, computational thinking is considered to be an indispensable basic ability in life, and computational thinking has been identified as the core literacy of the information technology discipline at the K-12 stage. Implementation and teaching methods are of concern to educational researchers and front-line teachers. Based on the development of computational thinking and computational thinking education, this article focuses on the core concepts and training methods of computational thinking, and refers to the 2017 high school information technology curriculum standards in China, and determines the teaching materials to implement calculations from the orientation and training methods of computational thinking. The four dimensions of thinking are used to compare the implementation of computational thinking in 5 Chinese textbooks, and corresponding teaching suggestions are provided to provide theoretical and practical references for the cultivation of computational thinking in students.

KEYWORDS
computational thinking, high school information technology, comparison of teaching materials, teaching suggestions

Kong, S.C., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Shih, J.L., Sin, K.F., Song, K.S., Specht, M., Sullivan, F., & Vahrenhold, J. (Eds.). (2020). *Proceedings of International Conference on Computational Thinking Education 2020*. Hong Kong: The Education University of Hong Kong.

计算思维在中国大陆五套高中信息技术教材中落实的方法与策略的比较研究

耿雅静[1]*，李锋[2]
[1]华东师范大学教育信息技术学系，中国
[2]华东师范大学开放教育学院，中国
51194108027@stu.ecnu.edu.cn，fli@srcc.ecnu.edu.cn

摘要
信息技术的快速发展与普及改变了人们的行为方式和思维特征，其中计算思维被认为是生活中不可或缺的基本能力，并且计算思维被确定为 K-12 阶段信息技术学科的核心素养，其落实和教学方法被教育研究者和一线教师所关注。本文在梳理计算思维和计算思维教育发展历程的基础上，围绕计算思维的核心理念、培养方式，并参照我国 2017 年高中信息技术课程标准，从计算思维培养指向性和培养方式确定了教材落实计算思维的四个维度，由此来比较五套我国大陆教材计算思维落实情况，并提出相应教学建议，为学生计算思维的培养提供理论和实践的参考。

关键词
计算思维；高中信息技术；教材比较；教学建议

1. 前言
各个国家均将计算思维纳入其 K-12 课程标准中，并被认为是数字化生存一种普适能力。2017 年，我国在新修订的高中信息技术课程标准中将“计算思维”确定为信息技术学科核心素养的一项核心内容，为教材的编写和教师教学提供了标准和依据。那么在教材中，计算思维是通过什么方式来落实的，教师如何利用教材来培养学生计算思维的就显得尤为重要。

2. 新课标视角下信息技术学科计算思维落实的方法与策略

2.1. 计算思维的内涵
新课标将计算思维素养的内涵界定为：在信息活动中，能够采用计算机可以处理的方式界定问题、抽象思考、建立结构模型、合理组织数据；通过判断、分析与综合各种信息资源，运用合理的算法形成解决问题的方案；总结利用计算机解决问题的过程与方法，并迁移到与之相关的其他问题解决中。

2.2. 计算思维的表现
课程标准将计算思维的具体表现总结为在解决问题过程中的形式化、模型化、自动化和系统化四个方面。

2.3. 计算思维再教材中的落实的方法
本文从计算思维培养的指向性（“教计算思维”和“学计算思维”）和方法（“插电”和“不插电”）两个维度建立坐标轴，如图 1，确定了“插电教计算思维”、“不插电教计算思维”、“插电用计算思维”、“不插电用计算思维”这四个比较维度，基于此对教材中计算思维的落实进行梳理与分析。

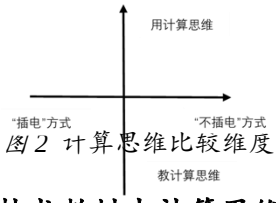


图 2 计算思维比较维度

3. 五套信息技术教材中计算思维落实的比较
“插电教计算思维”在教材中体现为运用计算机等电子设备，在体验程序设计与编码中学习计算机语言、程序设计与编码，软件工具的操作方式等；“不插电教计算思维”体现为通过叙述性课文形式、思考讨论、思维可视化的方式来让学生学习程序设计概念、方法和工具、算法的设计与描述等；“插电用计算思维”主要是让学生在用计算机等设备来实现算法、实现问题解决的关键步骤，在真实情境中体验利用计算机来解决问题；“不插电用计算思维”是在真实情境的项目活动中，将大问题分解成小问题、运用抽象化、模型化、系统化的思维来迭代和优化问题的解决方案，从而优质、高效地解决问题。

本文根据图 1 的分类，选择对人民教育出版社（人教版）、上海科技教育出版社（沪教版）、广东教育出版社（粤教版）、浙江教育出版社（浙教版）和教育科学出版社（教科版）出版的五本《数据与计算》模块内容中计算思维的落实进行比较，具体比较结果如图 2 所示。

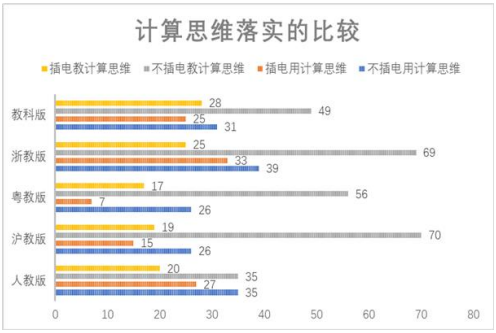


图 2 计算思维落实的比较

对五本教材进行比较发现，人教版和教科版对四种落实计算思维的方式设计的较为均衡，沪教版、粤教版以及浙教版比较看重“通过不插电教计算思维”的方式。其中五本教材也存在共性，即“通过插电的方式用计算思维”的内容在教材中所占比例较低，可以看出新教材真正落实了计算思维不是编程教育的核心理念。并且教材中都尝试通过用项目化学习的方式让学生用计算机解决问题的方式，即在一个完整的系统中将大问题

分解为子问题，让学生在子问题中进行形式化、模型化与自动化的不断迭代。

4. 基于计算思维的教学建议

4.1. 共性建议

各教材的共同目标就是培养创造性思维，提高问题解决能力与效率。故教师要结合教材优势和自身教学经验，加强学生抽象思维和逻辑思维的培养。要合理安排教学活动，让学生在独立分析思考、协作解决问题的过程中将知识转化为能力，结合运用“插电”和“不插电”的方式，充分全方位地发展学生思维。最后，要有效对学生进行评价。

4.2. 个性建议

通过以上落实计算思维的比较研究的结果可以看出，五套信息技术教材存在个性化差异，根据此种情况，教师在使用不同教材时，要结合教材特点，进行相应教学设计，以保证教学的有效开展以及学生计算思维的稳步提升。

5. 总结

本文通过厘清计算思维在教育中应用的基础上，构建了计算思维在教材中落实的比较方式，通过此方式选择了五套正在进行试点使用的大陆教材，比较其计算思维落实的情况，为后续教材的修改、教师的教学以及计算思维的实践应用提供有价值的借鉴。

6. 参考文献

邱美玲、李海霞和罗丹等（2018）。美国《K-12 计算机科学框架》对我国信息技术教学的启示。**现代教育技术，28，41-47。**

范文翔、张一春和李艺（2018）。国内外计算思维研究与发展综述。**远程教育杂志，36(02)，3-17。**

李锋（2018）。中小学计算思维教育:stem 课程的视角。**中国远程教育（综合版），2，44-49，78。**

李锋，熊璋（2017）。面向核心素养的信息技术课程:“数据与计算”模块。**中国电化教育，01，27-32。**

李锋和王吉庆（2015）。计算思维教育：从“为计算”到“用计算”。**中国电化教育，10，6-10，21。**

Bell, T., & Roberts, J. (2016). Computational Thinking is More about Humans than Computers, *Set 2016, 1*, 3–7.

Bocconi. S., Chioccariello. A., Dettori. G., Ferrari, A., & Engelhardt. K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for Policy and Practice*. European Commission: Joint Research Centre.

Caeli, E. N., & Yadav, A. (2019). Unplugged Approaches to Computational Thinking: A Historical Perspective. *TechTrends*, (6), 1-8.

Hermans, F., & Aivaloglou, E. (2017). To Scratch or Not to Scratch? A Controlled Experiment Comparing Plugged First and Unplugged First Programming Lessons. *Proceedings of the 12th Workshop in Primary and Secondary Computing Education*, 49–56.

ISTE & CSTA (2011). *Computational Thinking Teaching in K-12 Education: Teacher Resources (Second edition)*. Retrieved November 10, 2011, from http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf

Wing, J. M. (2011). *Computational thinking: What and Why?* Retrieved November 21, 2019, from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Yang, M. (2017). Why Is Computational Thinking Education Important as the Foundation for Innovation? *Proceedings of International Conference on Computational Thinking Education 2017*. Hong Kong: The Education University of Hong Kong.


CTE
2020

Coo/Think @JC

賽馬會運算思維教育

Inspiring digital creativity 啟發數碼創意

URL:
www.eduhk.hk/cte2020

Email:
cte2020@eduhk.hk

Created and Funded by

香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
同心同步同進 RIDING HIGH TOGETHER

Co-created by

香港教育大學
The Education University of Hong Kong

Massachusetts Institute of Technology

CityU
香港城市大學
City University of Hong Kong