

CoolThink@JC International Conference on Computational Thinking Education 2018

14-16 June 2018



Computational Thinking Education

Conference Proceedings

Created and Funded by



香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
HONG KONG JOCKEY CLUB RIDING HIGH TOGETHER

Co-created by



香港教育大學
The Education University
of Hong Kong



Massachusetts
Institute of
Technology



香港城市大學
City University of Hong Kong

Editors

Siu-cheung KONG

The Education University of Hong Kong, Hong Kong

Diana ANDONE

Politehnica University of Timisoara, Romania

Gautam BISWAS

Vanderbilt University, The United States

Tom CRICK

Swansea University, The United Kingdom

Heinz Ulrich HOPPE

University of Duisburg-Essen, Germany

Ting-chia HSU

National Taiwan Normal University, Taiwan

Ronghuai HUANG

Beijing Normal University, People's Republic of China

Robert Kwok-yiu LI

City University of Hong Kong, Hong Kong

Chee-kit LOOI

Nanyang Technological University, Singapore

Marcelo MILRAD

Linnaeus University, Sweden

Josh SHELDON

Massachusetts Institute of Technology, The United States

Ju-ling SHIH

National University of Tainan, Taiwan

Kuen-fung SIN

The Education University of Hong Kong, Hong Kong

Mike TISSENBAUM

Massachusetts Institute of Technology, The United States

Jan VAHRENHOLD

Westfälische Wilhelms-Universität Münster, Germany

Copyright 2018 The Hong Kong Jockey Club
All rights reserved.
ISBN: 978-988-77034-5-7
ISSN: 2664-035X (CD-ROM) / 2664-5661 (online)
Publisher
The Education University of Hong Kong

Preface

International Conference on Computational Thinking Education 2018 (CTE2018) is the second 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 students to apply digital creativity in their daily lives and prepare them to tackle future challenges in any fields. Computational thinking (CT) is considered as an indispensable capability to empower students to move beyond mere technology consumption and into problem-solving, creation and innovation. This 4-year initiative will educate over 26,000 upper primary students at 32 pilot schools on computational thinking through coding education. Through intensive professional training, the Initiative will develop teaching capacity of over 100 local teachers and help them master coding and computational thinking pedagogy. Over time, the project team targets to make greater impact by sharing insights and curricular materials beyond the pilot schools.

This year, the event is emerged with a 4-day Coding Fair to further outreach parents and students. The first two days are open for schools while the last two days are open for public. Through a series of coding and STEM workshops offered by 32 pilot schools and STEM partners, students aged 4-12 will go through an exciting journey of coding and computational thinking enlightenment. Teachers and students of the pilot schools will also get a chance to showcase their learning outcomes through booth exhibition. As support from parents are always the most important factor in determining the success of education, parent seminars with panel discussions are thus included at the Coding Fair to inspire parents to adapt and master computational thinking as a new bridge for parent-child communication. Over 6500 enthusiastic parents and students are going to join us at the Fair.

“Computational Thinking Education” is the main theme of CTE2018 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. CTE2018 gathers educators and researchers around the world to share implementation practices and disseminate research findings on the systematical teaching of computational thinking and coding across different educational settings. There are 15 sub-themes under CTE2018, namely:

- Computational Thinking
- Computational Thinking and Unplugged Activities in K-12
- Computational Thinking and Coding Education in K-12
- Computational Thinking and Subject Learning and Teaching in K-12
- Computational Thinking and IoT
- Computational Thinking Development in Higher Education
- Computational Thinking and STEM/STEAM Education
- Computational Thinking and Non-formal Learning
- Computational Thinking and Psychological Studies
- Computational Thinking and Special Education Needs
- Computational Thinking and Evaluation
- Computational Thinking and Early Childhood Development
- Computational Thinking in Educational Policy
- Computational Thinking and Teacher Development
- General Submission to Computational Thinking Education

The conference received a total of 60 papers (28 full papers, 22 short papers and 10 poster papers) by authors from 16 countries (see Table 1).

Table 1: Distribution of paper submissions for CTE2018

Country/Region	No. of submissions	Country/Region	No. of submissions
China	10	Malaysia	2
Taiwan	10	Australia	1
The United States	8	Canada	1
Germany	6	India	1
South Korea	6	Norway	1
Hong Kong	5	Spain	1
Singapore	5	Turkey	1
Croatia	2	Total	60

Each paper with author identification anonymous was reviewed by three International Program Committee (IPC) members. Related sub-theme Chairs conducted meta-reviews and made recommendation on the acceptance of papers based on IPC members' reviews. With the comprehensive review process, 44 accepted papers are presented (12 full papers, 23 short papers and 9 poster papers) (see Table 2) at the conference.

Table 2: Review results of submission acceptance for CTE2018

Sub-theme	Full paper	Short paper	Poster paper	Total
CT	2			2
CT and Unplugged Activities in K-12		1	1	2
CT and Coding Education in K-12	2	3	2	7
CT and Subject Learning and Teaching in K-12	1	1	1	3
CT and IoT		1		1
CT Development in Higher Education		2		2
CT and STEM/STEAM Education	2	4	1	7
CT and Special Education Needs	1		1	2
CT and Evaluation	1	5		6
CT and Teacher Development	2	6	2	10
General Submission to CT Education	1		1	2
TOTAL	12	23	9	44

The conference comprises keynote, invited speeches and forum by internationally renowned scholars; seminar, workshop, as well as academic paper and poster presentations.

Keynote and Invited Speeches

There are three keynote and two invited speeches at CTE2018:

Keynote Speeches

1. "Beyond Computational Thinking: Coding, Designing, and Making in the 21st Century" by Prof. Yasmin B. KAFAI, University of Pennsylvania, The United States
2. "The Power behind the Power Point®" by Prof. Judith GAL-EZER, The Open University of Israel, Israel
3. "What Lies Beneath? Towards the Cognitive Underpinnings of Computational Thinking" by Prof. Judy ROBERTSON, University of Edinburgh, The United Kingdom

Invited Speeches

1. “Computational Thinking for Social Change” by Mr. Nawneet RANJAN, Dharavi Diary, India
2. “Computational Thinking Goes to Science and Math Class: The Case for STEM+C” by Ms. Linda SHEAR, SRI International, The United States

Computational Thinking and Future Education Forum

Pioneers and experienced frontline practitioners in local and international education sectors formed a panel to exchange views and ideas on computational thinking and future education.

Panelists:

Principal Tsz-wing CHU, Baptist Rainbow Primary School, Hong Kong

Prof. Heinz Ulrich HOPPE, University of Duisburg-Essen, Germany

Prof. Chee-kit LOOI, Nanyang Technological University, Singapore

Moderator:

Prof. Siu-cheung KONG, The Education University of Hong Kong, Hong Kong

CoolThink@JC Senior Primary Coding Curriculum Dissemination Seminar

To make greater impact by sharing insights and curricular materials to more schools in Hong Kong, CoolThink@JC sheds light on the curriculum, how schools can adopt it and what supports they will get. Pilot schools teachers also share their experience in this seminar.

Speakers:

Prof. Siu-cheung KONG, The Education University of Hong Kong, Hong Kong

Mr. Tony LAM, Marymount Primary School, Hong Kong

Mr. Lee LAU, Baptist Rainbow Primary School, Hong Kong

Mr. Andy LI, Po Leung Kuk Dr. Jimmy Wong Chi-Ho (Tin Sum Valley) Primary School, Hong Kong

Workshop on “Interact with real world: MIT App Inventor and IoT (Internet of Things)”

Massachusetts Institute of Technology conducts a workshop on App Inventor and IoT (Internet of Things), in which the instructor guides participants to design their smart phone app by using MIT App Inventor.

Instructor:

Mr. David Chi-hung TSENG, Massachusetts Institute of Technology, The United States

Academic Paper and Poster Presentations

There are 10 sessions of academic paper presentation and an academic poster presentation with 44 papers (12 full papers, 23 short papers, 9 poster papers) in the conference. Worldwide scholars present and exchange the latest research ideas and findings highlighting the importance and pathways of computational thinking education covering K-12 education, special education, teacher development and STEM/STEAM education etc.

On behalf of the Conference Organizing Committee, we would like to express our gratitude towards all speakers, panelists, as well as paper presenters for their contribution to the success and smooth operation of CTE2018.

We sincerely hope everyone would enjoy and get inspired from CTE2018.

On Behalf of CoolThink@JC

Siu-cheung KONG

The Education University of Hong Kong, Hong Kong

Conference Chair of CTE2018

Tsz-wing CHU

Baptist Rainbow Primary School, CoolThink@JC Resource School, Hong Kong

Conference Chair of CTE2018

Table of Contents

COMPUTATIONAL THINKING.....	1
<i>Full Paper</i>	
A Complementary View for Better Understanding the Term Computational Thinking Marc JANSEN, Dan KOHEN-VACS, Nuno OTERO, Marcelo MILRAD.....	2
The Use of Computational Thinking Concepts in Early Primary School Ivica BOTICKI, Danica PIVALICA, Peter SEOW	8
COMPUTATIONAL THINKING AND UNPLUGGED ACTIVITIES IN K-12	14
<i>Short Paper</i>	
不插电的计算思维教学活动在高中课堂教学中的应用 ——以《二进制卡牌》课程为例 杨冰清，张进宝.....	15
<i>Poster</i>	
Design A Computational Thinking Board Game Based on Programming Elements Sheng-yi WU, Jia-cen FANG, Shu-mei LIAN	19
COMPUTATIONAL THINKING AND CODING EDUCATION IN K-12.....	21
<i>Full Paper</i>	
Analysis of Learner's Self-efficacy using Coding Education Support System for Understanding Complex Problem-Solving Steps In-seong JEON, Hyeon-jeong JEONG, Ki-sang SONG	22
Computational Concepts, Practices, and Collaboration in High School Students' Debugging Electronic Textile Projects Gayithri JAYATHIRTHA, Deborah A. FIELDS, Yasmin B. KAFAI	27
<i>Short Paper</i>	
A School-wide Approach to Infusing Coding in the Curriculum Sirajutheen Shahul HAMEED, Chee-wah LOW, Poh-tin LEE, Nur Illya Nafiza MOHAMED ,Wuay-boon NG, Peter SEOW, Bimlesh WADHWA	33
Learning to Code—Does It Help Students to Improve Their Thinking Skills? Ronny SCHERER, Fazilat SIDDIQ, Bárbara SÁNCHEZ VIVEROS	37
To Improve the Computational Thinking of Elementary School Students by Scaffolding Chien-i LEE, Sheng-chuan CHUANG, Shu-min WU	41
<i>Poster</i>	
A Curriculum and Contents of Programming Education for Computational Thinking Hyojin BYUN, Miyoung RYU, Sungwan HAN.....	45
Comparing with Scratch and Python in CT Concepts Tae-ryeong KIM, Sun-gwan HAN.....	47

COMPUTATIONAL THINKING AND SUBJECT LEARNING AND TEACHING IN K-12.....	49
<i>Full Paper</i>	
Students' Attitude Changes through Integrating Computational Thinking into English Dialogue Learning Xiaojing WENG.....	50
<i>Short Paper</i>	
基于 DBR 的高中生计算思维的培养——以信息技术课程为例 苏幼园，马秀麟，毛荷，王翠霞.....	56
<i>Poster</i>	
Promoting Computational Thinking and Collaborative Skills in Primary Robotics Classes Hyungshin CHOI, Jeongmin LEE	60
COMPUTATIONAL THINKING AND IOT	62
<i>Short Paper</i>	
A Design-based Approach to Implementing a Computational Thinking Curriculum with App Inventor and the Internet of Things Chi-hung TSENG, Mike TISSENBAUM, Wen-hsuan KUAN, Feng-chih HSU, Ching-chang WONG ...	63
COMPUTATIONAL THINKING DEVELOPMENT IN HIGHER EDUCATION	67
<i>Short Paper</i>	
The Use of Computational Thinking to Advance Learning in the Pre-university Subject of Digital Literacies Ildiko VOLCZ.....	68
應用 Scratch 之運算思維教材設計與教學成效分析 楊智為，李政軒，郭伯臣，謝承晏.....	72
COMPUTATIONAL THINKING AND STEM/STEAM EDUCATION	76
<i>Full Paper</i>	
A DSML for a Robotics Environment to Support Synergistic Learning of CT and Geometry Nicole HUTCHINS, Timothy DARRAH, Hamid ZARE, Gautam BISWAS	77
Introducing Computational Thinking Across the Curriculum with Virtual Reality Merijke COENRAAD, David WEINTROP	83
<i>Short Paper</i>	
A Development of a SW-STEAM Education Program using the Flipped Learning Hae-nam SONG, Sun-gwan HAN	89
Development of BIC-Science Module: An Interdisciplinary Approach of Computer Science and Primary Science Education Tracy MENSAN, Kamisah OSMAN.....	93
Thinking in Parts and Wholes: Part-Whole-Thinking as an Essential Computational Thinking Skill in Computer Science Education Nils PANCRATZ, Ira DIETHELM.....	97
創客奇航-遊戲任務導向之運算思維活動設計初探 黃淑賢，陳虹如，葉芯妤，蔡一帆，施如齡.....	101

<i>Poster</i>	
Examining a Secondary School Computational Action Curriculum Using App Inventor and the Internet of Things	
Mike TISSENBAUM, Josh SHELDON, Hal ABELSON, Mark SHERMAN.....	104
COMPUTATIONAL THINKING AND SPECIAL EDUCATION NEEDS	106
<i>Full Paper</i>	
The Application of Minecraft in Education for Children with Autism in Special Schools	
Wen-wen MU, Kuen-fung SIN.....	107
<i>Poster</i>	
結合運算思維在國小特殊教育需求的數學教學活動之發展	
廖晨惠，郭伯臣，白鎧誌，鄔珮甄.....	112
COMPUTATIONAL THINKING AND EVALUATION	114
<i>Full Paper</i>	
Evaluating Computational Thinking in Jupyter Notebook Data Science Projects	
Clara SORENSEN, Eni MUSTAFARAJ.....	115
<i>Short Paper</i>	
Assessment of Computational Thinking	
Nikolina BUBICA, Ivica BOLJAT.....	121
Cross Comparison of Multiple Computational Thinking Activities: a Grey-based approach	
Meng-leong HOW, Chee-kit LOOI	125
On Tools that Support the Development of Computational Thinking Skills: Some Thoughts and Future Vision	
Gregorio ROBLES, Jean Carlo Rossa HAUCK, Jesús MORENO-LEÓN, Marcos ROMÁN-GONZÁLEZ, Roberto NOMBELA, Christiane Gresse von Wangenheim.....	129
计算思维评估的研究现状综述（2013-2017）	
张安琦，陈枕，刘昱辛，程薇.....	133
基於專家知識地圖引導慕課學習思維	
曾建維，黃能富，李加安.....	137
COMPUTATIONAL THINKING AND TEACHER DEVELOPMENT	140
<i>Full Paper</i>	
Computational Thinking Reshapes the Teachers' Perspective on Human Mind towards Teaching and Learning Process	
Hew-mee CHEAH	141
Teacher's Perceptions and Readiness to Teach Coding Skills: A Comparative Study between China, Finland and Singapore	
Chee-kit LOOI, Jari MULTISILTA, Longkai WU, Pauliina TUOMI	147
<i>Short Paper</i>	
“It Opens Up a New Way of Thinking, but...”: Implications from Pre-Service Teachers' Introduction to Computational Thinking	
Yu-hui CHANG, Lana PETERSON.....	153

The Readiness of Computational Thinking Education in Taiwan: Perspectives from the K-12 Principals in 2017	
Ting-chia HSU	157
Two Studies of Perceived and In-Situ Readiness for Implementing the Computing Education in Singapore	
Longkai WU, Chee-kit LOOI, Meng-leong HOW, Liu LIU	161
中學資訊科技教師運算思維學科教學能力調查	
蔡旻穎，吳正己，游志弘	165
基于计算思维培养的教师培训课程设计与实践	
刘昱辛，陈桃，查思雨，张安琦	169
國小師資生 Code.org 運算思維課程實作與成效探討	
李政軒，楊智為，郭伯臣	173
<i>Poster</i>	
Designing Computational Thinking Assessment: A Case Study of a Pre-Service Teacher Course in Korea	
Mi Song KIM, Hyungshin CHOI	177
Which Parts of Computer Science Concepts Do Future Teachers Identify? First Results of a Part-Whole-Thinking Analysis in Computer Science Education	
Nils PANCRATZ, Ira DIETHELM	179
GENERAL SUBMISSION TO COMPUTATIONAL THINKING EDUCATION	181
<i>Full Paper</i>	
Developing a Framework for Computational Thinking from a Disciplinary Perspective	
Joyce MALYN-SMITH, Irene A. LEE, Fred MARTIN, Shuchi GROVER, Michael A. EVANS, Sarita PILLAI	182
<i>Poster</i>	
Virtuality Literacy: On the Representation of Perception	
Andreas DENGEL	187

Computational Thinking

A Complementary View for Better Understanding the Term Computational Thinking

Marc JANSEN^{1,2}, Dan KOHEN-VACS^{2,3}, Nuno OTERO^{2,4}, Marcelo MILRAD²

¹University of Applied Sciences Ruhr West, Bottrop, Germany

²Linneaus University, Växjö, Sweden

³Holon Institute of Technology (HIT), Holon, Israel

⁴Instituto Universitário de Lisboa (ISCTE-IUL), Lisboa, Portugal

marc.jansen@hs-ruhrwest.de, dan.kohen@lnu.se, nuno.otero@lnu.se, marcelo.milrad@lnu.se

ABSTRACT

The term *Computational Thinking* is closely related to efforts connected to teach a systematic and well-structured way of problem solving that includes a set of tools and techniques used in Computer Science. While substantial research in this field has shown promising outcomes concerning distinct intervention programs and teaching initiatives, the term Computational Thinking itself requires to be revised in order to get a wider consensus about its meaning and purpose. This paper contributes to the ongoing quest concerning the definition of the term by starting with a fundamental perspective on computational theory and corresponding concepts in order to describe the theoretical building blocks of a systematic view to further elaborate on an approach for teaching and learning about Computational Thinking. Additionally, based on this foundational effort, more advanced concepts are presented and discussed in order to better understand this domain. Finally, the paper identifies and discusses a set of relevant challenges taking a cognitive psychology perspective on Computational Thinking.

KEYWORDS

Computational Thinking, 21st century skills, computability, cognitive psychology, knowledge transfer, multiple external representations.

1. INTRODUCTION

Many developed countries are experiencing significant changes concerning organizational structures, work processes and daily routines. Technological innovations impact daily practices regarding the ways people socialize, work and administrate their activities (Kulkarni, 2017). Many of these changes have been enabled and are supported by new Information and Communication Technologies (ICT) (Horizon Report, 2017). The demands that this changing societal context pose are being reflected in new educational programs that aim at offering students an updated set of skills identified as crucial for the 21st century (Trilling & Fadel, 2009). In fact, competences concerning critical thinking as well as problem solving are seen as central in contrast to other competences often considered less useful to cope with the fast pace of current changes. Some researchers suggest that the acquisition of these skills should be provided in authentic settings (Doleck et al., 2017), where technologically supported creative

collaborative activities are proposed (Kong, 2014; Mishra et al., 2013).

In line with these developments, computer and learning scientists have been proposing that these skills can be fostered through educational programs involving computer programming and miming how computer scientists approach problem solving (Wing, 2014), including the expression of a solution in a computer solvable way. The term *Computational Thinking* (CT) has emerged to reflect this particular view on this topic. However, although the term is widely used, it requires to be revised in order to get a wider consensus about its meaning and purpose within the scientific community (Selby & Wollard, 2014).

This paper starts with a section describing the state of the art, after which we reflect on the term CT from a computer science theory perspective. We do so in order to identify the basic building blocks that allow problems to be framed and solved computationally. Thereafter, the next section discusses more advanced computer science topics related to CT. Concrete examples are shown and discussed in order to elaborate on the proposed ideas. We proceed by discussing the core ideas presented in the paper in order to widen the definition of Computational Thinking. Finally, we proceed by identifying and describing a set of relevant challenges for CT teaching and learning from a cognitive psychology perspective. We conclude the paper with a section providing an outlook describing our future efforts.

2. STATE OF THE ART

The benefits of computer programming for students' cognitive development have been explored and are well recognized. Clements & Gullo (1984), foresaw the important role of ICT in daily routines. They also examined the effects of computer programming on children while indicating on advantages in terms of development of cognitive skills (Papert, 1980). Additional approaches that would nowadays be gathered under the term CT have been made way before the definition of the term itself (Brennan, 2012; Resnick et al., 1998).

In addition, the benefits of computer programming were also recognized in terms of its potentials to foster creativity and meta-cognitive skills exercised as part of development tasks. More than 30 years ago, Pea & Kurland (1984) published the results of their research exploring aspects that are crucial for incorporating computer programming in educational studies. They pointed over challenges while expressing their concern about practicing such skill among

young ages. They also addressed goal definitions aligned to the requirements and knowledge that are needed prior and during the development of cognitive skills supported by computer languages. Additionally, they investigated the benefits of such skills in the light of individual work versus collaborative one.

As implied, nowadays, our daily routines are technologically enhanced in a way that emphasize the important role of programming as a tool to aid structured thinking processes as well as a tool for the implementation of solutions based on ICT. Consequently, *Computational Thinking* as an innovative approach for solving problems is increasingly recognized and incorporated in educational programs that need to be implemented across different subject matters and levels (Kong, 2014). This solution could be conceptualized and formulated in the form of a computer program expressing logical procedures towards a refined solution. CT offers the opportunity to exercise a generic and iterative process consisting of three steps. In the first step, students are provided with an educational opportunity to identify and formulate a problem or challenge on an abstract level. Thus, students can formulate the problem in a more generalized (and at the same time easier) way and try to solve this more general problem first. During the second phase, they can continue and express a possible solution to it. Finally and in the third phase, this solution is executed and evaluated as a part of the iteration enabling continuous refinement aspiring to optimized problem definition adapted with best solution.

Often, learning environments and activities guided by the ideas behind Computational Thinking incorporate motivational tools like robots (Bers et al., 2014) in order to increase students' motivation to work in a structured way and to provide procedures that support the solution of a given problem. Although Computational Thinking could be applied already to very early ages, significant efforts have also been undertaken in relation to older students, which have been proven successful also (Grover and Pea, 2013; Touretzky et al., 2013).

Nevertheless, as Selby & Wollard (2014) have described, the term Computational Thinking has several different connotations and it is used throughout literature in very different ways. Those different ways basically differ in the understanding of CT in terms of the definition of thinking, problem solving, computer science and imitation. Therefore, this paper makes an attempt to provide a distinct and complementary perspective to CT, based on computational theory. Starting from computational theory concepts, we move on by taking a step forward to more advanced topics that derive from the field of programming, based on the theory mentioned above.

3. THEORETICAL BACKGROUND

As already indicated earlier in the paper, one interpretation of the term "Computational Thinking" is that it refers to solving computational problems in the way computers do. In order to define what these kinds of problems are, it is worth looking to the definition of Alan Turing about computability (Turing, 1937). While Gödel (1931) already proved that there are theories in every axiom system that are

not provable, and therefore not computational, Turing proposed a formal definition of computational theorems by the definition of the Turing Computable Functions also referred to as Turing complete functions. Here, Turing Complete Functions, are functions that could be solved by a Turing Machine. According to the Church's theorem (Turing, 1939) the set of naive computable functions equals the set of Turing Computable Functions. Therefore, it could be said that every problem that is solvable, could be solved by a Turing Machine. Hence, one complementary perspective to the existing one on CT could be to have a look at the mechanisms that are used by Turing Machines and other approaches to computability in order to solve those kinds of problems. Especially, the theory of μ -recursive functions, loop-, while- and goto-computability are those under consideration. Analyzing these fundamental theories of computational functions, it shows that there are a couple of concepts necessary in order to address and tackle problems that are solvable by computers:

- conditions - as in Turing Machines in the form of the transition function
- loops - as in loop- and while-computable functions
- goto / subroutines - as in goto-computable functions
- recursion - as in μ -recursive

The following subsections will provide a short overview on the implications that the different concepts might have for teaching and learning Computational Thinking.

3.1. Conditions

Conditions basically allow for the distinction of cases. Usually also referred to as if-this-then-that (IFTTT), conditions allow to treat different states of a (sub)problem differently. States are usually expressed / modelled in the form of Boolean expressions. Often, those conditions also have an else part, that is executed if a certain Boolean expression does not hold. It could easily be shown, that the existence of an else part does not yield to more functions that are computable. A simple example for a condition that checks if a given number is even could be implemented in Scratch as shown in Figure 1. Scratch will be our visual programming language of choice for the remaining examples also, since we believe that it is widely accepted and at the same time easy enough to understand even if the reader does not have any pre-knowledge here.

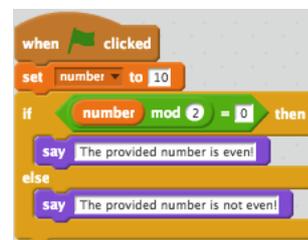


Figure 1. A simple condition in a visual programming language.

Interestingly, the way to model computer programs as a set of IFTTT expressions lately became more and more prominent, e.g., in the field of the Internet-of-Things (IoT) and / or blockchain based technologies. Both examples provide highly up to date questions, in which a large number of scenarios could be implemented based on simple IFTTT

conditions. This underlines the importance and power of this kind of modelling.

3.2. Loops

Loops are a means for repeating a certain task. Usually, two different types of loops are used in computational theory: count controlled loops (in which a certain task is executed defined times) and condition-controlled loops (in which the task is executed as long as a certain condition holds). It could easily be shown that count controlled loops could be expressed also as case-controlled loops, but not the other way around. Therefore, it could be said that the concept of case-controlled loops is richer than the concept of count controlled loops. Nevertheless, count controlled loops are often easier to understand since counting is a very basic task, while conditions are a bit trickier.

An easy to implement example based on a count-controlled loop is the Fibonacci number. A Scratch based implementation might look similar to the block shown in Figure 2.

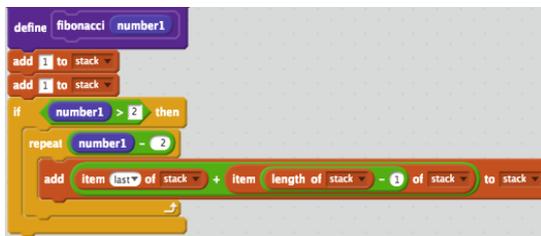


Figure 2. A count controlled loop in a visual programming language.

3.3. Goto / subroutines

Another class (equivalent to the condition-controlled loops) are Goto Computable Functions. Goto constructs basically allow to jump to certain parts of a program, while in contrast Turing Machines need to work sequentially through their memory. Although, as said before, the class of Goto Computable Functions are equivalent to the class of functions that can be computed with condition-controlled loops, the concept is worth noticing, because it provides a first way for implementing subroutines. Historically, this could best be seen in languages like Basic, which introduced (at least in some dialects) an additional keyword gosub (beside Goto) in order to allow for subroutines in Basic programs.

Subroutines are usually used in order to allow a re-use of the implemented functionality. Taking the example from above for checking if a given number is even or not, a subroutine that could be re-used could be implemented as new block in Scratch as shown in Figure 3.



Figure 3. A subroutine defined as a block in a visual programming language.

3.4. Recursion

Finally, after discussing that conditions and loops are the basic control structures of computational functions, another

mechanism also needs to be discussed. Although recursion is at first a mathematical mechanism used for functions that call themselves, it could also be used as a control structure since it influences order commands executed by a program. Beside this, it is a very powerful mechanism to describe some mathematical functions, e.g., the famous Fibonacci number. It could be shown that the class of primitive recursive functions is equivalent to the class of functions computable by count-controlled loops, which especially means that every primitive recursive function could also be expressed as a count-controlled loop. Taking up the example of the Fibonacci numbers based on a count-controlled loops as shown in 2, the corresponding implementation based on recursion looks like presented in Figure 4.

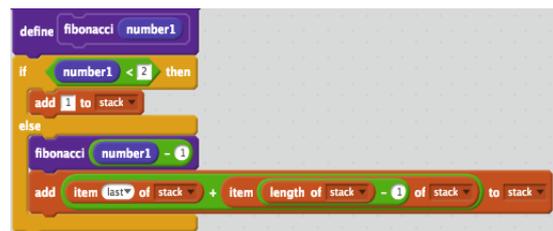


Figure 4. A recursive function implemented in a visual programming language.

Here, an interesting task from a CT perspective could be to switch the representation of simple functions from their recursive representation to a solution based on a count-controlled loop and vice-versa, in order to foster the understanding of both concepts. It is further known that the class of μ -recursive functions is equivalent to the class of function computable by condition-controlled loops.

As mentioned at the beginning of this section, the concepts presented here can be seen as the building blocks for enabling to frame a complementary way to solve problems from a computational perspective. Therefore, in contrast to more traditional views to CT that take a standpoint from social and behavioral sciences our approach results from a computational theory perspective. This proposed view aims to expand the current definition of CT by bringing central ideas and views based on this theory. In other words, the concepts discussed here are the fundamental tools that allow computer scientists think with in order to frame problems and explore solutions (and the fact that computational approaches are being used in very different domains with success supports its value). One of the key ideas behind CT is that this specific way of framing problems can be introduced to learners from an early stage and as such it has the potential to enhance their problem-solving skills in a variety of domains. The next section further elaborates on other useful concepts that extend this perspective.

4. MORE ADVANCED TOPICS BASED ON THE DESCRIBED THEORY

In the previous section, we presented stepping stones enabling the framing for solving problems taking a computational theory perspective. In this section, we go beyond the stepping stones referred to in the previous section and present a set of additional concepts to be offered to learners as tools applicable for their reasoning process on problems. More specifically, the more advanced topics that will be discussed are Object Orientation, Frameworks and

Design Patterns. The presentation and discussion of these ideas are illustrated through the implementation of an algorithm known as bubble sort capable of sorting a set of objects in a given list. The sorting process is achieved through repeated steps in which a pair of objects are compared and if necessary swapped. As implied, bubble sort includes steps that use the concepts previously introduced including conditions and loops. This particular example is provided in order to illustrate our particular and complementary view on CT.

4.1. Object Orientation

In this subsection, we address Object Oriented structures including their properties and functions built based upon concepts previously presented. Specifically, we propose to use them in order to describe objects that may interplay in cases in which learners aim to solve a given problem.

Here, an object is basically a combination of a data structure, together with methods operating on the data structure. Figure 5 shows an object representing a sorter responsible sort a list of numbers. Figure 5, provides an example of an Object-Oriented implementation made in Scratch. In this implementation, we included a method that gradually sort neighbor pairs of numbers till the list is completely sorted. In each iteration, a pair of number is sorted by another function operating according to the swap principal demonstrates in the previous subsection.

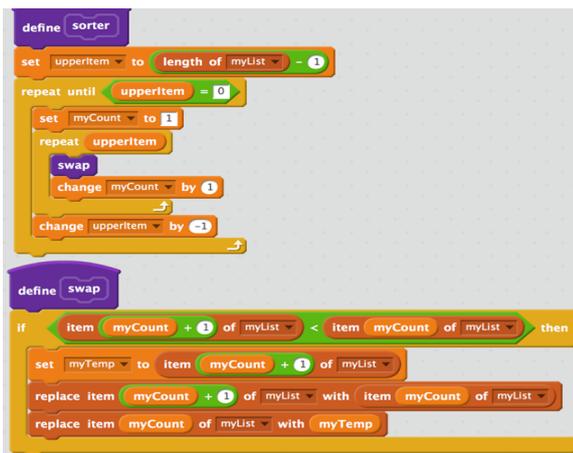


Figure 5. A simple object of a sorter in a visual programming language.

4.2. Frameworks

Frameworks are referred to an abstraction level in a way that enables to provide generic functionality that could be altered, deployed, implemented and reused to satisfy specific aspects of a problem. When discussing frameworks in the context of programming, this may include various components including libraries, compilers and APIs consolidated in order to enable development of complete systems. In our case, a sorter algorithm could be offered as a generic framework which represents a service implementable in systems requiring such kind of functionality.

4.3. Design Patterns

Design patterns represent general and reusable problem solution pairs for commonly occurring problems. The notion of sorting strategies in the light of Design Patterns

has been discussed by Nguyen & Wong (2001), while indicating on best ways to apply different strategies for sorting challenges. They specifically addressed different aspects of a typical sorting challenge including the interplay between involved objects, the selection of an optimized solution for sorting and ways to visualize the result of this sorting. They emphasized on the Model–view–controller (MVC) pattern enabling a separation of concerns between models (data to be sorted and the sorting algorithm itself), views (presentation layer for presenting the solution of a sort algorithm) and controller (logic layer responsible for connecting the model and the view).

In the last two sections, we present our perspective addressing various and central concepts later elaborated through advanced topics reflecting additional tools and techniques used in Computer Science. In the next section, we elaborate on the challenges related to cognitive perspectives on Computational Thinking. These ideas take into consideration the fact that CT approaches need to be implemented across different levels and subject matters.

5. CHALLENGES FROM A COGNITIVE PSYCHOLOGY PERSPECTIVE

We need to be aware that empirical evidence clearly showing the connection between learning how to program and improving reasoning and analytical skills is still scarce (see for example, Pea & Kurland, 1984, or Salomon & Perkins, 1987, for detailed reviews concerning the previous efforts on psychology of programming). Although CT goes beyond teaching how to program we must take on board the issues raised and incorporate these in a research program. Although revisiting all these topics is beyond the scope of this paper, when considering the teaching of a particular subject matter, a cognitive psychology perspective needs to account for two basic interconnected issues: what to teach at distinct stages of human development and how to teach it. However, the teaching of Computational Thinking poses particular challenges because it is not only a subject matter per se but it is intended to be a thinking tool that allows a distinct way to frame and tackle problems emerging from different disciplines. In other words, it is close to what has been termed as a transdisciplinary effort.

In relation to the concepts to teach and its suitability in relation to the different stages of human development, we have identified two main challenges:

- 1) Identify suitable and meaningful problems to the age group, enabling the introduction of the main concepts at an early stage and be able to iteratively refine them with increased levels of complexity.
- 2) Find appropriate ways to ensure transfer of knowledge between domain areas that utilized concepts from computational thinking.

Regarding the identification of suitable and meaningful problems for a certain age group, this stance clearly aligns itself with the early proposals by Bruner (1960) and Papert (1980) that rejected closed notions of development stages, considering that such approaches might miss the opportunity to introduce concepts at early stages and be able to leverage from it (see for example, Bruner, 1960; Papert,

1980; Pea & Kurland, 1984, Resnick, 1984; Resnick et al., 1998). Actually, current experiences with spiral approaches to programming curriculum development suggest that this is indeed possible (Armoni, Meerbaum-Salant & Ben-Ari, 2015). In relation to the problem of ensuring efficient transfer of knowledge, this is indeed an old question with distinct theories and conceptual approaches being proposed (a proper review of the theme is beyond the scope of this paper, however, see for example, Bransford & Schwartz, 1999) but CT might be, in fact, an enabler since it provides the necessary conceptual tools for connecting across domains. Nevertheless, a successful knowledge transfer approach for CT will need to include the following aspects: a) encourage the use of analogies so that the learners are stimulated to explore potential connections between subject matters, b) avoid excessive focus on the contextualization of problems so that learners are not submerged on detail and fail to abstract, and c) provide the necessary tools that facilitate abstraction in relation to the core concepts of computation.

Two other crucial aspects concern how to teach the different concepts and which tools seem suitable to support this process. In our perspective, the teaching of computational thinking needs to be closely tied to the learning activity of modelling distinct phenomena. Encouraging students to construct models of different phenomena is a well-established educational activity (see for example, Milrad, Spector & Davidsen, 2002 and Pinkwart, 2005). However, models can be of very distinct types, from qualitative to quantitative, using graphical/pictorial symbolisms and/or formal notations. From a psychological perspective, there is an ongoing debate regarding the way the distinct types of models can and should be integrated, not only in relation to the age group of the learner but also to the actual stage of problem comprehension. Considering the different notion involving computational thinking we need to assume that at some point learners will need to specify the model in such a way that it is amenable to computing. Thus, we believe, it requires some fair degree of formalization and such will need to be in line with the cognitive skills of the learners. Relevant questions that can be posed are then:

What are the appropriate levels of formalization for the models considering the age group, cognitive skills and previous knowledge of the learners?

How to ensure that the increasing levels of formalisms sophistication are clearly followed through by learners (in other words, do the learners understand the connections between the distinct formalisms)?

In fact, the aspect concerning learners' understanding of the distinct levels of formalisms sophistication also connects with the notion of using multiple external representations to foster the learning of computational thinking. The transdisciplinary nature of computational thinking themes clearly suggests the use of a varied range of external representations (some connected with computational concepts and some connected with the particular domains under scrutiny). But as previous research pointed out being able to establish the connections between distinct external representations is far from trivial (Ainsworth, 1999; Ainsworth & Van Labeke, 2004). Research needs to

account on how different external representations combine, looking for synergies and clearly justify cost/benefits of using them. Nevertheless, multiple external representations can support deeper understanding by promoting processes such as abstraction, extension or generalization of knowledge especially if efficient highlighting of the links between different representations is in place. Finally, the evaluation of computational thinking approaches need to consider not only the outcomes of learning events but also the processes. Relevant questions are: a) How to capture the learners' skills regarding the transfer of knowledge? b) How to capture and understand learners' representational skills in different educational contexts? c) What methods are particularly suited to account for a) and b) at distinct stages of human development?

6. OUTLOOK AND FUTURE WORK

This paper revisited the core concepts of computational theory and how these are related to the notion of CT. By doing so, we contributed to the clarification of the ongoing discussion around the term "Computational Thinking". While most common definitions result from an elaboration that takes social and behavioral sciences as a point of departure, we have used a computer science theory view and added a cognitive psychology perspective afterwards. In some sense, this might help us to re-focus on the fundamental concepts to be taught from a subject matter perspective. Then, we can identify, based on existing literature and empirical evidence produced, how to teach these. Additionally, we provided concrete computationally relevant instantiations of the concepts discussed in section 3 including conditions, loops, goto/subroutines and recursion. In this respect, we also addressed more advanced topics including Object Orientation, Frameworks and Design-Pattern.

The issues that have emerged from our reflections regarding these themes lead us to consider the following key broad steps: (a) identify the topics in distinct subject matters that are particularly suitable to be included in an initial curriculum sketch that implements the core computational concepts we referred to. This task should be carried out in close collaboration with experts in distinct subject matters and teachers of learners in different key stages; (b) reflect and create a pedagogical approach that takes into consideration the different issues stated as challenges from a psychological perspective and provide solid empirical evidence. In relation to relevant empirical studies we are considering starting with issues related to knowledge transfer and the use of different representations to support it; (c) design an intervention in order to evaluate how a pedagogical approach can be successfully implemented in an authentic context; and (d) implement a comparative evaluation study that will endeavor to clarify the putative benefits of the approach and contribute with empirical data to facilitate further refinements. Focusing on the subjects of Mathematics, Natural Science and Technology in grades 4-9, we are currently exploring and validating the ideas described in this paper in our ongoing projects with elementary school in-service teachers.

Acknowledgement

The work reported in this paper was partially funded by the EU project eCraft2Learn (Grant Agreement no. 731345).

7. REFERENCES

- Ainsworth, S. (1999a). A functional taxonomy of multiple representations. *Computers and Education*, 33(2-3), 131–152.
- Ainsworth, S., & VanLabeke, N. (2004). Multiple forms of dynamic representation. *Learning and Instruction*, 14(3), 241–255.
- Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to “Real” Programming. *Trans. Comput. Educ.*, 14(4), 25:1–25:15.
- Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011, March). Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 245-250). ACM.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, 72(C), 145–157. <http://doi.org/10.1016/j.compedu.2013.10.020>
- Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *Rev Res Educ*, 24, 61–100.
- Brennan, K., of, M. R. O. T. 2. A. M., 2012. (n.d.). New frameworks for studying and assessing the development of computational thinking. *Scratched.Gse.Harvard.Edu*.
- Bruner, J. S. (1960). *The Process of education Cambridge, Mass.: Harvard University Press.*
- Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of educational psychology*, 76(6), 1051.
- Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, 4(4), 355-369.
- Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, Band: 38, Nummer: 1, 173-198.
- Grover, S., Pea, R. (2013). Computational Thinking in K12 A Review of the State of the Field. *Educational Researcher* 42(1), 38-43
- Horizon report, (2017). Retrieved from <http://cdn.nmc.org/media/2017-nmc-horizon-report-HE-EN.pdf>
- Kong, S. C. (2014). Developing information literacy and critical thinking skills through domain knowledge learning in digital classrooms: An experience of practicing flipped classroom strategy. *Computers & Education*, 78, 160-173.
- Kulkarni, C. (2017, January 3). *15 Trends Every Business Leader Should Watch in 2017*. Retrieved January 01, 2018, from <http://fortune.com/2017/01/03/2017-tech-trends/>
- Milrad, M., Spector, J. M., & Davidsen, P. I. (2002). Model Facilitated Learning. *Learning and Teaching with Technology: Principles and Practices* (pp. 13–27). London, UK and Sterling, VA, USA: Kogan Page Publishers, UK.
- Mishra, P., Yadav, A., & Deep-Play Research Group. (2013). Rethinking technology & creativity in the 21st century. *TechTrends*, 57(3), 10-14.
- Nguyen, D., & Wong, S. B. (2001). *Design patterns for sorting* (Vol. 33, No. 1, pp. 263-267). ACM.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New ideas in psychology*, 2(2), 137-168.
- Pinkwart, N. (2005). *Collaborative modeling in graph based environments* (pp. I-VIII). University of Duisburg-Essen, Germany.
- Resnick, M. (1994). *Turtles, Termites, and Traffic Jams*. Cambridge, MA: MIT Press.
- Resnick, M., Martin, F., Berg, R., Borovoy, R., Colella, V., Kramer, K., & Silverman, B. (1998). Digital Manipulatives - New Toys to Think With. *Chi*, 281–287. <http://doi.org/10.1145/274644.274684>
- Selby, C. & Wollard, J. (2014). Refining an understanding of computational thinking. Retrieved 01, 08, 2018 from <http://eprints.soton.ac.uk/id/eprint/372410>
- Touretzky, D. S., Marghita, D., Ludi, S., Bernstein, D., & Ni, L. (2013). Accelerating K-12 computational thinking using scaffolding, staging, and abstraction. *Sigcse*, 609. <http://doi.org/10.1145/2445196.2445374>
- Trilling, B., & Fadel, C. (2009). *21st century skills: Learning for life in our times*. John Wiley & Sons.
- Turing, A.M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*. Vol 42, 230–265.
- Turing, A.M. (1939). *Systems of Logic Based on Ordinals*. Princeton University. p. 8.
- Wing, J. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014.

The Use of Computational Thinking Concepts in Early Primary School

Ivica BOTICKI^{1*}, Danica PIVALICA¹, Peter SEOW²

¹ Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia

² National Institute of Education, Nanyang Technological University, Singapore
ivica.boticki@fer.hr, danica.pivalica@fer.hr, peter.seow@nie.edu.sg

ABSTRACT

This paper presents a study on the use of computational thinking (CT) in early primary school. First grade primary school students were using a custom designed CT tool as part of their school lessons. The tool allowed for designing and delivering digital tasks with CT content coming from three subject areas with the goal of finding out how well students complete such tasks. The tasks were aligned with course contents and the curriculum and required students to choose, set or order CT primitives in an adequate way. The tool allowed for automated task solution evaluation in form of animations and visualizations reflecting the exact steps chosen by the students and prompting them to revise their choice if needed. The analysis of students' task completion process reveals that CT tasks including object properties and problems with loops were the most demanding, and that prior mathematics and reading skills impact early primary students' CT task completion performance across school subjects.

KEYWORDS

computational thinking, mobile learning, early primary school, student performance, mathematics

1. INTRODUCTION

Mindstorms is a book written by Seymour Papert in which he argues for the benefits of teaching computer literacy (Papert, 1983). It was in this book that the term computational thinking (CT) was coined and since then modern CT initiatives have become the subject of worldwide attention. Due to the profound transformation of today's society sparked by the rapid progress of digital technology, many educators and leaders started becoming interested in incorporating CT into education.

Even though CT has a rich history, its broader recognition began in 2006 with an essay by Jeannette Wing (Wing, 2006) in which she revived the previously coined CT phrase. CT was described as a general-purpose thinking tool which builds on natural and artificial information processes, and is about acknowledging limits in available resources, reducing problems to smaller parts, abstracting information and choosing appropriate problem and (or) solution representations. A few years later, the Cuny-Synder-Wing definition was proposed describing CT 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" (Wing, 2010).

Brennan and Resnick (Brennan & Resnick, 2012) argue there are three dimensions of CT: computational concepts,

computational practices and computational perspectives. Computational concepts are congruent with the fundamental concepts of programming languages. Computational practices refer to practices developed during CT activities while computational perspectives show in what way has learner's viewpoint changed after engaging in CT activities.

Regardless of CT receiving considerable attention nowadays, there is little agreement on how it should be integrated and used in primary and secondary education. This paper aims at discussing the use of key CT concepts as proposed within Brennan and Resnick's framework (Brennan & Resnick, 2012) with young primary school learners, with a special emphasis on the computational concepts dimension. A custom CT tool for early primary school learners along with the accompanying CT tasks covering the subject area of Mathematics, Science and Croatian language is proposed. This study examines how early primary school students completed CT tasks which include a variety of CT concepts and which were designed to cover the three subjects' contents.

2. COMPUTATIONAL THINKING CONCEPTS

Some researchers have adopted an assessment approach to evaluating computational thinking through code generated by students. Brennan and Resnick proposed that students' computational thinking competencies can be assessed through how they engage with computational concepts found in Scratch programming environment. The computational concepts are sequences, loops, parallelism, data, events, and conditionals (Brennan & Resnick, 2012). These computational concepts convey relevant vocabularies and notations to be used to describe computational processes such as ordering a list or how to do multiplication (Lu & Fletcher, 2009). For example, multiplication can be described as the number of loop iterations needed to add up the same number.

Students use computational concepts to develop projects such as stories, animations, games, tutorials and musical instruments through programming (Resnick et al., 2009; Ruthmann, Heines, & Greher, 2010). Based on this, Moreno-León and his colleagues developed Dr Scratch to give feedback on different dimensions of computational thinking competency to teachers and students in their Scratch projects. The dimensions measured are abstraction, logical thinking, parallelism, data representation and algorithmic sequencing (Román-González, Pérez-González, & Jiménez-Fernández, 2017).

Table 1. CT tasks organized into five CT task groups. Each task covers one or two school subjects and up to several CT concepts.

Group number	Task number in group	School subjects	CT concepts
1	1	Croatian language, Science	Sequence, algorithm, recognition and removal of unnecessary steps
	2	Croatian language, Science	Sequence
	3	Croatian language, Science	Sequence, algorithm
	4	Croatian language	Sequence, algorithm
	5	Croatian language	Sequence
2	1	Science	Object and its properties (sparrow)
	2		Object and its properties (frog)
	3		Object and its properties (bat)
	4		Object and its properties (hedgehog)
	5		Object and its properties (rabbit)
	6		Sequence, object and its properties
3	1 - 4	Mathematics	Problem task (selecting steps of a path)
	5 - 8	Croatian language, Mathematics	Problem task (selecting the right steps of a path and identifying the correct goal)
	9 - 10	Croatian language, Mathematics	Problem task with loops (selecting the right steps of a path and identifying the correct goal)
4	1 - 8	Mathematics	Problem task (numbers 1-10)
5	1 - 3	Mathematics	Problem task (numbers 11-19)
	4 - 7	Mathematics	Problem task with loop (numbers 11-19)
	8	Mathematics	Problem task with combining two loops (numbers 11-19)

This study focuses on the *computational concepts* CT dimension and examines several CT concepts implemented as CT Mathematics, Science and Croatian language subject tasks. Table 1 shows five groups of CT tasks which are aligned with the school subjects. By building on the presented state-of-the-art research in the field, a CT tool in form of a scaffolded environment with visualization and animation feedback on the proposed CT task solutions is designed, implemented and examined in early primary school contexts.

3. METHODS AND TOOLS

The study was conducted on a sample of 23 primary first grade students 7 to 8 years old, who study in a neighborhood primary school in Croatia. There were 12 female and 11 male students in this study. The study included five groups of computational thinking tasks (Table 1) and was carried out within the period of two months (May-June 2017), with each task group taking place on a single day and taking 2-3 hours of direct student time. Each student was using an individual tablet of his or her choice (an Android, iOS or Windows tablet) to complete the tasks.

Multiple tasks per group were designed with the help of the class teacher, so that they relate to the courses taught in first grade of Croatian primary schools and make use more contextualized and meaningful to the students.

To scaffold task delivery and collect usage data, a CT tool in form of a block-based visual environment in which students drag-and-drop blocks into a scripting pane to build a solution was designed. Such an environment is inspired by similar research targeting young children (Wilson & Moffat, 2010) and should reduce the efforts and challenges of learning programming and the underlying computational concepts such as sequence or objects. The tool included a narrative in form of a virtual character named Matko the robot, guiding students in the CT tool usage.

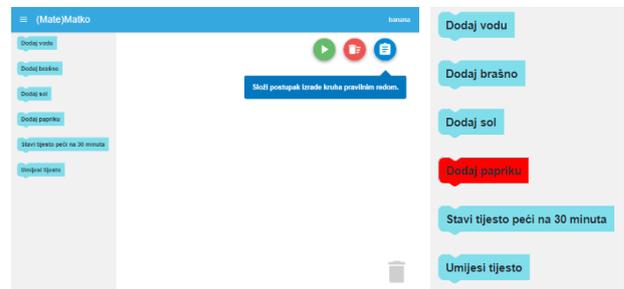


Figure 1. Left hand side: user interface of the CT tool presented in this study; right side: enlarged toolbox - an extraneous element not to be used in completing the task (in red color).

The tool was built on top of Blockly.js framework and its user interface is composed of the following elements: the toolbox (the surface with blocks available for use in the

current task), the working area (the surface on which students provide the solution), control button for starting the current task evaluation process, control button for deleting all blocks on the working area and the control button for displaying or hiding the current CT task text (Figure 1, left-hand side, blue button).

Since the participants were young and English was not their mother tongue, the tool interfaces were developed in Croatian language. Each student task attempt as well as the sequence of steps undertaken in solving a task were recorded (logged) (i.e. types of blocks she used, how did she connect them, when did she use them etc).

4. USING THE COMPUTATIONAL THINKING TOOL

The CT tool designed as part of this study is typically used by engaging in these steps: (1) the identification of suitable computational primitives from the toolbox, (2) placement and sequencing of the primitives onto the working area, (3) starting the solution evaluation via the control button, (4) examining the evaluation (visualization or animation) provided by the system, (5) modifying the primitives choice via the control buttons and (6) using the control buttons to open the next task (Figure 2).

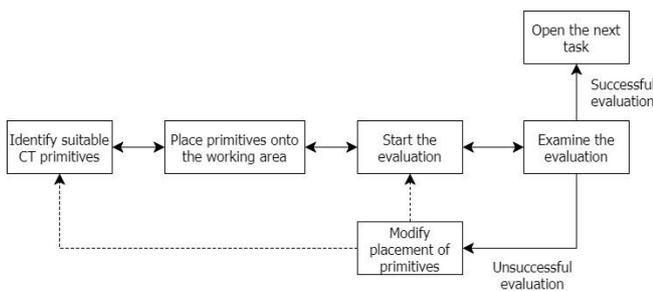


Figure 2. The CT tool usage process.

After a student identifies the suitable primitives in the step 1), and places blocks as part of the solution onto the working area in the step 2), and presses the control button to start the evaluation (step 3)), the tool automatically evaluates all student actions and data and provides feedback information about the current solution in form of an animation or visualization (step 4)) consisting of multiple steps to represent the chosen sequence of CT primitives.

In the *first task group* (see Table 1 for all task groups and the corresponding CT tasks), students were given sets of steps of certain well known algorithms, such as the recipe for making bread, as CT primitives, and were asked to place them in order. Some tasks had extraneous steps listed, which students needed to recognize and eliminate from their final solution (Figure 1, right-hand side). Once CT primitives are chosen and sequenced, the animation or visualization is run to represent the solution proposed by the student (Figure 3). It is to be noted that in the case of an incorrect step choice or sequencing done by the student, the displayed animation will reflect the incorrect choice (i.e. recipe/algorithm for making bread will intentionally be displayed in the wrong sequence and an indication to students will be given – right hand side of Figure 7).

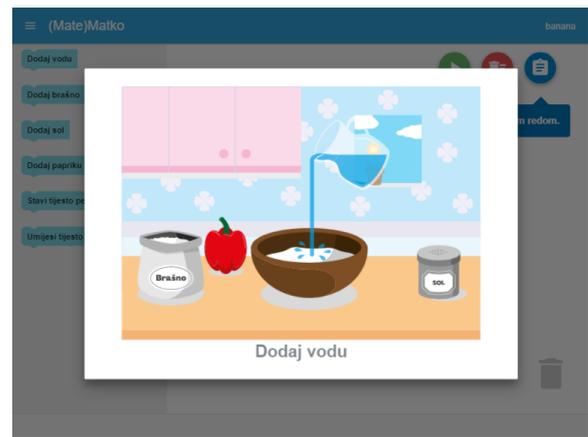


Figure 3. An animation displayed to a student following the choice of primitives and their placement and sequencing on the working area (task group 1 – recipe for making bread task).

Each task from the *second task group* had a dedicated animal well known to the students. To correctly solve a task, students needed to recognize which properties belong to an animal set in the task, for instance how many legs a frog has. In this task group, following a student primitives choice, animal properties are visualized step by step (Figure 4) in the central working area, prompting students to examine and reiterate their initial object property choice if needed.

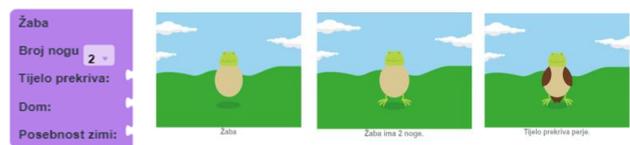


Figure 4. Choosing animal properties within an animal frog object representation (left hand side). A visualization displayed to a student following the choice of primitives on the working area (task group 2 – legs and feathers as incorrectly chosen properties of a frog).

To correctly complete a task from the *third task group*, students needed to define the path a bunny should use to get to the goal. In this task group, feedback provided to the students, after they submitted their task solution, consists of an animated bunny traversing the map according to the primitives sequence provided by the student (Figure 5). Some of the tasks from the third task group had multiple goals drawn, and the students needed to distinguish which goal should their bunny reach based on the supplied task text. The text specified whether the goal was the largest/smallest or left/right in relation to their bunny on the displayed map, which students needed to read, comprehend and apply in their solution.

The *fourth and fifth task groups* consisted of mathematics tasks where the students were given a start value and asked to supply the correct sequence of steps corresponding to adding or subtracting the pre-selected numbers to reach the correct end solution. Although in this task group the students needed to reach the correct solution by applying mathematical formulae, they were required to choose the right loop CT primitive to reach the final solution. The steps were selected by choosing the right blocks, where some of

the blocks were simple operations (e.g. “Create number 11”, “Add 5”), while the others included repetitions (e.g. “Repeat two times”). The fourth task group included the addition and subtraction of numbers from 1 to 10, while the fifth group included the addition and subtraction of numbers from 10 to 19. In both task groups, animated visual representations of mathematical equation elements related to each solution step of the calculation are shown to students (Figure 6). The right hand side the Figure 6 shows one of the expressions being calculated (1+1), currently chosen number to be added (+1 in blue color), and the end result (=2 in purple color).

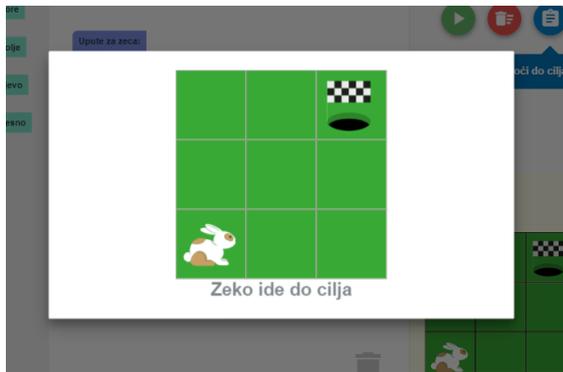


Figure 5. An animation displayed to a student following the choice of primitives and their placement and sequencing on the working area (task group 3 – guiding a bunny towards the goal).

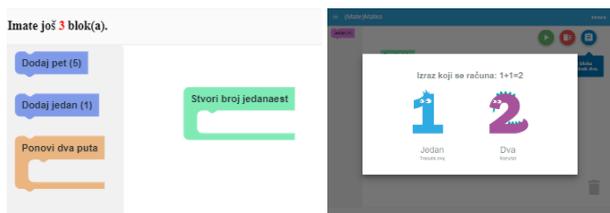


Figure 6. The animation displayed to a student following the choice of primitives and their placement and sequencing on the working area (task group 4 and 5 – assembling a formula and providing its result).

In all task groups, after a solution has been visualized or animated to a student, a message about its correctness is shown to the student completing the task (Figure 7). If the task was solved correctly, student should continue with the next task should one be available. If the provided solution is incorrect, student should choose to complete the task again.

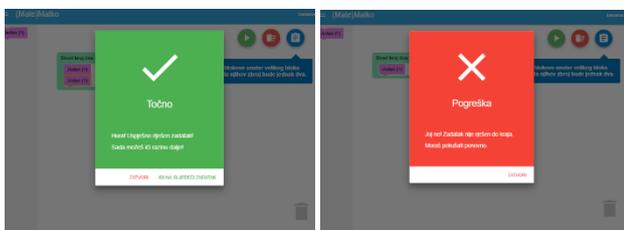


Figure 7. Feedback message after evaluating solution proposed by a student (left hand side – a correctly completed task, right hand side – an incorrectly completed task).

5. ANALYSIS AND RESULTS

The CT tool presented in this paper allowed for detailed data collection of students’ usage and performance data for each CT task group and its corresponding subjects and tasks. The collected data for each student for all five CT task groups included (1) the time students needed to complete a task, (2) the total number of attempts for a task, (3) the number of successful attempts for a task and (4) the number of unsuccessful attempts for a task (Table 2).

Table 2. Overall statistics of the collected data for all tasks across all five CT task groups.

	Mean	SD
Single task completion time (per student) (seconds)	62.49	19.14
Task completions (attempts per student)	78	25
Successful task completions (attempts per student)	35	12
Unsuccessful task completions (attempts per student)	43	18

The analysis indicates that, on average, students engaged in a single task for about one minute and, on average completed more than 70 tasks over the course of all 5 lessons and, on average, had slightly more unsuccessful attempts than the successful ones, with the mean success rate being $M=0.46$ ($SD=0.11$, $N=23$). The total time students spent on dealing with repeated solutions attempts amounted to only around 15% of the overall task completion time.

Table 3 indicates students spent most of time on engaging in the Science subject and on completing science tasks. What is more, in the Science subject students had the most successful and unsuccessful per-task attempts, with the unsuccessful attempts reaching high value of 2.48 attempts per task. These figures come with large standard deviation (SD) values indicating large between-student differences.

On average students spent as much as 243 seconds on completing CT tasks with object properties and only 68 seconds on completing problem tasks (the time includes all attempts in completing a single task). The difference is notable in per-task completion time as well, which is around two times larger in favor of object properties. The analysis indicates that the CT concept of object properties had the largest values of successful and unsuccessful task attempts, with substantial SD observed. SDs both in the case of object properties and problem tasks were high.

When solving tasks related to the sequence and object properties CT concepts a high number of successful and unsuccessful attempts was exhibited by the students. In the case of recognition and removal of unnecessary steps CT concept, SD for the total completion time was extremely high, indicating large differences in student performance.

Table 3. The analysis of total completion time, per task time and the number of successful and unsuccessful attempts for school subjects and specific CT concepts (time is indicated in seconds).

	Total completion time (mean)	Total completion time (SD)	Per task completion time (mean)	Per task completion time (SD)	Succ. attempts (mean)	Succ.atte mpts (SD)	Unsucc. attempts (mean)	Unsucc. attempts (SD)
Croatian language	143.67	49.46	62.76	23.31	0.99	0.58	0.97	0.43
Mathematics	94.72	49.59	41.79	21.93	0.85	0.38	0.76	0.49
Science	208.19	88.50	65.18	33.49	1.19	0.54	2.48	1.75
Sequence	167.42	78.12	68.34	38.35	1.11	0.97	1.01	0.68
Algorithm	161.60	130.02	70.27	50.11	0.75	0.44	1.00	0.92
Recogn. and rem. of unrec. steps	137.46	154.92	53.63	48.24	0.87	0.63	1.17	1.70
Object properties	242.57	103.09	69.79	36.93	1.36	0.72	3.25	2.64
Problem task	68.28	62.00	33.76	23.23	0.85	0.35	0.64	0.58
Problem task with loops	166.48	109.53	63.61	46.86	0.86	0.54	1.09	1.00

In the remainder of the analysis, the time and the number of successful and unsuccessful attempts were correlated with the students' skills in mathematics and the prior reading difficulty variable to check how student academic performance relates to their CT task performance. The teacher was asked to assess students' knowledge of mathematics on a scale from 1 to 10 prior to the study onset. The mean value for all students' mathematics knowledge was $M=7.91$ ($SD=1.62$). Reading difficulty was indicated by the teacher in 6 out of 23 students.

The correlation analysis, presented in Table 4, shows that students with good prior mathematics skills on general take more time and have more successful attempts in solving language CT tasks. On the other hand, students with identified reading difficulty are less successful in language task. For the Science subject, mathematics skills contribute to shorter completion time, while the students with reading difficulty spent more time on Science tasks. Interestingly, there were no correlations of mathematics and reading skills with the students' Mathematics subject performance.

In regards to the CT concepts, students with better mathematics skills are in general more successful in algorithms and have fewer unsuccessful attempts in solving problems tasks CT concept tasks. They spend less time on

object properties and problem tasks. Students with reading difficulty had less successful attempts in algorithm tasks and take more time in completing object properties and problem tasks. Surprisingly, students with reading difficulty were more successful and took less time in problem tasks with loops, however this result warrants for more research.

6. DISCUSSION

Early primary school children participating in this study were very enthusiastic about solving computational thinking tasks and were able to learn how to use the CT tool almost instantly. Researchers observed that children love the narrative of a robot named Matko which was the main avatar in the utilized CT tool. When solving the CT tasks, students on average failed slightly more times than they were successful, but they completed the repeated attempts very quickly and helped each other in the process.

One of the key findings of this study is the identified relationship between students' mathematics and language skills in completing CT tasks. Young primary school children are just beginning their schooling and some of them still lack reading and mathematics skills which is found to affect their performance in the Croatian language and Science CT tasks.

Table 4. The correlation of school subjects and CT concepts with students' mathematics skills and reading difficulty.

Subject/CT concept	Mathematics skills	Reading difficulty
Croatian language	Avg. completion time $r=0.419^*$	Num. of successful attempts $r=-0.453^*$
	Num. of successful attempts $r=0.470^*$	
Science	Tot. completion time $r=-0.521^*$	Tot. completion time $r=-0.438^*$
Algorithms	Num. of successful attempts $r=0.478^*$	Num. of successful attempts $r=-0.503^*$
Object properties	Avg. completion time $r=-0.501^*$	Tot. completion time $r=-0.564^{**}$
	Tot. completion time $r=-0.638^{**}$	
Problem tasks	Avg. completion time $r=-0.461^*$	Avg. completion time $r=0.435^*$
	Tot. completion time $r=-0.547^{**}$	
	Num. of unsuccessful attempts $r=-0.434^*$	
Problem tasks with loops	Num. of successful attempts $r=0.563^{**}$	Tot. completion time $r=-0.446^*$
		Num. of successful attempts $r=-0.463^*$

* $p<0.05$, ** $p<0.001$

Young primary school children need to have adequate reading skills to interact with CT primitives used in courses other than Mathematics more successfully (this was especially the case with the Science subject). Classroom observations during the in-class CT activities show students had less issues with using the tool interfaces and understanding how to manipulate the primitives than with understanding and applying some of the used vocabulary (i.e. the words up/down/left/right). This extended the students' time in completing the CT tasks and indicates that more interactive forms of content representation such as the puzzles or board games might be suitable for young students. Nevertheless, the identified gap proved as a great opportunity for teacher or peer facilitation of student work, whereby students get engaged in the task completion process even more.

In this study prior mathematics skills are identified as an important prerequisite in young children's successful and timely CT task completion across all subjects. With almost all CT concepts (algorithms, object properties and problem tasks) better mathematics skill was related to more success in solving CT tasks, and usually in less time. Such findings indicate conceptual similarities between the areas of mathematics and CT skills and warrant an adequate curriculum alignment of the Mathematics subject and other subjects using CT.

Students were fast and successful in completing mathematics CT tasks, with the exception of mathematics problems with the double loop CT concept, which proved to be fairly complex for young primary school learners. The CT concept of object properties caused misconceptions leading to most time spent and the largest proportion of successful and unsuccessful attempts. Classroom observations indicate students often reverted to trial and error method of completing such tasks since they found them both conceptually difficult and challenging to read and process.

The presented findings consistently indicate large differences between young students in solving CT tasks. It seems some students still struggle with basic language knowledge and basic mathematics, even though they are in the second semester of the 1st grade. On the other hand, some students are already doing well in language and mathematics, or were exposed to computer games and other computational tools at home or in kindergarten, leading to better CT task success. Such differences were alleviated with a small amount of scaffolding from the teachers or classmates, with all students being able to catch-up, excel and have inspiring aha-moments connecting previously unknown task elements.

7. CONCLUSIONS

The paper presented a study on computational thinking use in early primary school. The findings indicate reading and mathematics skills play an important role in students' CT task performance. Mathematics skills are of great importance and they help students in completing CT tasks in subjects such as language and science. Reading difficulty

presents an issue when young children are to process more complex CT tasks, warranting for contingency in terms of teacher and peer scaffolding. Large variation in students' performance seeks for an approach in which CT tasks of varied difficulty are used.

8. ACKNOWLEDGMENT

This work has been fully supported by Croatian Science Foundation under the project UIP-2013-11-7908. The authors would like to thank the staff of Primary School Tin Ujevic, especially Ines Falak, for their partnership in the realization of the study presented in this paper.

9. REFERENCES

- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Annual American Educational Research Association Meeting, Vancouver, BC, Canada, 1–25. <http://doi.org/10.1.1.296.6602>
- Lu, J. J., & Fletcher, G. H. L. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, 41(1), 260. <http://doi.org/10.1145/1539024.1508959>
- Papert, S. (1983). *Mindstorms: Children, computers and powerful ideas*. New Ideas in Psychology (Vol. 1). [http://doi.org/10.1016/0732-118X\(83\)90034-X](http://doi.org/10.1016/0732-118X(83)90034-X)
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... Kafai, Y. (2009). *Scratch: Programming for All*. *Communications of the ACM*, 52, 60–67. <http://doi.org/10.1145/1592761.1592779>
- 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. <http://doi.org/10.1016/j.chb.2016.08.047>
- Ruthmann, A., Heines, J., & Greher, G. (2010). Teaching computational thinking through musical live coding in scratch. *SIGCSE '10 Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, 351–355. <http://doi.org/10.1145/1734263.1734384>
- Wilson, A., & Moffat, D. C. (2010). Evaluating Scratch to introduce younger schoolchildren to programming. *Proceedings of the 22nd Annual Workshop of the Psychology of Programming Interest Group*, 64–75.
- Wing, J. M. (2006). Computational Thinking. *Communications of the Association for Computing Machinery (ACM)*, 49, 33–35. <http://doi.org/https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>
- Wing, J. M. (2010). Computational Thinking: What and Why? The link - The Magazine of the Varnegie Mellon University School of Computer Science, (March 2006), 1–6. Retrieved from <http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>

Computational Thinking and Unplugged Activities in K-12

不插电的计算思维教学活动在高中课堂教学中的应用

——以《二进制卡牌》课程为例

杨冰清*, 张进宝

北京师范大学教育学部

bingqingbnu@mail.bnu.edu.cn, zhangjb@bnu.edu.cn

摘要

计算思维在 21 世纪备受关注, 培养计算思维能力的教学活动设计与开发是教育学者关注的热点问题。不插电的计算思维教学活动主要在 K12 低年级阶段开展, 缺乏在高年级实施不插电教学活动的研究。经分析发现, 高中阶段计算机科学概念类知识的学习, 传统的教学方法效率相对较低, 并且容易造成学生对计算思维的误解, 而任务驱动模式教学不仅能够避免这些发生, 还能促进提升学习成效。本研究以基于计算思维的任务驱动教学模式为指导, 在高中进行实践。结果表明, 相对于传统教学, 不插电教学活动更能激起学习兴趣, 提升学生的计算思维能力。除此之外, 学生在活动中体验乐趣, 并表明非常乐意在未来的学习中参与不插电的计算思维教学活动课程。

关键字

计算思维; 不插电活动; 课堂教学; 任务驱动

1. 研究背景

随着科学研究的发展和进步, 计算机作为重要工具, 已经成为现在和未来学习中必不可少的工具, 计算机与人日常的交互使计算机的操作方法开始逐渐影响并引领人的思维。计算思维能力被称为 21 世纪全球人类必须具有的基本素养之一(Bocconi, S. & A. Chiocciariello, et al, 2016), 计算思维的培养在 K12 教育领域受到了愈发多的关注。人们多过的关注计算机工具的使用而忽视了其中蕴含的计算思维, 计算思维真正受到广泛关注始于 2006 年, 周以真将计算思维定义为是运用计算机科学的基础概念进行问题求解、系统设计以及人类行为理解等涵盖计算机科学之广度的一系列思维活动(Wing, 2006), 是所有人必须掌握的思维技能。周以真的观点引发了学界对“计算思维”的热烈探讨, 极大地推动了计算思维的理论 and 实践研究领域的发展。

在 K12 阶段的相关研究中, 已有足够多的研究关注环境的设置, 工具的开发以及课程设计, 现在更加需要的是针对真实教学环境的实践研究。开展计算思维教学多以计算机科学, 信息学知识为基点, 以计算机作为辅助工具进行编程或者以不插电教学活动两种形式展开。Matthias 等人指出, 相对不插电来说, 编程教育是困难的, 因为它要求学生掌握和理解抽象(Matthias H. A., 2018), 另外, 由于资源有限, 中国仅有部分地区能够为 K12 阶段的学生提供足够多的资源实现计算机支持的编程教育, 而不插电的教学可以随时随地进行, 不受资源限制。在 K12 阶段不插电的教学活动更能提升

学生兴趣, 加强学习效果。值得一提的是, 虽然计算思维多与编程联系起来, 但理解编程更加重要, 正如周以真教授所说, “像计算机科学家一样思考比有计算机编程能力重要的多。”不止编程能够锻炼计算思维, 还有更多的方式。研究表明, 非编程的方法是实用的而且简单易操作, 不插电的教学活动不仅让学生理解了计算机科学的概念, 更重要的是提升了学生的思维能力(王芬与何聚厚, 2017), 而非编程的方法更能吸引学生进行计算思维的学习, 而学生的收获也超出计算机编程学习。

2. 文献综述

2.1 计算思维

计算思维的相关研究可以追溯到 20 世纪 50 年代, 麻省理工学院终身教授西蒙·派珀特(Seymour Papert)在《认知器演算法》中提出了“计算思维”。由于当时计算机教育的发展有限, 这一概念没有引起相关研究领域的广泛关注。而如今, 由于计算思维与计算机科学, 信息学, 数字素养密切相关, 计算思维能力的培养有助于学生解决实际问题, 教育工作者越来越关注计算思维并将其融入到课堂教学中。

计算思维引入到课堂教学给学生的学习带来了积极影响(Yihua L, 1998)。培养计算思维能力的课程包括计算机科学, 信息技术, 计算课程等。国内以计算机科学课程和信息技术课程为主, 低年级阶段的信息技术课程关注技能的培养, 高年级阶段聚焦计算思维能力的培养。相关课程在 K12 阶段有不同的着重点, 小学阶段注重实践和信息技术的应用, K7-K9 阶段注重信息技术的应用及如何利用计算思维解决问题, 高中阶段注重通过探索计算机科学的概念来使学生理解计算思维并培养计算思维能力。高中阶段计算机科学概念的学习, 计算机科学家和教师有不同意见。Chiu-fan 等人针对中学生计算机科学概念的学习态度做了调研, 教师相对于计算机科学家来说, 更加赞同中学生接触计算机科学概念的学习, 因为教师知道如何将复杂的概念简单化并教授给学生, 计算机科学家们则显得较为保守, 他们认为计算机科学概念应该在大学阶段学习(Hu, C., Wu, C., & Wang, A., 2017)。实际上, 计算机科学的学习不仅限于大学, 将复杂的计算机科学概念简化, 融入中学课堂是非常必要的。

在中学计算机科学概念类学习课程中最大的问题是, 传统教学方式针对概念类知识的学习, 主张讲解和记忆, 忽视了思维能力的培养, 枯燥的概念学习和强硬

的背诵记忆造成学生不仅对课程反感，对计算思维也产生了误解。Debroah 等人的研究表明，高中计算机科学课程中，运用适当的教学策略，能够使中学生有效地学习计算机科学概念，改变对计算思维的误解，从多方面提升学生的计算思维能力，让计算思维更加实用 (Fields, D. A., Debra, L. U. I., & Kafai, Y. B. 2017)。虽然，K12 阶段的课程培养有不同的着重点，但不插电的计算机科学活动适合不同学段和任何国家的学生 (Bell, T., Alexander, J., Freeman, I., & Grimley, M. 2009)，不插电的教学活动形式非常适合计算机科学概念的学习。这种形式的教学受到学生的青睐，它不仅能消除编程障碍，使学生直接通过活动学习计算机科学和信息学的基本概念和原理；再者，不插电的计算思维教学使学生不局限于电脑前，有充分的活动空间；另外，不插电的教学方式使学生沉浸在活动情境中，在具体的实践中培养学生解决问题的能力。

2.2 不插电的计算思维

不插电的计算机科学教学活动由新西兰的 Tim Bell, Lan H. Written 和 Mike Fellowes 三位老师发起，指的是通过有趣的游戏、谜题来让学生理解计算机科学的概念，提高学生对计算机科学的兴趣，达到不用打开计算机就可以很好的理解计算机科学的概念。过去的几年，由 Tim 研究团队发起的不插电的计算机科学项目受到国际多个国家的吸收和使用，并且受到美国的关注并加入了 ACM 义务教育 K12 阶段的课程。将计算思维融入到 K12 教学中的一个途径是任务的设计与描述，计算思维教学的任务应该区别于其他教学任务 (Barr, V., & Stephenson, C. 2011)。任务驱动的计算思维教学模式具有可行性和高效性 (吕会庆、张巍, 2012)，牟琴等人提出了基于计算思维的任务驱动教学模式 (TDMCT) 并进行实践，结果表明这种教学模式在培养学习者的自我建构和创新思维上有较大的进步，不仅能够提高学习效率，而且可以培养学习者的计算思维能力。

基于以上分析，本研究以二进制卡牌课程为例，以不插电的形式开展教学。并提出假设：假设 1，不插电的计算思维教学活动相对于传统教学方式，能够帮助学生学习理解计算机科学的概念并提升学习兴趣，使概念的学习不止停留在抽象阶段而变得更加有形。假设 2，任务驱动的学习在高中阶段是可行的，能够提升学生的高阶计算思维能力。

3. 研究设计

3.1 研究方法和工具

本研究是实证研究，主要用课堂观察法和问卷调查法。课堂观察的目的指向学生学习的改善，在课堂活动中观察学生学习状态。问卷法是通过由一系列问题构成的调查表收集资料以测量人的行为和态度的心理学基本研究方法之一，利用问卷调研收集学生学习反馈数据，采用李克特五维量表 (Likert Scale)，主要从课程体验、课程满意度，未来学习偏好三个维度对学生进行调研，经检验问卷量表 α 系数为 0.71，符合信效度要求。复杂的任务由多个学生一同完成最为合适，将学生分小组进行课堂活动，主要用到二进制卡牌和小组任务表。卡牌作为在课堂活动中为小组完成十进制和二进

制的相互转化的任务提供辅助和支撑，每小组成员一副二进制卡牌。

3.2 不插电的教学活动设计

二进制卡牌课程已经有许多实践，本次课程基于计算思维的任务驱动教学模式对进行教学设计，将教师和学习者之间通过任务连接，教学过程以任务为主线，教师作为引导，学生为主体运用计算思维方法来完成任务。二进制算法和不同进制之间转换的算法是大学的内容，以抽象化的概念在计算机科学课程中体现，但本课程以二进制卡牌作为辅助工具，将抽象的算法简化后，使其能够在高中展开教学。课程教学设计，主要包括课堂引入，小组任务，总结拓展三个阶段。小组任务包含三个任务，小组活动任务表内容如表 1 所示。

表 1 小组任务表

初级任务	任务 1	十进制转化为二进制
	任务 2	二进制转化为十进制
高级任务	任务 3	符号表示的二进制转化为十进制数值

任务 1 和任务 2 在情境引入之后，其目的是让学生利用工具，实现十进制和二进制的相互转化。前两个任务是初级任务，能够调动学生学习的主动性，利用卡牌完成任务的同时，能对二进制算法与十进制算法的相互转换有基本了解。接下来是高级任务，任务 3。在前两个任务的基础上，完成此任务，其内容是识别符号表示的二进制并进行转换，目的是为了检验所学，并提升学生的高阶计算思维能力，任务三需要学生能够通过符号抽象进行转换，激发学生思考，提升计算思维中的抽象思维能力，完成所有任务后，进入课堂总结阶段，由教师带领学生一起探讨和归纳不同进制数的共同规律。

3.3 研究对象及过程

课程的参与者是某高中二年级的学生，共 52 人参与课程 (男 34 女 18)。课程进行之前对学生参与课程进行情意调研，80% 以上的学生在此次课程前参与过有趣的教学活动。个别同学在高中以前，曾多次参与有趣的教学活动。但是由于高中课程紧张，同学们基本没再参与有趣的教学活动，对于本次课程学生非常期待参与。将班级的 52 名同学分成 24 个小组，2-3 人一组。每组成员之间相互熟悉，这更利于合作探究学习的开展，以及小组任务的完成。

高中阶段学生很大程度上对计算机科学有一定误解，认为学计算机就是编程或者学习枯燥的概念。实际上，不插电的计算思维教学活动能够让计算机科学的原理和概念以有趣的活动的方式展现，不插电的课堂活动旨在给学生带来好的学习体验，引起学生兴趣，在一定程度上纠正其对计算机科学不正确的认识。高中阶段学生已经有整体的逻辑思维能力的，有效的教学方法对培养计算思维能力的培养起着至关重要的作用 (丁玲, 2017)。另外，学生对于十进制非常熟悉，关于二进制，

所有同学都知道二进制计数法存在，对其具体算法并不了解，同学们对所学内容充满好奇并带着兴趣和好奇心进入课程。

4. 结果与讨论

课程以学生为中心开展，学生是学习的主体，所以课程中最主要的教学活动即小组合作完成任务的活动，小组活动期间有教师仅作为指导者参与，通过对问卷数据的分析，以及对两个假设的分析进行讨论。

首先通过问卷数据收集进行分析。在课程体验方面，我们发现与 Nicole 等人的研究类似，计算思维的学习领域男生和女生表现出不同的自信等级 (Hutchins, N. M., Zhang, N., & Biswas, G. 2017)。虽然在小组合作的过程中，无论是男生还是女生，都能够互帮互助一起完成任务，有良好的交互。但在学生的课程体验方面，男生在活动中更加自信，如表 2 所示。

表 2 男女生自信等级 (%)

	高	较高	正常	低
男生	14.71	76.47	5.88	2.49
女生	11.11	50.09	38.89	0

在课程满意度方面，80% 以上的学生对课程非常满意，在未来学习偏好方面，超过 84% 的学生在未来的学习中愿意再次参与课程，有同学说：“高中紧张的学习期间，有一些新颖的课堂活动是很好的，对于这次课程，既可以通过活动学习知识，又丰富‘学业生活’，可谓两全其美，希望老师常来。”有学生说：“我认为这样的课程应多讲一些，既可以锻炼与同学的合作能力，又可以学习新的知识，我十分赞同多开展几次。”班级原授课教师也惊叹学生从未如此认真地参与到教学活动中。

再者，对于假设 1。通过课堂观察和对比，不插电的教学方式，确实更能够提升学生的学习兴趣与参与度，在传统教学方式讲授二进制算法及其与十进制相互转化的课程中，由于课程内容与高考无关，50% 以上的学生学习不专注或者做其他学科的任务，学生对枯燥的概念和复杂的算法并不感兴趣，甚至认为是课程内容加重了学习负担。但在本次课程中，所有同学都参与到教学活动中，并且认真主动完成任务并对其算法进行小组探究，对二进制的理解不仅是抽象的概念，而转化成了可用的算法工具，在完成的任务后寻找不同进制的算法规律，进行归纳演绎，所有同学都能归纳出不同进制数值算法的一般规律。实际上，学生不仅学会了二进制的概念，同时也提升了如抽象和归纳的高阶计算思维能力。

对于假设 2，任务驱动的学习在高中阶段是可行的，运用任务驱动的教学策略能够提升学生学习主动性，并提升学生的计算思维能力。针对课堂活动进行分析。从活动过程来看，活动开展之前，课堂引入生活的小问题成功地引起了学生的兴趣，调动学生的积极性，为下面小组任务的开展和探究二进制计数法的活动奠定了基础。小组成员进行合作，依次完成小组的三个任务。期间教师仅作为指导者，帮助学生解答疑惑，帮助学生解决问题，继续完成小组任务。从学习结果来看，教学活动三个任务的其正确率如下图所示。

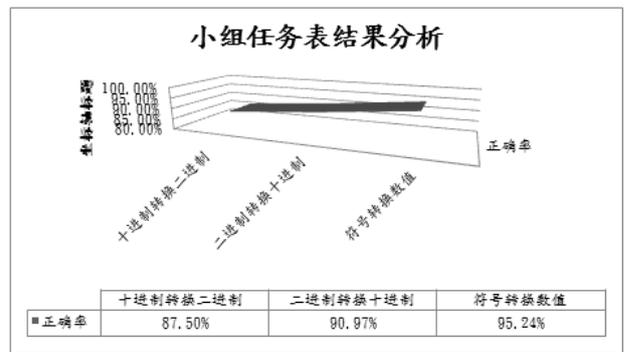


图 1 任务表结果

尽管任务难度逐渐增加，但三个任务的正确率几乎呈现直线增长。在二进制和十进制相互转化的初级任务中，第一个任务的小组任务结果正确率为 87.50%，在最后一个高级任务中，正确率以及达到 95%。小组的任务完成后，学生已经基本掌握利用工具将二进制和十进制进行转换，并且能够抽象理解二进制数。

小组任务结束后，进入学生提问阶段和课堂总结活动。经过课堂的总结和拓展，学生了解到计数法不止有二进制十进制，还有八进制，十六进制等等，并归纳出所有进制数的共同规律。在课后的反馈中，我们发现不插电的计算思维课程在高中阶段是可行的，不插电的教学活动能够培养学生的计算思维能力，更能够使学学生勤加思考，调动学习主动性。

5. 总结与展望

本次课程活动比较成功，无论是课程过程还是学生课后的反馈来看，高中学生对于不插电的计算思维教学活动非常感兴趣并且享受这种课程。不插电的计算思维教学活动以做中学 (Learning by Doing) 为宗旨，现今的数字化时代，学生更应该掌握在做中学 (Tan and Kim, 2015)，尤其是 K12 阶段的高年级学生，比起背诵概念，在实践中理解并运用概念更为重要。

对于本研究也存在一定的局限和不足，在高中的课堂中开在不插电的教学，班级人数较多，教师不能关注到每一位学生，在提供指导和帮助学生解决小组困难的问题上，有待提升。Halil 指出，对于计算思维的相关研究中，多关注问题解决，技术和思维方式，对学生本身的关注较少 (Haseski, H. I., Ilic, U., & Tugtekin, U., 2018)。对本研究来说，在关注个性化学习方面确实有所欠缺，在深入了解学生本身特征方面存在不足。未来的研究中，将缩减班级人数为 30 人以内，以便关注每一位学生，关注学生个性化，并对学生的学习特征进行分析，改进课程进行研究。

6. 参考文献

丁玲 (2017)。高中信息技术课程中学生计算思维的培养。教育，3，154。

吕会庆、张巍 (2012)。基于计算思维的计算机任务驱动教学模式。计算机教育，7，94-96。

牟琴、谭良和周雄峻 (2011)。基于计算思维的任务驱动式教学模式的研究。现代教育技术，21，44-49。

- 王芬、何聚厚 (2017)。不插电的计算机科学展计算思维的有效途径。《教育现代化》，20，45-47。
- 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.
- 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.
- Bocconi, S. & A. Chiocciariello, et al. (2016). Exploring the Field of Computational Thinking as a 21st Century Skill. *8th International Conference on Education and New Learning Technologies*
- Fields, D. A., Debra, L. U. I., & Kafai, Y. B. (2017). Teaching computational thinking with electronic textiles: High school teachers' contextualizing strategies in exploring computer science. *Siu-cheung KONG The Education University of Hong Kong*, 67-72.
- Hu,C.,Wu,C.,&Wang,A (2017). How computer scientists and computing teachers think different in the concepts to be included in a secondary school computing curriculum. *Siu-cheung KONG The Education University of Hong Kong, Hong Kong*, 50-54.
- Haseski, H. I., Ilic, U., & Tugtekin, U. (2018). Defining a New 21st Century Skill-Computational Thinking: Concepts and Trends. *International Education Studies*, 11(4), 29.
- Hutchins, N. M., Zhang, N., & Biswas, G. (2017). The Role Gender Differences in Computational Thinking Confidence Levels Plays in STEM Applications. *Siu-cheung KONG The Education University of Hong Kong, Hong Kong*, 34-38.
- Matthias H Andrea Adamoli, S.(2017). The program is the system. *Proceedings of the 17th Koli Calling Conference on Computing Education Research 11 (4):29-42*
- Tan, L. & B. Kim (2015). Learning by Doing in the Digital Media Age, *Springer Singapore*.
- Wing, J. M. (2006). "Computational thinking." *ACM Sigcse Bulletin* 49 (3): 3-3.
- Wismath, S. L. & D. Orr (2013). Collaborative Learning in Problem Solving: A Case Study in Metacognitive Learning. *Canadian Journal for the Scholarship of Teaching & Learning* 6 (2): 23-32.
- Yihua L. (1998). Exploration on Database Teaching Based on Computational Thinking. *Boletín Técnico*, 55(17), 363-370.

Design A Computational Thinking Board Game Based on Programming Elements

Sheng-yi WU^{1*}, Jia-cen FANG¹, Shu-mei LIAN²

¹Department of Science Communication, National Pingtung University, Taiwan

²ChungCheng Junior High School, PingTung, Taiwan

digschool@gmail.com, pray9821@gmail.com, dido8877@gmail.com

ABSTRACT

Computational thinking (CT) in education has been considered significant to future national competitiveness and development. Programming is seen as the most direct way to develop the CT skills. However, it is also argued that young starters are easily frustrated and discouraged when they have difficulties in programming syntax and concepts. Thus, a programming course with adaptive visualization or game-based learning is considered a better solution to encourage higher-order thinking that benefits young students. The unplug educational board game features low selling price, intensive interaction among players, and is able to play without any computing equipment. It is therefore attracting growing attention from schools and teachers. In this study, we designed a CT board game, *Interstellar Explorer*, on the theme of space. Users play *Interstellar Explorer* by controlling on-screen spaceship and defending with obstacles to find the most residential planet in an animated outer space. *Interstellar Explorer* challenges players to design and implement strategies to carefully control the movement of spaceship and successfully build a defense. This study contributes to developing player's logical thinking and problem-solving ability as well as inspiring their imagination and creativity.

KEYWORDS

computational thinking, programming, coding, board game, interstellar explorer

1. BACKGROUND

Computational thinking (CT) in education has been considered significant to future national competitiveness and development ever since the notion of CT was reintroduced by Jeanette Wing from Carnegie Mellon University in 2006. In some countries such as America, England, Australia, and Estonia have started including computing in the school curriculum and teacher training.

CT uses basic concepts of computing and information science to solve problems, design systems and understand human behavior (Wing, 2006). Along with reading, writing and arithmetic, CT is a requirement of a part of core knowledge. CT consists of skills like induction, embedding, transition, and simulation, which help to solve complicated problems in the way we are familiar with.

2. MOTIVATION and PURPOSE

The change of trend in education and global realization to the importance of CT has identified the significance of developing CT skills at a young age. In other words, programming is seen as the most direct way to develop the

CT skills. (Buitrago Flórez, Casallas, Hernández, Reyes, Restrepo, Danies, 2017). However it is also argued that young starters are easily frustrated and discouraged when they have difficulties in programming syntax and concepts (Costelloe, 2004 & Powers, Ecott & Hirshfield(2007). Furthermore, traditional programming can be boring to young students mostly due to its requirement of various syntax inputs (Mannila, Peltomäki & Salakoski, 2006). Thus, a programming course with adaptive visualization or game-based learning is considered a better solution to encourage higher-order thinking that benefits young students (Brusilovsky & Spring, 2004).

At present on the market, teaching materials to develop CT skills can be generally divided into three kinds of design: 1) blocks-based visualization, like code.org and Scratch; 2) real-robot control, like mBot and Dash & dot; 3) unplug educational board game with cards, like Robot turtles, King of Pirate, Doggy code, Code master, Robot Wars Coding Board Game. Each of the designs has its advantages and limits. The unplug educational board game features low selling price, intensive interaction among players, and playing without any computing equipment. It is therefore attracting growing attention from schools and teachers.

The five board games mentioned above are designed in accordance with programming elements. Yet, the elements are incompletely considered due to age setting and game mechanism. Thus, we try to design a CT board game based on programming elements, allowing players at any age to develop and practice CT skills.

3. DESIGN OF CT BOARD GAME AND ANALYSIS OF PROGRAMMING ELEMENTS

We design a CT board game, *Interstellar Explorer*, on the theme of space. Users play *Interstellar Explorer* by controlling on-screen spaceship and defending with obstacles to find the most residential planet in an animated outer space (see Figure 1).

Designed for players aged 8+, *Interstellar Explorer* challenges players to design and implement strategies to carefully control the movement of spaceship and successfully build a defense. This contributes to developing player's logical thinking and problem-solving ability as well as inspiring their imagination and creativity.

We create a set of cards to use in the game. Players place these cards in linear arrangement in the way similar to visual programming language learning. Players are also allowed to create his/her own conditional environment and implement

rules with blank cards where they can define clearly the condition and rules. Interstellar Explorer provides a game-based learning environment to teach basic programming and CT skills including sequences, events, loops, conditional, parallelism, names, operators, and data (Brennan & Resnick, 2012) (see Table 1).



Figure 1. Description of Interstellar Explorer

Table 1. Programming Concepts

Concept	Gameplay Instruction
sequences	Starting starship
events	adding meteorite, clearing meteorite, pause card, observing planets with telescope, beam card, deflector shield card, destroying meteorite in front, changing character, calling for character's skill, multifunction card, preference card
loops	flying card effect X n
conditionals	condition card 1-5
parallelism	controlling the opponent's ship
names	creating function card, calling for function card, blank condition card, blank implement card
operators	meteorite explosion, alien attack, magic power recorder
data	magic power recorder, supplies card

4. CONCLUSION

Promoting CT skills in education has become a global trend. Introducing CT concepts to young learners is even a significant step to develop problem-solving ability and logical thinking at a young age. Thus, we design a CT board game, Interstellar Explorer, based on programming concepts to help young learners in learning CT concepts and skills. In the future, we will carry out a study to explore the effectiveness of this CT board game, Interstellar Explorer, using the computational thinking scale and the behavior model.

5. REFERENCE

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada.

Brusilovsky, P., & Spring, M. (2004). Adaptive, engaging, and explanatory visualization in a C programming course. In L. Cantoni & C. McLoughlin (Eds.), *Proceedings of World Conference on Educational Media, Hypermedia, and Telecommunications 2004*, 1264-1271. Chesapeake: VA: AACE.

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*, 0034654317710096.

Costelloe, E. (2004) Teaching Programming The State of the Art. Department of Computing, Institute of Technology Tallaght, Dublin 24. *CRITE Technical Report*.

Falkner, K., Vivian, R., & Falkner, N. (2015, January). Teaching Computational Thinking in K-6: The CSER Digital Technologies MOOC. In *Proceedings of the 17th Australasian Computing Education Conference (ACE 2015)* (Vol. 27, p. 30).

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

Mannila, L., Peltomaki, M., & Salakoski, T. (2006). What about a simple language? Analyzing the difficulties in learning to program. *Computer science education*, 16(3), 211-227.

Powers, K., Ecott, S., & Hirshfield, L. (2007). Through the looking glass: teaching CS0 with Alice. *Proceedings of the 38th SIGCSE technical symposium on Computer science education SIGCSE '07*, 39 1, 213-217.

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

Computational Thinking and Coding Education in K-12

Analysis of Learner's Self-efficacy Using Coding Education Support System for Understanding Complex Problem-Solving Steps

In-seong JEON, Hyeon-jeong JEONG, Ki-sang SONG*

Korea National University of Education, Korea

wf212@naver.com, dolimofe@naver.com, kssong@knue.ac.kr

ABSTRACT

In this study, Levenshtein distance algorithm-based coding education support system which shows learners' progress in real time was developed in order to observe how learners would solve complex problems in the coding environment and a pilot test was conducted on elementary school students. When the teacher used the developed system to teach students, there was a statistically significant difference in *integrated regulation, external regulation, and introjected regulation* among the sub-factors of learning motivation compared to the conventional classes. Among the sub-factors of self-efficacy, *efficacy dimension* showed statistically significant difference.

KEYWORDS

Coding Education, Software education, Learning Motivation, Self-efficacy

1. INTRODUCTION

The core of Computational Thinking(CT) is to break down a complex problem into familiar and easier sub-problems (problem decomposition), solve the problems by applying algorithm, review how such problems can be transferred to similar problems, and decide whether to use computers to solve them better (Yadav, et al., 2016). In other words, Computational Thinking is the ability of computer scientists to solve problems by using computing technology as a way of thinking to solve problems, so CT plays an important role in solving complex problems.

Among the methods to graft CT into school education, there is automation. This element requires learners to connect to the computer. Learners can learn modeling and simulation using computing technology in this learning environment. These automation tools are developed in Scratch or in the local versions of Scratch so that programmers can learn CT skills easily through the coding process.

For effective programming learning, learners should have interest and internal motivation and they should be provided with the learning method considering the level of individual learners and interest (Katai, Z. & Toth, L., 2010). However, in reality, general programming education is conducted for a large number of learners in a classroom. For a learner to get an appropriate feedback from a teacher in this situation, adequate time allowance as well as the teacher's teaching ability is necessary (Han, K. W., Lee, E. K. & Lee, Y. J., 2010). Accordingly, related studies such as scoring according to the efficiency of the algorithm for a given problem or analyzing and evaluating the learners' learning are actively being performed (Jang W. Y. & Kin S. S., 2014).

Yet, automated tools to provide a convenient and flexible evaluation method are not available. Unlike the programming environment on the computer, it is very difficult to maximize the learning effect of software education in a limited environment where the learner's programming is analyzed and the task evaluation is performed manually (Kim M. H., 2007). To overcome these difficulties, studies on programming learning analysis system, where the teacher can analyze the learning status with an automatic and efficient method, give optimal feedback and easily evaluate tasks of the learners, should be made. In this study, a block programming education support system based on Levenshtein distance algorithm, which is designed to support the teaching of the teacher and promote the motivation of learners in the field of coding education, was developed and it was applied to 10 units of classes to analyze its effects on learning motivation and the self-efficacy of learners.

2. BACKGROUND

2.1. Algorithm Learning Analysis System

In the algorithm learning analysis system, when a learner creates a solution to a given problem using a programming language, the prepared source code is stored in the server, and the analysis program repeats the execution several times while confirming the number of times the server code is stored in the server at a predetermined time interval. Each time it is executed, data prepared in advance is entered to the program, and the result is compared with the value of the prepared answer data. The existing algorithm learning analysis system uses text-type programming language to compare strings, line, and compilation results of source code and answer code according to a certain algorithm, and provides error or score through message feedback.

These systems work well in environments that use text type programming languages and have the advantage of being able to feed back the results immediately. However, there is no learning analysis system developed for the purpose of performing such a function in the block type programming language environment which is currently used at the elementary and secondary school level. Therefore, the teacher should observe students roaming around the classroom and it is difficult for a teacher to identify what a learner thinks difficult.

2.2. Related Studies

Kim (Kim, M.-H., 2007) designed and implemented a web-based programming task evaluation system that allowed the teacher to automatically evaluate the performance of the program and easily check the style and plagiarism of the program with appropriate feedback (Kim M.-H., 2007).

Song (Song, J.-H., 2011) designed and developed an automatic scoring-based programming education system that could perform learner-centered self-directed learning by performing programming education more efficiently. Jang & Kim (Jang W. Y. & Kim S. S., 2014) developed a client-server based system by enforcing teaching-learning functions of the existing Online Judge style system and found its significant effect on programming learning. Jeong (Jeong, J.-K., 2010) developed a system that can be used for programming learning and evaluation of science high school students unlike Online Judge system which is used for evaluation in competitions.

From the analysis of previous studies, it was found that various automatic scoring systems for programming learning and evaluation had been developed. However, there have been no studies related to the development of a system that supports block programming languages for elementary and secondary school students and teachers up to now. Therefore, in this study, a pilot version of a system that performs block programming language learning analysis function was developed and the effect of the system on learning motivation and self-efficacy of learners through the classes where a teacher uses the system was verified.

3. SYSTEM DESIGN

In this pilot system, a learner can program using Entry, a block-type programming language, through a web server, and click the save button to analyze the source code in real time.

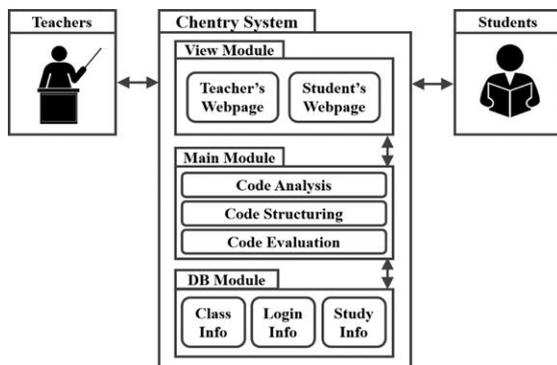


Figure 1. Developed pilot system [J.-H. Kim, et al. 2018]

The system applied to this study divides the web agent into a view agent to which the teacher and learners connect, a core agent that analyzes and structures the code, and a DB agent that stores class information, account information, and learning information. The teacher opens a class by entering the class name and the URL address of the answer code.

The learner accesses the system with his/her account, enters the URL address of the source code, and programs in Entry environment. The system checks whether the learner clicks the save button at a pre-determined time interval and saves accumulated achievements using the Levenshtein distance algorithm when the save button is clicked.

Levenshtein distance is a kind of edit distance technique that calculates the minimum number of edits such as deletions, insertions, or substitutions required when a string is converted into another string (Levenshtein, 1966).

$$\text{lev}(s_1, s_2) = \frac{\text{dist}(s_1, s_2)}{\max(|s_1|, |s_2|)}$$

In this study, the progress of students' learning was calculated through individual block group agreement and whole block group agreement.

$$\text{Individual block group agreement(\%)} = 100 \times \left(1 - \frac{\text{Levenstein block distance}}{\text{MAX(No. of Teacher block, No. of student block)}}\right)$$

$$\text{Whole block group agreement(\%)} = 100 \times \left(1 - \frac{\sum \text{Levenstein block distance}}{\sum \text{MAX(No. of Teacher block, No. of student block)}}\right)$$

In addition, if the block type is the same but only the variable or the block parameter is wrong, it is considered to be corrected 0.5 times rather than 1.

The teacher can check which block each learner has used, how the block of the learner has been changed for a certain time, and what the degree of final achievement is. The teacher can also provide corrective feedback to the student with low achievement level. The algorithm of the overall pilot system through the above agents is shown in Fig. 2.

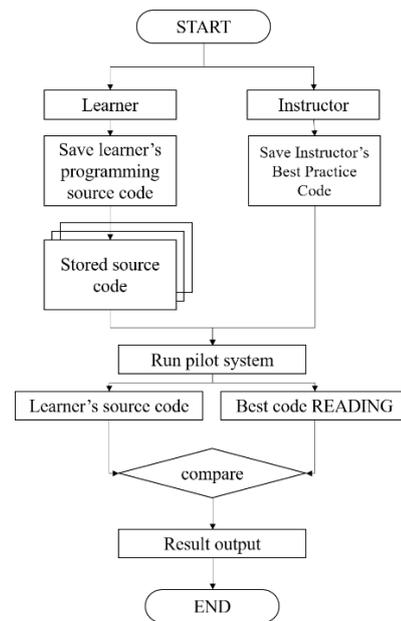


Figure 2. Algorithm of pilot system

4. Learners' Self-Efficacy in Developed Pilot System Utilization Education

The factors that affect programming education will vary. Wiedenbeck (2005) presented programming experience, self-efficacy, and knowledge organization as learning factors for non-professionals to successfully achieve programming learning. In this paper, we have proposed various learning examples for learners to gain programming experience and consider learning strategies to enhance learner's self-efficacy. To do this, we conducted a programming education with a constructive approach and a cognitive approach to designing teaching and learning. The learner was able to receive the feedback of the instructor and confirm the achievement degree of each block and construct the cognitive processing through the meta - cognition.

In order to verify the educational effects of the pilot system developed in this study, an experiment was administered on 40 grade 5 students in S elementary school in Gwangju, Korea. The experimental group and the control group received a total of 10 units of programming education using a block type programming language. In the case of the experimental group, the teacher identified the progress of the learners with the developed system, intervened appropriately, and let the learners compare their progress with the teacher's answer. For the control group, on the other hand, the traditional teaching method was used where the teacher roamed around the classroom to observe and advise the learners.

4.1. Program Design

The curriculum of 10 classes applied to the two groups was structured to learn computer science subjects such as sequence (SE1), repetition (R), selection (SE2), simple variable (V), list (L) and concatenation (C) included in the software education curriculum of elementary school in Korea. The summary of the curriculum is shown in Table 1.

Table 1. Curriculum for Programming education.

Time	Title	Elements
1	Basic programming language manipulation Use movement, shape block	C
2	Create simple block application program	C, R
3	Make 'bears meet bees' Use repetition with variables	C, R, V
4	Make 'Bee shot bear' Use selection Structures	C, R, SE2
5	Make 'Shark Avoid' Game	C, R, SE2
6	Develop games: Use Replication Blocks	C, R, SE2, V
7	Complete the game: Use lists and variables	C, R, L, V
8	Make a gift lottery program Use random number	C, R
9	Make 'Producer Speaker' Use Random Numbers, Arithmetic Expressions	C, R, L, V
10	Draw a polygon Use pen-block, basic functions	C, R, F

In the experimental group, learners were asked to check their learning achievement through the pilot system constantly during the 10 units of classes. The teacher checked the achievements of learners during the class and provided corrective feedback to the student who kept having low achievement level for quite a long time by analyzing the causes of low achievement. In the control group, the teacher gave a lecture just like in the conventional teaching method and gave feedback directly to the learners while roaming around the classroom.

4.2. Research Method

In order to measure the learning motivation of the students, the motivation test tool for youth developed by Lee M. H. and Jung T. Y. (2007) was modified for elementary school students. The learning motivation test tool was composed of 26 questions in total; specifically 5 items of Amotivation, 5 items of External Regulation, 5 items of Introjected Regulation, 5 items of Identified Regulation and 6 items of Integrated Regulation. Amotivation is the status wherein the desire to learn is not generated regardless of external stimuli, External Regulation is behavior control by external factors, Introjected Regulation is to act through influence of past experiences such as reward and punishment, Identified Regulation means that integrated control as external factor is changed into internal factor, and Integrated Regulation is the motivation to create and achieve something on its own. Table 2 shows the items and reliability of sub-factors of learning motivation.

Table 2. Reliability test of Learning Motivation.

Elements	Quantity	Item number	Reliability
Integrated Regulation	6	1,3,12, 17,20,23	.839
Amotivation	5	4,6,7, 9,26	.674
Introjected Regulation	5	11,16,18,19, 25	.750
External Regulation	5	5,13,14, 15,21	.818
Identified Regulation	5	2,8,10, 22,24	.775

In order to measure the self-efficacy of students in programming language, the self-efficacy test tool in computer programming language education environment developed by Kim (Kim, K. S., 2014) was modified. The self-efficacy test tool is composed of 30 questions in total; specifically 10 questions about language, 10 questions about Efficacy Factor and 10 questions about Efficacy Dimension. Language refers to the challenging spirit to develop a program by knowing the general structure of a programming language and the terminologies of variables, expressions, control statements, operators, arrays and functions and utilizing them. Efficacy Factor refers to whether they have direct or indirect experience with success or failure. Efficacy Dimension is the learner's perception on the level of difficulty of a given task, the willingness to challenge more difficult problems, and whether to generalize it. Table 3 shows the items and reliability of sub-factors of self-efficacy.

Table 3. Reliability test of Self-efficacy.

Elements	Quantity	Item number	Reliability
Language	10	1,2,3,4,5,6,7,8, 9,10	.875
Efficacy Factor	10	11,12,13,14,15, 16,17,18,19,20	.874

Efficacy Dimension	10	21,22,23,24,25,26,27,28,29,30	.835
--------------------	----	-------------------------------	------

4.3. Result

In order to verify the homogeneity of the experimental group and the control group, pre-test was administered to measure learning motivation and self-efficacy of two groups. As Table 4 shows, there was no significant difference, which confirms the homogeneity of the groups.

Table 4. Homogeneity test of group to measure for Learning Motivation & Self-Efficacy

Area	Group	N	M	SD	t	P
Learning Motivation	Experimental	20	73.90	7.873	-.421	.676
	Mental	19	75.16	10.658		
L1	Experimental	20	22.50	4.274	.432	.668
	Mental	19	21.79	5.903		
L2	Experimental	20	10.25	3.226	-.934	.357
	Mental	19	11.32	3.888		
L3	Experimental	20	12.20	3.122	-.959	.344
	Mental	19	13.26	3.784		
L4	Experimental	20	10.95	3.900	-.401	.691
	Mental	19	11.53	5.037		
L5	Experimental	20	18.00	3.356	.580	.565
	Mental	19	17.26	4.520		
Self-efficacy	Experimental	20	93.20	16.979	.165	.870
	Mental	19	92.11	23.949		
S1	Experimental	20	29.75	7.813	.024	.981
	Mental	19	29.68	9.310		
S2	Experimental	20	30.90	5.330	-.814	.421
	Mental	19	32.84	9.167		
S3	Experimental	20	32.55	6.362	1.343	.187
	Mental	19	29.58	7.434		

* p < .05

L1=Integrated Regulation, L2=Amotivation, L3=Introjected Regulation, L4=External Regulation, L5=Identified Regulation
S1=Language, S2=Efficacy Factor, S3=Efficacy Dimension

The results of the post-test on learning motivation and self-efficacy are shown in Table 5. There were significant differences in Integrated Regulation, External Regulation and Introjected Regulation among the sub-factors of learning motivation, but there was no significant difference in Amotivation and identification control. There was a significant difference in Efficacy Dimension in the sub-

elements of self-efficacy, but there was no difference in Language and Efficacy Factor.

Table 5. Post-test of group to measure for Learning Motivation & Self-Efficacy

Area	Group	N	M	SD	t	P
L1	Experimental	20	22.30	4.567	.2074	.044*
	Mental	19	19.33	4.902		
L2	Experimental	20	10.74	3.374	-	1.623
	Mental	19	12.43	3.529		
L3	Experimental	20	12.57	4.660	-.997	.324
	Mental	19	13.86	3.851		
L4	Experimental	20	10.78	4.680	-	1.863
	Mental	19	13.38	4.555		
L5	Experimental	20	18.65	3.688	2.308	.026*
	Mental	19	15.76	4.603		
S1	Experimental	20	33.35	7.352	.6511	.519
	Mental	19	31.86	7.844		
S2	Experimental	20	37.04	8.054	.650	.106
	Mental	19	33.24	7.162		
S3	Experimental	20	34.70	8.578	1.994	.050*
	Mental	19	30.05	6.659		

* p < .05

L1=Integrated Regulation, L2=Amotivation, L3=Introjected Regulation, L4=External Regulation, L5=Identified Regulation
S1=Language, S2=Efficacy Factor, S3=Efficacy Dimension

In order to analyze the difference of the self-efficacy before and after the application of the system in the experimental group, the mean and the standard deviation were calculated by dividing the test period. Table 6 shows the paired t-test results. As presented in the table, the sum of self-efficacy after system application is significantly higher than before the system application. In the analysis of sub-factors, there was a statistically significant difference in Efficacy Dimension and there was no difference in Language and Efficacy Factor.

Table 6. Paired Samples t Test of group to measure for Self-Efficacy.

Area	Group	Paired Differences			t	P
		N	M	SD		
Total	Pre	20	14.10	28.10	2.244	.037*
	Post					
S1	Pre	20	4.20	11.91	1.578	.131
	Post					
S2	Pre	20	6.60	8.18	3.606	.002*
	Post					
S3	Pre	20	3.30	11.32	1.303	.208
	Post					

* p < .05

S1=Language, S2=Efficacy Factor, S3=Efficacy Dimension

5. CONCLUSIONS

This study analyzed the effectiveness of learning motivation and self-efficacy by developing a pilot system that supports coding education using a block programming language for elementary school students and teachers. The results are as follow.

First, in the verification of effects on learning motivation and self-efficacy, it was found that the teaching method allowing learners to check their achievements constantly and enabling the teacher to identify students with low achievement from time to time and to give them one-to-one feed-backs would give more interest to learners and enforce them to achieve the goal.

Second, students were able to gain experience of success in the programming learning structure through the pilot system presented in this study. It gives them indirect experience of continuous success and the ability to create additional programs by showing achievement unlike the existing error checking method.

Based on the process and results of this study, it is necessary to provide a web-based lecture support system where the teacher can monitor learners in real time and provide a more convenient learning environment. In addition, it is necessary to study a system that can integrate and manage the tasks and achievements associated with the curriculum in conjunction with CMS or LMS.

6. ACKNOWLEDGMENTS

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Education) (No. 2017S1A5A2A01026058)

7. REFERENCES

Chang, W. Y. & Kim, S. S. (2014). Development and application of algorithm judging system: analysis of

effects on programming learning. *Journal of Korean Association of Computer Education*, 22(2), pp.15-24.

Han, K. W., Lee, E. K., Lee, and Y. J. (2010). "The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course". *IEEE Institute of Electrical and Electronics IEEE transactions on education*, 53(2), pp.318-327.

Jeong, J. K. (2010). Design and Construct of Programming Assessment System based on "Online Judge" for a Science High School student. master's thesis, Korea National University of Education.

Kim, K. S. (2014). Measuring and Applying the Self-efficacy in Computer Programming Education. *Journal of The Korean Association of Information Education*, 18(1), pp.111-120.

Katai, Z., Toth, L. (2010). "Technologically and artistically enhanced multi-sensory computer-programming education". *Teaching and teacher education*, 26(2), pp.244-251.

Kim, J.-H., Choi, J.-H., Shadikhodjaev, U., Nasridinov, A., and Song, K.S. (2018) "Chentry: Automated Evaluation of Students' Learning Progress for Entry Education Software," to be published in the *Advances in Intelligent Systems and Computing*, Springer.

Kim, M. H. (2007). Design and Implementation of an Automatic Grading System for Programming Assignments. *Journal of Internet Computing and Services*, 8(6), pp.75-85.

Lee, C. H., Kim, S. H. & Kim, D. M. (2016). Understand and actualization of software education. Seoul: Yangseowon.

Lee, M. H., & Jung, T. Y. (2007). Development and Validation of the Learning Motivation Scale. *Studies on Korean Youth*, 18(3), pp.295-321.

Levenshtein, Vladimir I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals". *Soviet Physics Doklady*, 10(8), pp.707-710.

Song, J. H. (2011). An Automated Assessment based Programming Education System for Self-Directed Learning. Doctoral dissertation, Soongsil University.

Wiedenbeck, S.(2005). Factors Affecting the Success of Non-Majors in Learning to Program. *The International Computing Education Research*, pp. 13-24.

Yadav, A., Hong, H., and Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms, *TechTrends*, Vol. 60, pp. 565-568.

Computational Concepts, Practices, and Collaboration in High School Students'

Debugging Electronic Textile Projects

Gayithri JAYATHIRTA^{1*}, Deborah A. FIELDS², Yasmin B. KAFAI¹

¹University of Pennsylvania

²Utah State University

gayithri@gse.upenn.edu, deborah.fields@usu.edu, kafai@upenn.edu

ABSTRACT

Debugging, a recurrent practice while programming, can reveal significant information about student learning. Making electronic textile (e-textile) artifacts entails numerous opportunities for students to debug across circuitry, coding, crafting and designing domains. In this study, 69 high school students worked on a series of four different e-textiles projects over eight weeks as a part of their introductory computer science course. We analyzed debugging challenges and resolutions reported by students in their portfolios and interviews and found not only a wide range of computational concepts but also the development of specific computational practices such as being iterative and incremental in students' debugging e-textiles projects. In the discussion, we address the need for more studies to recognize other computational practices such as abstraction and modularization, the potential of hybrid contexts for debugging, and the social aspects of debugging.

KEYWORDS

computer science education, programming, debugging, electronic textiles, making

1. INTRODUCTION

Debugging, the process to fix problems in code that prevent a computer program from functioning as intended, is recognized as a key computational thinking practice in engineering and computing (College Board, 2017; McCauley et al., 2008). In addition to being an important practice, debugging can also illuminate various areas of student struggle and provide opportunities for correction and support (Griffin, 2016). This is evident in studies where novice programmers' errors have illuminated misconceptions about specific concepts such as logical operators or understanding of control-flow statements (e.g. Brown & Altadmri, 2014).

Yet, debugging is an issue not just in computer science but also in engineering education (e.g., Patil & Codner, 2007). Electronic textiles construction kits, that include sewable microcontrollers, sensors, and actuators (Buechley, Peppler, Eisenberg, & Kafai, 2013), bring engineering and computer science together and generate, at times, interconnected problems for debugging. For instance, during the creation of an e-textile project, problems can occur in the code, in the circuitry, and in the crafting and physical design itself, and students need to test and isolate problems, often fix multiple co-occurring issues that add to the complexity of the project (e.g., Kafai, Fields, & Searle, 2014). Thus these hybrid projects provide an opportunity to promote deeper learning of debugging in engineering and computing, especially if we

consider debugging as a type of in-the-moment problem solving of projects (not just code) with errors.

In this paper, we investigate high school students' (14-18 years) debugging in the context of an eight-week long e-textiles curricular unit that took place within three introductory *Exploring Computer Science* classrooms (hereafter *ECS*, Margolis & Goode, 2016). During the unit, students from three classrooms created a series of four open-ended projects of increasing difficulty. In order to understand their debugging more deeply, we studied the problems that students reported they had to debug. Using end-of-unit written portfolios and interviews where students reflected on the challenges they encountered while creating their e-textiles projects, we studied the following questions: What types of challenges did students face, and in what content areas as they were making these projects? What kinds of computational practices did students report in relation to solving problems that came up? What social resources did they draw on to debug projects?

2. BACKGROUND

Debugging has been recognized as a key part of computational thinking for many years (Grover & Pea, 2013). As Papert (1980) noted, "[e]rrors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it" (p. 114). The historical teaching of debugging strategies has focused on helping students discover their own syntax problems (e.g., Robertson et al., 2004) or providing them with strategies for fixing and finding bugs (Carver & Risinger, 1987) through a variety of methods, such as debugging exercises and logs, reflective memos, and collaborative assignments (e.g., Griffin, 2016). Researchers have also developed different technical supports in the form of debugging tools. For instance, Tubashat (2001) provided tracing tools, while Thomas, Ratcliffe, and Thomasson (2004) offered visualizations and Robertson and colleagues (2004) investigated the timing of interruption tools. Nearly all of this research focused on on-screen programming since it was common in introductory programming courses then. As McCauley and colleagues (2008) noted in their comprehensive review of debugging research, it is unclear how findings and strategies developed from these earlier studies apply to visual programming languages and hybrid construction kits such as e-textiles which also involve collaborative work.

More recently, scholars have started to identify computational *practices* in computer science education, a focus not just on what concepts students are learning but how they are learning it and what thinking strategies they

develop. For instance, in their examination of students learning Scratch, Brennan and Resnick (2012) identified four computational practices: being iterative and incremental, testing and debugging, reusing and remixing, and abstracting and modularizing—each of which can result from rich programming experiences. Similarly, Sullivan (2008) outlined seven types of scientific thinking that student exhibited while thinking aloud about solving robotics problems: observing the problem, isolating the problem, generating a hypothesis, testing a hypothesis, controlling variables, manipulating variables, evaluating the solution, and estimating and computing. Together, these studies suggest taking a broader view of the thinking processes that debugging involves.

Several studies have shown that e-textiles can provide a complex context for debugging. The hybrid nature of e-textiles means that problems can occur in several overlapping areas of craft, design, circuitry, and coding (Kafai, Fields, & Searle, 2014; Lee & Fields, 2017). This means that identifying underlying problems is potentially tricky. However, prior studies of debugging in e-textiles have largely focused on areas of circuitry and physical craft, with only elementary computing concepts appearing in studies of debugging (see Litts, Kafai, Searle, & Dieckmeyer, 2016; Fields, Searle, & Kafai, 2016). Lack of time may be a reason for this since most e-textiles projects rarely exceed 16-20 hours of time on projects and rarely include more than one project requiring programming sensors or actuators. In our study, one goal of the e-textiles curricular unit design was to engage students more deeply in computational aspects of e-textiles for more time (roughly 40 hours of class time) with two projects involving coding.

Further, we intentionally looked at whether students discussed getting help from others in their descriptions of debugging in an effort to understand the collaborative nature of debugging. Previous debugging studies have focused mostly on individuals as if learning to debug was solely an individual endeavor (e.g. Fitzgerald et al., 2008). Yet learning in computer science does not happen in isolation. Kafai and Burke (2014) called for a reconceptualization of computational thinking as computational participation, explicitly recognizing the collaborative nature of computing. As collaboration is recognized as a key computational practice for learners to develop (College Board, 2017), some studies have noted the role of others in problem solving with computers or robotics. For instance, Deitrick and colleagues' (2015) analysis of a programming class through a socio-historical lens uncovers the intricacies of collaborative contexts where students, teachers and tools play a definite role in computational learning. Further, Jordan and McDaniel (2014) found that peers serve as a resource for managing uncertainty during problem solving. Yet much more needs to be understood about collaboration with debugging, especially in informal or ill-structured groups (versus pairs or small groups).

3. CONTEXT AND PARTICIPANTS

The ECS initiative comprises a one-year introductory computer science curriculum with a two-year professional development sequence. This inquiry-based curriculum has been successfully implemented with over 20,000 students.

In 2016, we co-developed an e-textiles unit for the ECS curriculum and piloted it with two teachers, focusing on teacher practices of making (see Fields, Kafai, Nakajima, Goode, & Margolis, in press). We revised the unit in 2017 and piloted it with three teachers, this time with a focus on student learning (the broader focus of this paper).

The revised unit took place over eight weeks and consisted of a series of four projects: 1) a paper-card using a simple circuit, 2) a wristband with three LEDs in parallel, 3) a classroom-wide mural project where pairs of students created portions that each incorporated two switches to computationally create four lighting patterns, and 4) a "human sensor" project that used two aluminum foil conductive patches that when squeezed generated a range of data to be used as conditions for lighting effects. Student artifacts included stuffed animals, paper cranes, and wearable shirts or hoodies, all augmented with the sensors and actuators. All the students also documented their projects in portfolios in which they summarized their projects, shared challenges that they faced, and reflected on their learning during the e-textiles unit.

In Spring 2017, three high school teachers, each with 8-12 years of computer science classroom teaching experience, piloted the e-textiles unit in their ECS classes in three large public secondary schools in a major city in the western United States. All three schools had socioeconomically disadvantaged students (59-89% of students at each school) with ethnically non-dominant populations (i.e., the majority of the students at each school include African American, Hispanic/Latino, or southeast Asian students). In School 1, Angela taught 22 students (6 girls and 16 boys), in School 2, Ben taught 36 students (17 girls and 19 boys), and in School 3, José taught 29 students (20 girls, 9 boys). All the students were of 14-18 years of age. All names of teachers and students are pseudonyms.

4. DATA COLLECTION AND ANALYSIS

Data for this project include all written portfolios submitted by consenting students (69 students from 3 classrooms) and interviews with pairs of students from each classroom (16 students total) discussing problems they encountered while making their e-textiles artifacts. We began analysis by identifying debugging episodes that students reported in their interviews and portfolios. We then grouped these episodes student-wise (69 students), combining two or more challenges whenever a student shared the same issue, both in the interview and the portfolio. This resulted in 210 total debugging episodes.

We coded the debugging episodes in a number of ways, drawing on concepts and frameworks from prior studies whenever applicable. To begin, each episode was classified by content (crafting, circuitry, programming, and design) and then sub-classified within more specific areas of these domains. For instance, we subdivided circuitry based on codes by Peppler and Glosson's (2012) research on e-textiles: connections, polarity, and current flow. For programming, we drew on Brennan and Resnick's (2012) framework of computational concepts: data, events, sequence, conditionals, logic operators, and loops. We also included syntax, an issue specific to text-based

programming language. However, with very little prior research done to understand student challenges in designing and crafting, we needed to develop new codes to categorize these challenges, including sewing mechanics, physical construction, and three-dimensional issues of design. Multiple codes could be used for each episode, since areas often overlapped (e.g., a problem involving both circuitry and code). We also included a “general” subcategory in cases of vaguely described problems.

In addition to analyzing content domains, we looked at computational practices students exhibited in their descriptions of the debugging process. For this we used both Brennan and Resnick’s (2012) framework of computational practices and Sullivan’s further subdivision of problem solving with robotics (see Section 2 for descriptions). Notably, Brennan and Resnick classify “testing and debugging” as one computational practice. However, while problem solving their projects, students often reported practices such as being iterative, so we included all practices identified by Brennan and Resnick and Sullivan in our coding of debugging episodes.

Finally, we considered the larger context of debugging, specifically what resources students used to resolve problems, including digital tools (e.g., Arduino IDE error message bar), physical tools (e.g., seam rippers or curved needles), or social resources (e.g., peers, teachers). Few students reported the use of digital or physical tools. However, many students frequently listed collaboration as a key resource while debugging. Below we share overarching findings from this analysis, focusing on computational concepts, computational practices, and collaborative resources to debug e-textiles projects.

5. FINDINGS

In the following sections, we report our findings under three categories—computational concepts, practices, and the collaboration that emerged from student portfolios and interviews analysis.

5.1. Computational Concepts Involved in Debugging

In earlier studies of debugging with e-textiles, crafting, circuitry, and simple computational challenges were the primary areas of debugging (Litts et al., 2016; Fields et al., 2016). In this study we found similar reporting of problems that arose in crafting and circuitry, but we also identified two other areas of debugging that were not discussed in earlier studies. First, students in our study reported *coding* challenges almost as often as crafting or circuitry and this highlighted some key coding concepts. Second, students also encountered new challenges in *three-dimensional design*. We describe these two areas in more detail below.

Among the 210 total debugging episodes, concepts discussed were almost evenly distributed across coding (29%), crafting (30%), and circuitry (28%). Within the episodes that discussed coding challenges and resolutions, a wide variety of concepts were reported, ranging from simple problems with syntax and labeling to more advanced issues with logical operators and control-flow statements. Forty-three students across three classes mentioned coding challenges at least once: a total of 61 episodes. Of these

debugging episodes focused on code, 64% of included “simple” issues that involved syntax, mislabeling variables or incorrect usage of constants. For example, some of these bugs included fixing brackets in conditional statements and functions, and mislabeling a sensor as “OUTPUT” instead of “INPUT.” While these are still relatively simple issues, resolving syntactical and labeling bugs such as these is a key practice in coding (McCauley et al., 2008).

However, 36% of the coding issues shared revolved around more complex computational concepts such as determining mathematical expressions for ranges of sensor values and managing multiple conditional statements. Consider David (School 1), who had difficulty determining the most effective ranges for his human sensor project. This project included two conductive patches that created a range of numerical values depending on how hard someone squeezed. Students had to create four ranges of these values and program them to trigger different lighting patterns. As David expressed, “it was harder to think of how big your range had to be so that it would actually react to how you want it to be.” After he realized his first attempt at coding ranges was inadequate, he iteratively tested the sensor, and represented a sequence of readings on a number line. Many students struggled with coding the ranges on their patches and took substantial time to fix them. Other more complex challenges that students faced included organizing multiple conditionals, especially if they involved two stages (i.e., using “if __, else __” instead of just a series of “if” statements), using additional sensors (e.g., light sensor) or in-built functions (e.g., random number generator). The variety and relative complexity of coding challenges reported by students highlight the affordances of e-textiles to support debugging both simple and advanced computational coding concepts.

Besides struggles with coding, another new area of struggle involved designing circuits on a three-dimensional artifact such as a stuffed animal or sweatshirt, especially common in the human sensor project. These designs required students to plan their circuitry two-dimensionally on paper but translate it onto a three-dimensional item. This posed new challenges to students. Thirteen of the 69 students (19%) specifically mentioned this issue within their debugging. For instance, while making his “Angry Bird” stuffed animal project, Rodrigo (School 1) realized he had to change his circuitry once he started working in three dimensions. “I made these changes because it was difficult planning out a 3D model on paper and if I hadn’t made changes to the pin numbers, then the paths would have crossed,” he explained. Photos from his portfolios are visible in Figure 1, where he showed two sides of the stuffed animal as well as his final circuitry diagram highlighting those same two sides (front and bottom). Though issues of three-dimensional circuitry design have not appeared previously in work on learning with e-textiles in K-12 education, it has come up with university students during clinical interviews (Lee & Fields, 2017), suggesting it may be an area of debugging that students face while working on more advanced projects. This also raises opportunities to consider spatial thinking in e-textiles design.

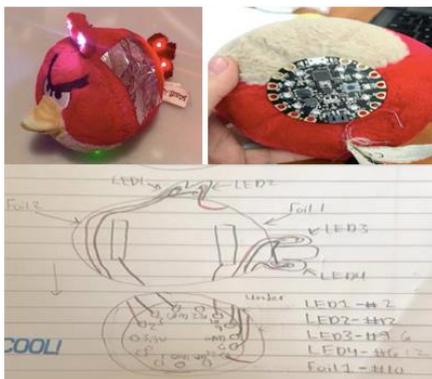


Figure 1. Rodrigo's Angry Bird project (top left to right clockwise): Upper view; bottom view (showing microcontroller); Circuit diagram.

5.2. Computational Practices Related to Debugging

In addition to content areas of debugging, we also sought to better understand the process of debugging, analyzing this through the computational practices lenses. Out of 69 students, 60 shared at least one of the four standard computational practices suggested by Brennan and Resnick (2012) in their framework. Out of these four practices, testing and debugging was the most mentioned (47 students), followed by iterative and incremental practices (35 students). The two other practices, abstraction and modularization, and reusing and remixing were rarely discussed. This may be because of how the questions were phrased in interviews and in the portfolio, which focused on challenges students faced. For instance, in their focus on problems, students did not mention remixing designs although remixing and reusing daily-use items such as backpacks and soft toys was an integral part of their human sensor project. Further, though there were opportunities for applying abstraction and modularity (i.e., breaking down a project and/or code into parts), this did not seem to be a conscious way that students thought about this process with regard to problem solving. However, yet another computational practice that emerged from student descriptions was collaboration, which is also presented as a perspective in Brennan and Resnick's (2012) framework. Thirty-six students reported on collaboration as an integral aspect of fixing errors, leading us to suggest collaboration as more of a computational practice rather than a perspective developed, which we will elaborate shortly.

Though all debugging episodes concerned students fixing issues, in some instances students shared more specific details about how they identified, isolated, and otherwise focused on understanding a particular problem. In these 47 instances, we coded for specific areas that Sullivan (2008) identified. The most prominent of these were observing the problem (46 students), isolating the problem (43 students), and generating a hypothesis about the cause of the problem (35 students). As an example, consider how Alexa and Antonio (School 2) worked through a series of circuitry problems in their Pacman-themed mural project (see Figure 2). As Alexa expressed in her portfolio: "[In] our first design we wanted the playground on the back of project. When we tried that, the conductive thread crossed each other... We dealt with our problem by redesigning our project, so that the playground was in the front and the conductive thread

wasn't touching." Alexa and Antonio first observed the source of the error as the short-circuit (crossed threads) and hypothesized that the spatial placement of the Circuit Playground (microcontroller) at the back of their Pacman mat was causing the short circuit. They were able to isolate specific locations where these short circuits occurred and plan their next iteration to fix them.



Figure 2. Alexa and Antonio's Pacman project

Along with testing and debugging, being incremental and iterative was another other key computational practice evident in student narrations. Of the 35 students who shared about this, 29 discussed incrementally revising their project design and 10 shared about repeatedly testing their sensor values and adjusting their project code to suit the varying values. (Note: we classified *repeated* testing of a problem under iteration rather than testing and debugging). One of the key challenges underlying revisions was translating project plans from paper representations to physical artifacts. As previously mentioned, many students had to revise their project upon realizing that their plan on paper did not work when sewn in three dimensions. For instance Alma (School 2) expressed that "[W]hen sewing [our project] we realized that everything was basically backwards" and had to substantially change the placement of each LED so to have "clean lines" without short circuits.

Besides design translations, the other major area of being iterative and incremental was in testing the sensor patches. Here David (School 1) again provides an explanation for what iterative testing looked like:

So from my last project, it was a human sensor and my scales were... pretty much wrong to the point where only one pattern worked... [T]o fix the problem... I slowly started testing out. So, I touched it. Okay, this is the values for a light touch, just inputted that. I said, 'let's squeezed it harder.' [sic] I looked at the values, and inputted that... As I looked at the values, I am like, okay, the range from this to the next pattern, it's kinda too small. So I have to make it bigger so that it can be a bit more sensitive.

This encouraging example of iteration demonstrates the careful way that some students had to work to program their sensors. Often their first attempt would result in poorly thought-out ranges, and, like David, students had to proceed through cycles of testing and adjusting the range of values corresponding to squeezing. Though only 10 students described this particular process, it is a practice that could be expanded on more intentionally in future iterations of the curriculum and in debugging pedagogy more generally.

5.3. Collaboration Contexts Related to Debugging

One unexpected finding was how often students' debugging involved collaboration with classmates, partners and

teachers. Most students (75%) explicitly mentioned help they received from peers or a teacher in at least one of the challenges they described (in 36% of the challenges overall). Unlike an earlier study that observed low peer collaboration in e-textiles (Litts et al., 2016), this analysis revealed student engagement with different types of collaborators throughout their e-textiles debugging, from their immediate partners on a project, to students at the same table, to the wider class community.

Students reported different kinds of supports that they received from peers and teachers across a range of issues—from identification of syntactical errors to understanding concepts such as conditional statements. An example for a simple support includes Ethan's (School 3) reporting of dim lights in his quilt project. His classmate helped him locate and isolate the problem: missing a line in the setup section of the code that initialized the pin to OUTPUT. Students also mentioned getting help with more complex struggles. For instance, Allie (School 2) used her classmates to test the sensors of her human sensor project, using "different people's pressure" and changing the ranges in her project. Surprisingly, students rarely mentioned teacher participation in debugging (close to 11% of challenges).

Collaboration was mentioned frequently in students' reports of debugging although students were graded individually for this unit. That so much collaboration was evident in these contexts suggests that there is much more to discover about unstructured peer-to-peer debugging in students' e-textiles design processes and in debugging open-ended computational projects.

6. DISCUSSION

Our analysis of student challenges and solutions demonstrates that debugging open-ended e-textiles projects can provide a rich context for students to experience a range of computational concepts and practices. Our study noted promising new areas of conceptual struggles for e-textiles students, specifically in the domains of coding and three-dimensional design. We think this is because students were able to go deeper in these areas with two advanced e-textiles projects compared to prior studies that only had one such project (e.g., Fields et al., 2016; Litts et al., 2016). This suggests that pursuing a *series* of challenging e-textiles projects may provide more opportunities for deeper learning of computing concepts and practices than just one or two projects. It also raises the potential for supporting debugging more generally by creating a series of projects in other computational domains, not just e-textiles.

In addition to conceptual learning, students in this study reported using certain computational practices such as being iterative, testing and debugging, and collaboratively problem solving. Interestingly, within the area of debugging, students' reports consistently highlighted the need to identify and isolate problems, something that should not be trivialized. Unlike other studies of debugging that focus solely on debugging code (e.g., Brown & Altadmri, 2014), students with e-textiles projects had to consider the origin of a bug from among several possibilities: code, circuitry, craft, or spatial design. Yet, we also recognize that this study was limited to students' *reporting* of bugs rather than a study of

observing of how they actually solved them. This opens up the need for deeper research on students' in-the-moment debugging to see whether students engage in other steps of debugging such as manipulation of variables, evaluation of solutions, and estimation of data.

One other key finding was frequent student collaboration during problem solving. Students shared collaboration not only at the level of formal pairs and small groups but within the broader classroom, turning the class into a community of learners. The physical layout of the classroom with tables and shared supplies along with the teachers' allowing students to move between tables may have encouraged this fluid collaboration (Fields et al., in press). More so, these findings call for a reconceptualization of collaboration in these spaces to better understand the roles taken on by different participants. A closer look at these types of settings may help us understand and classify different kinds of supports students provide to each other. Such an analysis could also help us understand the supportive role of teachers in creating collaborative classrooms, informing the development of new pedagogical approaches for students and professional development for teachers.

The interdisciplinary nature of e-textiles provided a unique opportunity to study debugging in a hybrid context. Further, the ability of debugging exercises to develop computational thinking and practices in learners has called for "explicit instruction in debugging [to] be fundamental to any beginning programming class" (p. 86, McCauley et al., 2008). If debugging is a core area of computation, then as a field we need to look beyond code-only settings of computation to hybrid settings (including but not limited to e-textiles) where students are introduced to debugging in more challenging situations which demand multiple iterations of revising and testing. Further, more studies of debugging are needed in many contexts that look at it less as an individualistic and more as a social practice, moving from computational thinking to computational participation (Kafai & Burke, 2014).

7. ACKNOWLEDGEMENTS

This work was supported by grants from the National Science Foundation to Yasmin Kafai, Jane Margolis, and Joanna Goode (# 1509245), and Yasmin Kafai and Mike Eisenberg (#1742140). Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of NSF, the University of Pennsylvania, or Utah State University. Special thanks to Tomoko Nakajima for her help with data collection and to Debora Lui, Justice T. Walker, and Mia Shaw for their valuable feedback.

8. REFERENCES

- Brennan, K. and Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. Annual Meeting of the American Educational Research Association Vancouver, BC, Canada.
- Brown, N. C., & Altadmri, A. (2014, July). Investigating novice programming mistakes: Educator beliefs vs. student data. In *Proceedings of the tenth annual*

- conference on International computing education research (pp. 43-50). New York, NY: ACM.
- Buechley, L., Pepler, K. A., Eisenberg, M. & Kafai, Y. B. (Eds.) (2013). *Textile Messages: Dispatches from the Word of Electronic Textiles and Education*. New York, NY: Peter Lang Publishers.
- Carver, S. & Risinger, S. (1987). Improving children's debugging skills. In G. Olson, S. Sheppard & E. Soloway (Eds.), *Empirical Studies of Programmers: Second Workshop* (pp. 147-171). Norwood, NJ: Ablex.
- College Board (2017). *Advanced Placement Computer Science Principles Course Guide*. Retrieved from <https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf>
- Deitrick, E., Shapiro, R. B., Ahrens, M. P., Fiebrink, R., Lehrman, P. D., & Farooq, S. (2015, July). Using distributed cognition theory to analyze collaborative computer science learning. In *Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 51-60). New York, NY: ACM.
- Fields, D. A., Searle, K. A., & Kafai, Y. B (2016). Deconstruction kits for learning: Students' collaborative debugging of electronic textile designs. In *FabLearn '16, Proceedings of the 6th Annual Conference on Creativity and Fabrication in Education* (pp. 82-85). New York, NY: ACM.
- Fields, D. A., Kafai, Y. B., Nakajima, T. M., Goode, J. & Margolis J. (in press). Putting making into high school computer science classrooms: Promoting equity in teaching and learning with electronic textiles in *Exploring Computer Science. Equity and Excellence in Education*
- Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, 18(2), 93-116.
- Griffin, J. M. (2016, September). Learning by taking apart: deconstructing code by reading, tracing, and debugging. In *Proceedings of the 17th Annual Conference on Information Technology Education* (pp. 148-153). New York, NY: ACM.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Jordan, M. E., & McDaniel Jr, R. R. (2014). Managing uncertainty during collaborative problem solving in elementary school teams: The role of peer influence in robotics engineering activity. *Journal of the Learning Sciences*, 23(4), 490-536.
- Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.
- Kafai, Y., Fields, D., & Searle, K. (2014). Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review*, 84(4), 532-556.
- Lee, V. R. & Fields, D. A. (2017). Changes in undergraduate student competences in the areas of circuitry, crafting, and computation after a course using e-textiles. *International Journal of Information and Learning Technology*, 34(5), 372-384.
- Litts, B. K., Kafai, Y. B., Searle, K. A., & Dieckmeyer, E. (2016). Perceptions of productive failure in design projects: High school students' challenges in making electronic textiles. *International Conference of the Learning Sciences*, 498-505.
- Litts, B. K., Kafai, Y. B., Lui, D. A., Walker, J. T., & Widman, S.A. (2017). Stitching codeable circuits: high school students' learning about circuitry and coding with electronic textiles. *Journal of Science Education and Technology*, 26(5), 494-507.
- Margolis, J., & Goode, J. (2016). Ten Lessons for CS for All. *ACM Inroads*, 7(4), 58-66.
- McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, 18(2), 67-92.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY: Basic Books.
- Patil, A., & Codner, G. (2007). Accreditation of engineering education: review, observations and proposal for global accreditation. *European Journal of Engineering Education*, 32(6), 639-651.
- Pepler, K., & Glosson, D. (2012). Stitching circuits: Learning about circuitry through e-textile materials. *Journal of Science Education and Technology*, 22(5), 751-763.
- Robertson, T., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, F., Beckwith, L., et al., (2004). Impact of interruption style on end-user debugging. In E. Dykstra-Erickson & M. Tscheligi (Eds.). *Proceedings of CHI'04* (pp. 287-294). New York, NY: ACM.
- Sullivan, F. R. (2008). Robotics and science literacy: Thinking skills, science process skills and systems understanding. *Journal of Research in Science Teaching*, 45(3), 373-394.
- Thomas, L., Ratcliffe, M. & Thomasson, B. (2004). Scaffolding with object diagrams in first year programming classes: Some unexpected results. *ACM Inroads*, 36(1), 250-254.
- Tubaishat, A. (2001). A knowledge base for program debugging. In *Proceedings of the International Conference on Computer Systems and Applications* (pp. 321-327). Beirut: IEEE Press

A School-wide Approach to Infusing Coding in the Curriculum

Sirajutheen Shahul HAMEED¹, Chee-wah LOW¹, Poh-tin LEE¹, Nur Illya Nafiza MOHAMED¹, Wuay-boon NG¹,
Peter SEOW², Bimlesh WADHWA³

¹Bukit View Secondary School, Singapore

²National Institute of Education, Nanyang Technological University, Singapore

³National University of Singapore, Singapore

sirajutheen_shahul_hameed@moe.edu.sg, chee_wah_low@moe.edu.sg, poh_tin_lee@moe.edu.sg,
nur_illya_nafiza_mohamed@moe.edu.sg, wuay_boon_ng@moe.edu.sg, peter.seow@nie.edu.sg, bimlesh@nus.edu.sg,

ABSTRACT

This paper shares a school's journey in the implementation of a school-wide programme for students to learn and apply computing over the years in school. The school leaders see the value of having students learn computing and coding as it provides students with opportunities to understand how technology works and applies to solve problems in the world. The school worked towards the design of a programme that enables all students to learn coding. The design of the programme is underpinned by Papert's theory of Constructionism which postulates that students learn best when engaged in concrete experiences of creating artefacts. In implementing a school-wide programme that would span over a student educational experience in school, the school is cognizant of issues of a packed school curriculum and teachers' lack of experiences in computing. The school addressed the issue of a packed curriculum by structuring time for learning computing in the time table and integrating with school subjects. Training partners were engaged to work with teachers in designing and integrating computing with the subjects to address the lack of teachers' experience. Teachers' capacity continue to be developed as they gain competence in computing. The school continues to improve the programme for sustainability and richer learning experience for the students.

KEYWORDS

Coding, Computing, school-wide programmes, Implementation.

1. INTRODUCTION

Computing technology has changed the way we live, work and learn about the world around us. Beyond being users of technology, there is a need to understand how computing technology works and apply this understanding to solve problems and innovate new ideas that would improve our lives. Computing technology is transforming manufacturing with the vision of Industry 4.0 integrating Cybernetics, data analytics, cloud computing, Machine learning and Internet of Things to create Smart factories that monitor processes and make decentralized decisions. The emergence of how various computing technologies are integrated and utilised is creating a demand for new skills and capacity to drive the economy. Hence, there is a need for students to better understand how computing technology works and harness it use to improve lives. Around the world, there is a growing emphasis on introducing Computational Thinking and coding to students in schools (Brown, Sentence, Crick and Humphreys, 2014). Countries such as England have made the teaching of CT skills compulsory in the curriculum and all students will learn programming (DfE, 2017). Japan,

Korea and Malaysia have announced plans to introduce programming as part of students' compulsory education (Japan Times, 2017; APFC, 2017). However, implementing programming as compulsory education and the integrating Computational Thinking into the curriculum is challenging (Sentence and Csizmadia, 2017)

In our context, we find, teachers lack the content and pedagogical knowledge of Computing and Computational Thinking to know how to integrate into the curriculum. Compared to established fields of study in Sciences such as Physics or Chemistry, the study of Computer Science in K-12 is relative new as the computing technology is still evolving. Teacher training institution does not offer programmes to train teacher in teaching Computing as a subject. Interested teachers do not have the opportunities to learn Computing and pedagogy of teaching Computing. Teachers need to have a Computing background or undergo Computing training to teach Computing to students. The lack of teachers with required skills and content knowledge impedes the teaching of Computing and Computational Thinking if all students in a school are to be taught. Second, teaching Computing and Computational Thinking is still an emerging field of study in K-12 education settings compared to tertiary education. At the tertiary level, students pursue computing degrees which provides opportunities for them to learn the theory of Computing and develop the practice over the course of the study. Their programming skills and craft are developed when they work on projects and assignments in various topics like learning programming languages, operating systems, artificial intelligence, computer networks, data sciences and databases. Computing covers a wide field of study with concepts that are difficult to introduce in K-12 settings. Teachers need to know what topics are relevant and appropriate to teach the students in K-12 schools. Lastly, there is lack of the curriculum time and space for schools to integrate the teaching of Computing and Computational Thinking. Students in Singapore secondary schools are required to take the core subjects of English, Mother Tongue, Mathematics, Sciences, Social Studies, and Humanities. In addition, there are Co-Curricular activities which all students are to participate as part of their holistic education. There are school wide programs in niche areas of learning such as leadership, entrepreneurship, drama or environment science with programs. Even if schools want to introduce Computing to all students, it is a challenge to find the time to implement the teaching of Computing and Computational Thinking in a crowded curriculum space.

This paper documents a school's journey in developing a school-wide programme – Coding and Computational

Thinking infUSED Curriculum (CaCTus) for teaching Computing and Computational Thinking. The school leaders and teachers worked together to implement a school-wide curriculum that introduces students to the concepts of Computing and Computational Thinking over 3 to 4 years of their studies in the school. The programme is designed for students to learn and apply Computing concepts through integration with school subjects that they are learning in the classroom.

2. SCHOOL BACKGROUND

Bukit View Secondary School (BVSS) resides in the typical suburban neighbourhood in Singapore. The student enrolment is 1005 with 61% Chinese, 18% Malay, 16% Indian and 5% Others (compared to 74.3% Chinese, 13.3% Malay, 9.1% Indian and 3.3% Others nationally) housed in 25 classes in 2017. About 45% of the students speak English (compared to 36.9% nationally) as their main language of communication at home with the rest using their Mother Tongue. The profile of the parents' highest education attained for Secondary/ITE, Pre-U/Polytechnic and University are 46%, 25% and 25% respectively (compared to 26.1, 14.6, 30.7 nationally).

BVSS has offered Computer Studies as an O-Level subject since 2006. Since 2017, BVSS is only one of 19 schools in Singapore that offers the new Computing syllabus.

3. DESIGN OF THE CACTUS CURRICULUM

The school leaders and teachers saw the importance of learning Computing as it has the potential to develop problem solving skills for students in the world they live in. Also, the school leaders saw strong connection of Computing to other disciplines such as Mathematics, Engineering, Science, and Design and Technology. Acquiring skills in computing can be applied in above subjects. Also, using Computing can help solve problems in the domains in health care, environment, business, and engineering. Finding solutions to some of these problems requires computational skills and knowledge. Above observations formed basis for setting out following goals of CaCTus:

- To enable BVSS students to better understand the fast evolving world due to digitalization.
- To improve BVSS students' thinking skills in applying the concepts to solve problems in a dynamic way.
- To open doors to a host of opportunities for BVSS students in the future, regardless of the career path.

The design of the CaCTus draws from the ideas of Papert's idea of Constructionism (Paper and Harel, 1991). Based on Piaget's constructionist theory [] of learning where students construct their own knowledge from their prior understanding, Papert extends it by stating that students learn as they are engaged in meaningful concrete experiences. These concrete experiences can be in a form of designing, constructing and programming an artefact like a robot or building a kit to measure the quality of water in a

pond. Following such an approach, students are participate in the process of identifying a problem, experimenting with various ideas, designing, constructing and testing a solution to the problem. Through these processes, mental models of the world around them and their naïve scientific concepts can be constructed and refines. Computers and mobile phones are now part of everyday lives but they operate very differently from the mechanical devices with gears and levers. If students are limited to being users of computing devices, i.e. seeing only printed circuit board with chips and LED displays, they will have naïve mental models of how the devices actually function. Having primitive or incomplete mental models about computing devices could impede their learning about and with computers in future. Developing codes and knowing how computing systems are created to solve problems can help children to construct mental models of how technology and their different parts work together. More importantly, students "can use programs to understand their world, and manipulate their world" (Guzdial, 2012). In the CaCTus programme, our goal is not for students to become Computer Scientists, but for all students to better understand more about the world around and their thinking processes as they use technology in concrete ways to solve problems. It is through a constructionist approach as Seymour Papert iterated that "computers might enhance thinking and change patters of access to knowledge." (Papert, 1980).

In the design of CaCTus programme, the following ideas guided us:

- Every student in Bukit View Secondary should have access to the same learning experiences.
- The learning experiences should be continuous and connected over their stay in the school.
- Students should develop 21st century skills such as developing critical thinking skills, solving problems, collaborating with others and developing creativity.
- Each student should have a rich experience in using Computing and Coding to solve problems in authentic contexts.
- There is a diversity of learning experiences for the students to learn, construct and apply their knowledge.

Based on above ideas, the school leaders and teachers worked together with partners in designing rich learning experiences for students to apply technology in authentic context to better understand their world.

4. IMPLEMENTATION OF CACTUS

Since 2013, the school had separate enrichment programmes for students at various levels to learn coding such as Scratch. The enrichment programmes were consolidated and reorganized as CaCTus in 2016 for all students in the school to have a continuous learning experience of Computing and coding from Secondary 1 to 3. A school-wide approach was adopted. However, implementing a school-wide approach competed with the demands of curriculum and co-curricular activities. To

address this issue, CaCTus was structured into the school timetable with two 40-minute periods each week over a 20 week semester. A modular approach was taken so that all students were able to participate and experience computing programmes. Efforts were also made to integrate the modular activities with the curriculum in subjects such as Math, Science, Geography and Design and Technology. For example, Secondary 1 students were introduced to use Scratch and create a visual simulation of the Water Cycle in their Science lesson. For Secondary 2 students, drone programming was introduced to make them understand how such technology can be used to study geographical features in their Geography lessons.

Table 1 shows the various modules that were designed for students in school from Secondary 1 to Secondary 3.

Table 1. Cactus Modules

Programmes	
Secondary 1	Scratch Animation – Bio Water Cycle IDA Lab on Wheels Coding – Spheros Programmable Robot Hour of Code by Salesforce Learning journey to Salesforce
Secondary 2	Scratch Coding – National Education Salesforce talk on Coding Sea Perch – Collecting data and analysis of water quality Drone programming Coding workshop @ Nanyang Polytechnic
Secondary 3	Learning journey to IMDA* Advanced Elective Module Coding workshops @ Nanyang Polytechnic

*IMDA – Infocomm Media Development Authority

The programmes were designed to provide students opportunities to learn computing and coding to understand the world around them. In the Sea Perch (See Fig. 1), students collect water quality data from the pond and analyse the collected quality. It is through the data analysis that students find meaningful interpretations and understand the chemical content in the water that otherwise would not be obvious to them. They have the opportunity to observe and experience how data is collected through sensors, manipulated and visualized to determine the quality of water.

In the design of CaCTus, the goal is to provide every student with several rich authentic experiences in using computing and coding to solve problems during their school years. The students' learning experiences are not one-off but continue to build on their prior experiences as they move to the next grade. To operationalize CaCTus, the school planners are cognizant of the challenges in the design and implementation of the programme. First, there were not enough teachers to design and run the computing programmes. Most of the teachers are subject teachers in Sciences, Mathematics, Humanities and the Arts, and they do not know much about coding or integrate coding into their respective subject curriculum. Teachers' would typically not buy-in into the program if they feel that they

lack the skills or experience to teach the students. Secondly, with a packed curriculum schedule, it was a challenge to implement a programme for all students. A typical student in Singapore secondary school takes 7 subjects. In addition, each student would also participate in Co-curricular activities during the school day as well as other school-wide programmes such as the Applied Learning Programmes (ALP) or outdoor programmes such as the Outward Bound School (OBS).



Figure 1. Sea Perch collecting water quality data.

To address the challenge of lack of computing skills among teachers, the school leaders and core teachers partnered with technology training vendors, experienced in coding, and government agencies such as the Infocomm Media Development Authority (IMDA). The school leveraged on funding and resources from IMDA to design and run the programs. In the initial stages of the implementation, the training vendors designed the learning experiences with a selected group of teachers. The teachers ensured that the learning experiences are aligned to the goals and ideas of CaCTus. Working with different vendors and activities, the teachers looked at how the various experiences are connected and applied to the subjects. Efforts were made to ensure that students' experiences are built upon and continued as they progressed from one grade to another. For example, students introduced to visual programming tools in Secondary 1, continue to use the tools such as Scratch and Microbit Block-based programming in Secondary 2.

In the second year of implementation, school leaders engaged the training vendors to conduct workshops for all teachers to learn and participate in coding and computing experiences. During such workshops, teachers built games such as Tic-Tac-Toe, and explored various ways to program Microbit board e.g. displaying their name with the LED display. Teachers also learnt about algorithmic thinking by creating sequenced codes to control a robot and drone. Creating these experiences provided opportunities for teachers to equip themselves for introducing simple concepts of how these technologies function, to the students. Eventually, the goal is for all teachers to think about how computing and coding can be integrated into their teaching subject areas such as Math, Science or Humanities. Also, The school worked with industry partners such as Salesforce, IMDA and Nanyang Polytechnic for learning journeys and workshops. The

exposure to industry and polytechnics is aimed to enthuse students in seeing the practice of computing outside school.

5. FUTURE PLAN FOR CACTUS

The school's Applied Learning Programme (ALP) and CaCTus programme were run as independent modules over each semester since 2016. In 2017, the BVSS team reviewed the ALP and CaCTus programmes and saw synergies in both. Consequently, the school has decided to integrate both programmes and move into a year-long programme for each level. The integrated programme has been named as the Junior Outstanding Leaders in Energy for Sustainability (JOULES) programme. It is a distinctive programme that focuses on Science, Technology, Engineering and Mathematics (STEM) education. This expanded 4-year programme provides students with knowledge and experience in design thinking and coding for environment and sustainable energy.

JOULES emphasizes on STEM and environmental advocacy to develop leaders of the future who will continue to champion sustainable development through the use of technology. Raising the innovation quotient amongst the student population is another aim of the programme. It is also hoped that the enriching experience of the JOULES programme will inspire their students to pursue relevant STEM courses in their higher education, and contribute positively to Singapore and the world.

The student outcomes include the following skills and dispositions: problem solving, design thinking, computational thinking, scientific literacy and inquiry, and mathematical reasoning.

6. CONCLUSION

This paper shares the experiences of designing and implementing a school-wide approach for all students in the school to learn and apply computing. The school leaders recognized the importance for all students to have experiences in learning computing by structuring programmes into the time table as a subject and integrating computing into subjects such as Mathematics, Science, and Geography. Students could learn to apply coding into the subjects and teachers could better integrate computing into their subjects. The school leveraged on training partners to work with school teachers in the initial phases to address the lack of computing experience. In the later stages, the school sought to develop teachers' competence in using computing

for their subjects. To better sustain the programme and provide students' a richer learning experience, the new programme aims to develop students' skills in Computational thinking, design thinking, inquiry and problems solving. We hope that sharing the school's journey would provide some understanding in how schools can implement programmes in learning computing for all students in the school.

7. REFERENCES

- APFC (2017). *Preparing Students for South Korea's Creative Economy: The Successes and Challenges of Educational Reform*. Retrieved Feb 13, 2017, from <http://www.asiapacific.ca/research-report/preparing-students-south-koreas-creative-economy-successes>
- Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE)*, 14(2), 9
- DfE. (2017). *National Curriculum in England: computing programmes of study*. Retrieved Feb 13, 2017, from <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study>
- Guzdial, M. (2012, May). 21st Century Literacy includes Computing for Everyone [Video file]. Retrieved from https://www.youtube.com/watch?v=mGc6clf_Wt4&feature=youtu.be&t=16m33s
- Japan Times. (2017). *Computer programming seen as key to Japan's place in 'fourth industrial revolution'*. Retrieved Feb 13, 2017, from http://www.japantimes.co.jp/news/2016/06/10/business/tech/computer-programming-industry-seen-key-japans-place-fourth-industrial-revolution/#.WKG2P_197b0
- Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*, 36(2), 1-11.
- Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.
- Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2)

Learning to Code—Does It Help Students to Improve Their Thinking Skills?

Ronny SCHERER^{1*}, Fazilat SIDDIQ², Bárbara SÁNCHEZ VIVEROS³

¹ Centre for Educational Measurement at the University of Oslo (CEMO), Faculty of Educational Sciences, Oslo, Norway

² The Nordic Institute for Studies in Innovation, Research and Education (NIFU), Oslo, Norway

³ Humboldt-Universität zu Berlin, Berlin, Germany

Ronny.scherer@cemo.uio.no, Fazilat.siddiq@nifu.no, Saviverb@hu-berlin.de

ABSTRACT

Learning to code is claimed to be associated with improvements in other cognitive skills, such as creative thinking, reasoning, and mathematical skills. Although the claims surround this transferability of coding skills have already been made in the 1980s and 1990s, the existing body of research does not provide clear insights into the transfer effects of learning to code. The current meta-analytic review sheds light on these effects. We retrieved an overall sample of 105 experimental and quasi-experimental studies with posttest-only or pretest-posttest treatment-control group designs and extracted 539 effect sizes. A three-level random-effects modeling approach revealed an overall transfer effect size of $g = +0.49$. Differentiating between the types of cognitive skills (i.e., coding, reasoning, creativity, and math skills), however, indicated differential effects. Study and sample characteristics were further examined as possible moderators. Overall, this study identifies positive transfer effects of learning to code on cognitive skills.

KEYWORDS

Coding skills, transfer effects, meta-analysis, cognitive skills

1. INTRODUCTION

Undoubtedly, the rapid developments in technology have impacted many areas of society. Even in education—a field that is known for its slow progress—things are changing: Educational systems around the world include teaching programs that will help students to acquire skills beyond literacy and numeracy. Among others, these skills comprise complex problem solving, global competences, critical thinking, creativity, digital literacy, and computational thinking (Binkley et al., 2012; ICILS, 2018). Interestingly, the latter has recently gained considerable attention. Bill Gates, for example, established its importance by stating that “Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that I think is helpful in all domains”. The claim that learning to code—a critical step in the process of acquiring computational thinking skills (Denning, 2010; Grover & Pea, 2013; Shute et al., 2017)—transfers to other cognitive skills, however, stands on shaky legs. Scherer (2016) concludes that studies examining transfer effects disagree in the extent to which these effects can be established for specific cognitive skills. Sala and Gobet (2017) warn against the assumption that learning a specific skill improves other skills as well. The authors further propose to examine hypothesized transfer effects meta-analytically to synthesize the body of existing evidence. At this point, we notice that the concept of computational

thinking is broader than coding, albeit coding is the essential part of it (Shute et al., 2017).

Although the discussion surrounding the transfer-ability of learning to code on other cognitive skills dates to the 1980s and 1990s, the existing body of research abounds in conflicting findings, and previous attempts to meta-analyze the transfer effects were flawed (Scherer, 2016). For instance, Liao and Bright (1991) extracted 432 effect sizes from 65 studies and summarized them to an overall transfer effect size of $d = +0.41$. Although this finding indicates that positive transfer to other cognitive skills may exist, the authors neglected (a) the clustered structure of their data set (i.e., effect sizes are nested in studies), and (b) the possible differences in effects between cognitive skills. Later, Liao (2000) provided an update and presented an overall effect of $d = +0.76$, obtained from only 22 studies. Since then, the critical question whether learning to code improves cognitive skills has not been addressed explicitly in meta-analyses.

The present study tests the claim that learning to code transfers to the acquisition of other cognitive skills. Synthesizing the empirical evidence on transfer effects, we take two main steps: First, an overall effect size is presented, and its variation within and across studies is quantified. Second, possible moderation effects of selected study characteristics are explored to explain this variation.

2. METHODOLOGICAL APPROACH

This section describes the meta-analytic procedures, including the literature search and screening, the sample obtained from them, and the statistical approaches taken to summarize the transfer effects of coding skills.

2.1. Literature Search and Screening

Relevant literature was identified in existing databases—including PsycINFO, ERIC, IEEE Xplore, ACM Digital Library—next to academic journals relevant to the field (e.g., *Computers & Education*, *Computers in Human Behavior*), existing reviews and meta-analyses (e.g., Liao & Bright, 1991; Liao, 2000; Shute et al., 2017), and informal resources (e.g., ResearchGate, personal contact with authors, publication lists of scholars). The literature search was constrained to studies that had been published between 1965 and 2017. After an initial screening of titles and abstracts with respect to their topic fit (i.e., computer coding) and the empirical nature of the presented study, abstracts and full texts were submitted to a more fine-grained screening. This screening was based on the following inclusion criteria:

- (a) *Study design and control group*: Only studies were considered with an experimental or quasi-experimental design and at least one control group

(i.e., a group of participants not exposed to the coding intervention).

- (b) *Outcomes*: Only studies were considered with performance-based outcome measures.
- (c) *Study context*: Only studies were considered that conducted the experiment or quasi-experiment in an educational context.
- (d) *Sample*: Only studies were considered with non-clinical samples, because clinical samples often involve participants with conditions that may interfere with their performance on cognitive skills tests.
- (e) *Effect sizes*: Only studies were considered that reported effect sizes directly or provided statistics sufficient to calculate transfer effects.

2.2. Sample

The initial literature search resulted in 5,193 publications. As these entries were subjected to an initial screening and the application of inclusion criteria, more than 80 % of them were excluded and no longer considered for further coding and data extraction—overall, 105 studies were retrieved, and 539 effect sizes could be extracted. Of these 105 studies, 89 studies reported interventions implemented in regular classroom lessons, 8 studies reported interventions as part of extra-curricular activities; all other studies reported interventions outside of schools but in an educational context. The sample of studies spanned all educational levels, ranging from pre-kindergarten to adult education. Concerning the coding tools used in the interventions, both text-based and visual coding languages were used to help students learn to code. All studies contained cognitive skills measures that assessed either coding skills or skills outside of the coding domain. Among others, these skills include: Creative thinking (i.e., skills related to the originality, fluency, flexibility, and elaboration of ideas and generating ideas), reasoning skills (i.e., logical thinking, intelligence, critical thinking, and problem solving), and mathematical skills (i.e., understanding mathematical concepts, mathematical problem solving and modeling).

2.3. Statistical Approach

Given the hierarchical nature of the sample of studies—as indicated by the availability of multiple effect sizes for single studies—the statistical approach taken to aggregate transfer effect sizes had to represent this nature adequately. Although several approaches exist in the meta-analytic literature, only few qualify for application in this study. Because most studies did not report correlations between multiple outcome variables, we adopted a three-level modeling approach, allowing for within- and between-study variation of effect sizes (Moeyaert et al., 2017). This approach performs reasonably well in the presence of hierarchically structured datasets with effect sizes nested in studies (Cheung, 2014).

Besides focusing on an overall effect size, we further examined the extent to which variation in it could be explained by possible, moderating variables. Introducing these explanatory variables extended the three-level

random-effects model to a mixed-effects model (Cheung, 2015).

All analyses were conducted in the *R* package *metaSEM* (Cheung, 2015) based on Hedges' *g*, a standardized effect size representing the transfer effects.

3. RESULTS

This section presents (a) the overall transfer effect size, (b) effect sizes differentiated by types of transfer, (c) moderator analyses, and (d) analyses of publication bias.

3.1. Overall Effect Size

The three-level modeling approach resulted in an overall effect size of Hedges' $g = +0.49$, 95% CI = [0.37, 0.61], suggesting a moderate, positive, and statistically significant transfer effect of learning to code on cognitive skills. This effect size showed significant variation within studies ($\tau^2 = 0.20$, 95% CI = [0.16, 0.25]) and between studies ($\tau^2 = 0.28$, 95% CI = [0.17, 0.39]), suggesting the adequacy of the three-level approach. Moreover, the overall test of homogeneity indicated that effect sizes varied, $Q(538) = 2985.2$, $p < .001$.

3.2. Mixed-Effects Modeling

Given the evidence for significant variation of effect sizes across studies (see 3.1.), we further examined the extent to which selected study characteristics and the types of cognitive skills measures explained this variation. The resultant findings suggest possible moderation effects by cognitive skills measures.

3.2.1. Study Characteristics

Study design. Studies with a pretest-posttest control-treatment group design exhibited a slightly higher overall effect size ($g = +0.50$, 95% CI = [0.13, 0.90]) than studies with posttest-only designs ($g = +0.47$, 95% CI = [0.30, 0.65]). This difference, however, was statistically insignificant ($Z = 0.25$, $p = .80$).

Randomization. Studies performing a random assignment of participants to the experimental conditions exhibited larger transfer effects ($g = +0.56$, 95% CI = [0.16, 0.95]) than those without randomization ($g = +0.43$, 95% CI = [0.27, 0.59]); yet, this difference was not statistically significant ($Z = 1.04$, $p = .30$).

Other characteristics. Considering further study and sample characteristics, we did not find significant moderation effects by the

- Educational level of learners ranging from kindergarten to college/university;
- Type of coding language (i.e., visual vs. text-based languages);
- Intervention length (in hours);
- Coding context (i.e., coding embedded in the curriculum as part of regular school lessons vs. coding as an extra-curricular activity).

3.2.2. Cognitive Skills Measures

The overall effect for *coding skills* was $g = +0.75$ (95% CI = [0.39, 1.11]). The overall effect for skills other than coding was $g = +0.47$ (95% CI = [0.35, 0.59]).

Differentiating between different cognitive skills, we found positive and significant transfer effects on *creativity* ($g = +0.73$, 95% CI = [0.27, 1.20]), *reasoning* ($g = +0.37$, 95% CI = [0.23, 0.52]), and *mathematical skills* ($g = +0.57$, 95% CI = [0.34, 0.80]).

3.3. Publication Bias

To assess the presence of publication bias in the meta-analytic dataset, we took several steps (Borenstein et al., 2009):

- (1) *Trim-and-fill analyses*: No further study would have been needed on the left side of the outcome-standard error plot to achieve symmetry.
- (2) *Rosenberg's fail-safe N*: To achieve null effects, 134,706 additional studies with negative effects would have been needed. Given the size of this number, it seems unlikely that this many studies were not identified by our search protocol.
- (3) *P-curve*: The *P*-curve did not provide evidence for severe publication bias—a possible file-drawer effect is therefore unlikely.
- (4) *Moderation by publication type*: Comparing effect sizes between published studies ($k = 62$) and 'grey' literature ($k = 43$ effects, including dissertations and unpublished research reports) indicated significant effects favoring published studies, $Q_M(1) = 19.9$, $p < .001$. The transfer effect for published studies was $g = +0.53$, 95% CI = [0.31, 0.76]; for 'grey' literature, the effect was lower, $g = +0.25$, 95% CI = [0.15, 0.35]. This finding indicates some degree of publication bias in the data.

4. DISCUSSION

This meta-analysis tested the claim that learning how to code improves coding and other cognitive skills. To test these hypothesized transfer effects, experimental and quasi-experimental studies presenting computer coding interventions were synthesized. The aggregated transfer effect size was moderate, positive, and statistically significant ($g = +0.49$). Unlike existing discussions around the existence of transfer effects from specific domains of training (Sala & Gobet, 2017)—discussions that called into question the existence of such transfer effects and thus transfer of learning in general—the current study provides evidence that other cognitive skills may indeed benefit from coding instruction. This finding supports Liao's and Bright's (1991) early and Liao's (2000) later meta-analyses on the topic. Our explanation for this supportive finding lies in the very subskills coding requires: As Shute et al. (2017) note in their systematic review of computational thinking, the concept—which mainly comprises coding-relevant skills—represents a form of problem solving. Even further, the steps involved in coding (e.g., evaluating information, representing the problem, testing code or code elements systematically) align with current models of problem solving and even creativity (e.g., Scherer, 2016; OECD, 2014).

At the same time, our study showed that these transfer effects are not uniform across cognitive skills. We identified

stronger benefits for creativity and mathematical skills than for other skills (excluding coding skills themselves). These differential benefits may also be traced back to the subskills involved in them. In fact, there are differences between the processes creative thinking and, for example, mathematical thinking entail (e.g., Baer, 2015; Sak & Maker, 2006)—these differences may provide an explanation of this finding. To further explore alternative explanations, our future analyses target possible moderation effects of sample and study characteristics, including the content domains of the cognitive skills tests.

Overall, this meta-analysis contributes to the field of computational thinking in two ways: First, it provides evidence for the potential benefits of learning to code—an activity critical to the acquisition of computational thinking. This evidence substantiates existing claims surrounding the emphasis of coding skills. Second, it encourages researchers to take a differential perspective on the transferability of coding skills by considering *multiple* cognitive skills as possible outcome variables at the same time.

5. REFERENCES

- Baer, J. (2016). *Domain specificity of creativity*. New York, NY: Academic Press.
- Binkley, M., Erstad, O., Herman, J., Raizen, S., Ripley, M., Miller-Ricci, M., & Rumble, M. (2012). Defining Twenty-First Century Skills. In P. Griffin, B. McGaw, & E. Care (Eds.), *Assessment and Teaching of 21st Century Skills* (pp. 17-66). Dordrecht: Springer Netherlands.
- Borenstein, M., Hedges, L. V., Higgins, J. P., & Rothstein, H. R. (2009). *Introduction to meta-analysis*. Chichester, West Sussex: John Wiley & Sons, Ltd.
- Cheung, M. W.-L. (2015). *Meta-Analysis: A Structural Equation Modeling Approach*. Chichester, West Sussex: John Wiley & Sons, Ltd.
- Cheung, M. W. L. (2014). Modeling dependent effect sizes with three-level meta-analyses: A structural equation modeling approach. *Psychological Methods*, *19*(2), 211-229. doi:10.1037/a0032968
- Denning, P. J. (2010). Great Principles of Computing. *American Scientist*, *98*, 369-372. doi:10.1511/2010.86.369
- 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
- ICILS. (2018). *International Computer and Information Literacy Study*. Retrieved from: <http://www.iea.nl/icils> [12 January 2018]
- Liao, Y.-k. C. (2000). *A meta-analysis of computer programming on cognitive outcomes: An updated synthesis*. Paper presented at the Proceedings of world conference on educational multimedia, hypermedia and telecommunications.
- Liao, Y.-K. C., & Bright, G. W. (1991). Effects of Computer Programming on Cognitive Outcomes: A Meta-Analysis. *Journal of Educational Computing*

Research, 7(3), 251-268. doi:10.2190/e53g-hh8k-ajrr-k69m

- Moeyaert, M., Ugille, M., Natasha Beretvas, S., Ferron, J., Bunuan, R., & Van den Noortgate, W. (2017). Methods for dealing with multiple outcomes in meta-analysis: a comparison between averaging effect sizes, robust variance estimation and multilevel meta-analysis. *International Journal of Social Research Methodology*, 20(6), 559-572. doi:10.1080/13645579.2016.1252189
- OECD. (2014). *PISA 2012 Results: Creative Problem Solving: Students' Skills in Tackling Real-Life Problems* (Vol. V). Paris: OECD Publishing.
- Sak, U., & Maker, C. J. (2006). Developmental Variation in Children's Creative Mathematical Thinking as a Function of Schooling, Age, and Knowledge. *Creativity*

Research Journal, 18(3), 279-291.
doi:10.1207/s15326934crj1803_5

- Sala, G., & Gobet, F. (2017). Does Far Transfer Exist? Negative Evidence from Chess, Music, and Working Memory Training. *Current Directions in Psychological Science*, 26(6), 515-520. doi:10.1177/0963721417712760
- Scherer, R. (2016). Learning from the Past—The Need for Empirical Evidence on the Transfer Effects of Computer Programming Skills. *Frontiers in Psychology*, 7(1390). doi:10.3389/fpsyg.2016.01390
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142-158. doi:10.1016/j.edurev.2017.09.003

To Improve the Computational Thinking of Elementary School Students by Scaffolding

Chien-i LEE, Sheng-chuan CHUANG*, Shu-min WU

Department of Information and Learning Technology, National University of Tainan, Taiwan
leeci@mail.nutn.edu.tw, pengin0913@gmail.com, t10455101@stumail.nutn.edu.tw

ABSTRACT

MOE (2016) will bring Computational Thinking ability into the National Basic Curriculums in order to promote the students' problem solving ability is emphasized by many advanced countries. Although learning programming design is an important way to develop computational thinking. However, learning programming involves many abstract concepts of program syntax. It's hard for teachers to solve the problems in class one by one and provide individual guide which will result in poor learning aspiration and low learning achievement. Therefore, this study focused on providing a Scaffolding Guidance System during the process of solving problems and aimed to explore the effect of the system design on computational thinking. The study was bases on quasi-experimental design, and 48 students from two classes in an elementary school in Tainan. The 24 students in the experimental group were taught with the system design. The 24 students in control group were treated by traditional instructions. The experiment lasted for eight weeks and the data were analyzed with ANCOVA statistical method to explore the differences in learning efficiency between the system design and traditional instructions. The results showed that: (1) There were significant differences between the experimental group and the control group in learning efficiency; (2) After receiving the experimental teaching, the low level of student presented the most significantly different on computational thinking learning efficiency.

KEYWORDS

Computational Thinking, Visual Programming Language, Learning Efficiency, Portfolio

1. INTRODUCTION

In 2006, "CT" proposed by Wing won universal attention and recognition from many countries and scholars. In recent years, the connotation of various Computational Thinking has also been proposed and discussed by scholars, and gradually formed a consensus. (Wing, 2006; Wing, 2008) identified five core aspects of CT which are conditional logic, distributed processing, debugging, simulation and algorithm building. (Brennan et. al, 2012) use Scratch (designed by MIT Media Lab) -- a programming environment that enables young people to create their own interactive stories, games, and simulations, and then share those creations in an online community with other young programmers from around the world -- to develop a computational thinking framework: *computational concepts* (the concepts designers engage with as they program, such as iteration, parallelism, etc.), *computational practices* (the practices

designers develop as they engage with the concepts, such as debugging projects or remixing others' work), and *computational perspectives* (the perspectives designers form about the world around them and about themselves).

(Lahtinen et al., 2005) indicated that programming is not an easy subject to be studied. It requires correct understanding of abstract concepts. Many students have learning problems due to the nature of the subject. In addition, there are often not enough of resources and students suffer from a lack of personal instruction. Also the student groups are large and heterogeneous and thus it is difficult to design the instruction so that it would be beneficial for everyone. This often leads to high drop-out rates on programming courses in the universities. At present, there are many difficulties in teaching and learning activities of programming languages in the schools. In the process, they encounter complex syntax instructions, how to implement ideas in programming languages, and differences in student's level. (Robins et al., 2003; Lahtinen et al., 2005; Gomes & Mendes, 2007)

Programming itself is a highly logical thinking course different from the learning of package software. The current teaching methods will certainly not be able to meet the learning needs of each student. In addition, due to the constraints of classroom time, the lecturers did not have enough time to give individual guidance and provide immediately feedbacks about all students' questions. When students encounter difficulties, they often give up because they can't get the help. (Gomes & Mendes, 2007) Therefore, this study developed a "Scaffolding Guidance System" for primary schoolchildren. When students encounter learning difficulties of programming, the system will automatically provide the appropriate scaffolding guidance.

2. SYSTEM DESIGN

The Scaffolding Guidance System was built in Linux-based server, running Java-web-based application at Apache Tomcat, and recording portfolio with MySQL database. After logging with identity, the upper parts of interface are links of programming tasks (including flying bat, underwater world, monkey banana, whack-a-mole, and shooting game) and user information. Each task has a simulation animation on left side, and a main functional block on right side, including code-comparison analysis, project-code of user, and prompt of similar project-code. Figure 1 shows the main interface of system.



Figure 1. The main interface of system.

The purpose of the system is to parse the programming task of students, and to produce guided scaffolding to assist students learning in the system, which provide the following two functional modules: 1) parsing and prompts, 2) linking with experience. At coding time, students sometimes forgot or miss some vital blocks so that they could not accomplish the task. It's helpful that giving suitable prompts when students fall into troubles. The module of parsing and prompts will reach the aims that troubleshoot the above situations. This mechanism is set by the teacher about how many blocks to complete the task. When the critical blocks don't exist, what should students be prompted? Table 1 included below figures out critical blocks about the task.

Table 1. The fish of underwater world to prompt.

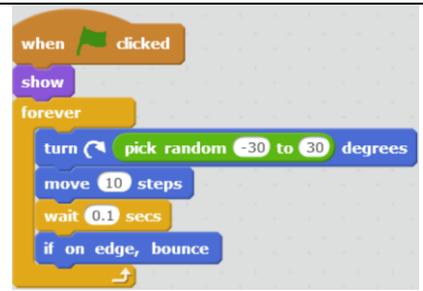
Agent & Necessary Blocks	Prompts While Missing Blocks
 	 <p>The fish to swim randomly in the seabed, it must be placed "turn right block" with "random parameter" and "move block" within a "forever block". Then the fish can swim around.</p>

Use Scratch's API (<https://wiki.scratch.mit.edu/wiki/JSON>) to render Scratch visualizer blocks into JSON-text-format, where each block is converted to a specific JSON list. Table 2 included below figures out how to map blocks to JSON.

Table 2. The mapping between blocks and JSON-text.

Agent	Scratch's API
	

Visualizer Blocks



```
JSON list  [ ["whenGreenFlag"],
["show"],
["doForever",
    [ ["turnRight:",
["randomFrom:to:", -30, 30]],
["forward:", 10],
["wait:elapsed:from:", 0.1],
["bounceOffEdge"]
]]]]
```

In additions, to link with experience of students, we use "Cosine Similarity" to judge similarity of two tasks. Because each task has several agents, we use agent as basic unit to compare code-similarity. Table 3 included below is for illustration of how to calculate similarity of agents.

Table 3. The similarity of agents.

Agent		
Blocks		
List JSON (Step1)	[whenGreenFlag, doForever, turnRight:, randomFrom:to:, forward:, wait:elapsed:from:, bounceOffEdge]	[whenGreenFlag, doForever, forward:, wait:elapsed:from:, nextCostume, turnRight:, randomFrom:to:, bounceOffEdge]
Combine (Step2)	[whenGreenFlag, doForever, turnRight:, randomFrom:to, forward:, wait:elapsed:from:, nextCostume ,bounceOffEdge]	

Vector	[A ₁ ,A ₂ ,A ₃ ,A ₄ ,A ₅ ,A ₆ ,	[B ₁ ,B ₂ ,B ₃ ,B ₄ ,B ₅ ,B ₆ ,
Transform	A ₇ ,A ₈]	B ₇ ,B ₈]
(Step3)	= [1,1,1,1,1,0,1]	= [1,1,1,1,1,1,1]
Calculate	$\text{similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$	
Similarity		
(Step4)	The value is greater than 0.8 mean that these two agents are similar.	

By comparing high similar codes such as similarity > 0.8, students can observe or practice similar tasks to discover the logic of their own programs.

3. EXPERIMENTAL DESIGN

This study designed a "Scaffolding Guidance System" to help schoolchildren of elementary to learn Scratch programming. When they got stuck on programming, system will provide suitable scaffolding guidance for them. Furthermore, investigating further to analyze the effect of visual programming and the influence of raising CT.

The quasi-experimental design was used in this study, which chose two five-grade classes of a primary school in southern Taiwan to participate this experiment. We random chose one class as the experimental group, and the other class as the control group. Students in the experimental group were enrolled in our purposed Scaffolding Guidance System into the Scratch-Programming course; Control group performed a traditional teaching method. Each group were taught a total of 8-weeks by the same teacher, each lesson 40 minutes, a total of 320 minutes. After the end of the programming course, we performed post-test: designing a computational practice of game. In Additions, to realize the responses of students in experimental group about using Scaffolding Guidance System, we performed a semi-structured interviews with two-groups students separated to low, middle, and high level respectively according their previous-semester grade.

About the instruction design, the teacher conduct the operation of Scratch interface at first-two weeks, and then the students of two groups have to implement five-tasks programming-design in the next six weeks respectively. The experimental-group students will use Scaffolding Guidance System to learn programming: viewing the animation about the tasks first, then decomposing the problems and describing the features of each role in Scratch, and finally coding. When they got stuck in programming, system would give them assistance. For the control-group students, teacher use traditional instruction.

Students involved in the experimental group must complete the "flying bat" and other five scaffolding guided program tasks, which are based on (Brennan et al, 2012) proposing CT framework including these two dimensions: "computational concept" and "computational practice". Finally, to assess the performance of CT, this study used game scenarios to test students' ability to practice. In this

game scenario there are two game agents (parrots and obstacles, as Figures 2) and a stage design. There are also having procedural issues in the agents and the stage. For example, in the agent of obstacle, students are required to use program blocks to solve the problem of "obstacle generation and movement".

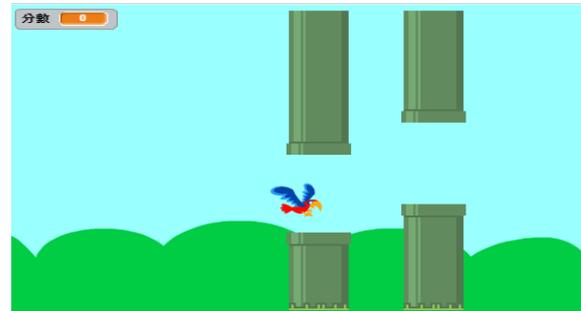


Figure 2. The evaluation of CT through practicing a game.

4. RESULTS AND DISCUSSION

Table 4 shows that there was no significant difference of Group times Grades between two groups, that is, we can accept the null hypothesis and that meets the condition for homogeneity of regression so that we can continue to do ANCOVA.

From Table 5, the results of ANCOVA between experimental and control group showed that F = 7.062, p = .011 < .05 reached significant difference. That is, after adopting different pedagogical methods to conduct experiments, the results of CT test of students in experimental group and control group reached significant differences, indicating that accepting the activities with "Scaffolding Guidance System" have significant improvement.

Table 4. The tests for homogeneity of regression.

Sources	Type(III)) SS	df	Mean Squar e	F	p
Group*Grade s	333.6	1	333.6	1.25 2	.26 9
Error	11729.6	4 4	266.5		

$p^* < 0.05$, $p^{**} < 0.01$

Table 5. The ANCOVA of two groups.

Group	Mean	SD	N	F	p
Experimental	59.9583	16.10692	24	7.062	.011*
Control	42.7083	24.14536	24		

$p^* < 0.05$, $p^{**} < 0.01$

To further understand the impact of the scaffolding guidance system for students of different levels, the students in two groups separated to low, middle, and high level respectively according their previous-semester grade. Only low level about ANOCVA achieved statistically significant (as Table 6), which indicated that

accepting the activities with "Scaffolding Guidance System" of low level have significant improvement.

Table 6. The ANCOVA of two groups about low level.

Group	Mean	SD	n	F	p
Experimental	47.5000	12.29402	8	8.437	.012*
Control	24.1250	15.81534	8		

$p^* < 0.05$, $p^{**} < 0.01$

Finally, by semi-structured interviews with two-groups students and questions about statistic of missing blocks of five-tasks, we found that some blocks learned on the previous task, but when in a different scenario or agent, students still need to be prompted to complete the task. This phenomenon is similar to that of (Gomes & Mendes, 2007; Robins, Rountree, & Rountree, 2003): "Students are often confined to the surface knowledge of programs and can't apply what they have learned to new problems."

5. CONCLUSIONS

This study from the CT and learning effectiveness, different levels of students, and learning portfolio to discuss the following conclusions:

First, students who accepted the teaching activities of "Scaffolding Guidance System" performed better than the ones with traditional teaching. Secondary, for students of low level achievement, this study provided an approach to assistant them by using "Scaffolding Guidance System". Finally, teachers can analyze the portfolio of students to discover the learning problems that can't be found from the surface information. For the future works, researcher

can be directed towards the fields of automation of system and adaption of students.

6. REFERENCES

- Brennan, K., & Resnick, M. (2012, April). *New frameworks for studying and assessing the development of computational thinking*. In Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada (pp. 1-25).
- Code.org. (2017). *Computational Thinking*. Retrieved from <https://studio.code.org/s/course3/stage/1/puzzle/1>
- Gomes, A., & Mendes, A. J. (2007). *Learning to program-difficulties and solutions*. Paper presented at the International Conference on Engineering Education–ICEE.
- Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). *A study of the difficulties of novice programmers*. Paper presented at the ACM SIGCSE Bulletin.
- MIT Media Lab, *Scratch*. <https://scratch.mit.edu>
- MOE (2016). 2016-2020 General Information Education Blueprint. Taipei City: Ministry of Education.
- Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and teaching programming: A review and discussion*. Computer science education, 13(2), 137-172.
- Wing, J. (2006). *Computational thinking*. Communications of the ACM, 49(3), 33-35.
- Wing, J. (2008). *Computational thinking and thinking about computing*. Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences, 366(1881), 3717-3725.

A Curriculum and Contents of Programming Education for Computational Thinking

Hyojin BYUN¹, Miyoung RYU², Sungwan HAN^{2*}

¹Dept. of Steam Education, Gyeongin National University of Education, Korea

²Dept. of Computer Education, Gyeongin National University of Education, Korea
mydream.bhj@gmail.com, ddochi29@naver.com, han@gin.ac.kr

ABSTRACT

Computational thinking is emerging as a core competency for creative and efficient resolution of complex problems in a rapidly changing society. In Korea, software education is introduced into the 2015 revision curriculum and emphasizes creative problem solving process through CT and programming learning. In this study, Scratch was selected with an educational programming language suitable for use in elementary school, and programming curriculum for improving CT was developed and tested by expert group.

KEYWORDS

Educational Programming Language, Curriculum and Content, Computational Thinking, Programming Education

1. INTRODUCTION

As the role of SW in modern society grows, the necessity of strengthening SW competency is emphasized. SW is recognized as a means of solving problems related to human higher thinking ability beyond SW functional aspect. As a result, CT is attracting attention as a core competence for solving various complex problems in the future.

CT is to define a problem from the viewpoint of computing, to search for the solution to the problem, and to a resolve the problem through efficient resolution procedures.

In Korea, awareness that computational thinking is the core competency of the future, the contents of the existing information-related curriculum were reorganized into software education contents through the 2015 revision curriculum.

Therefore, this study aims to develop and present contents for software education using Scratch in order to acquire CT through programming and to develop creative problem solving ability based on it

2. THEORETICAL BACKGROUND

2.1. Programming Education

Programming is a technique for implementing an abstract algorithm in a specific computer program using a specific programming language.

In the elementary school, the direction of programming education is to enhance the thinking ability of the learner's logical thinking ability, creative thinking ability and problem solving ability.

2.2. EPL and Scratch

The programming language to be used in elementary school software education should be a visual environment in which the expression of grammar and algorithm should be simple.

Scratch is a language designed for programming experience for children ages 8 to 16. The feature is that it is easy to learn the programming language itself with a simple grammar, a block-stacking algorithmic representation, and a variety of multimedia such as graphics and sound.

2.3. SW education in Korea

In Korea, the term 'SW education' was used in the 2015 revision curriculum, and the software education was made mandatory for elementary and junior high school students from 2018. In the 2015 revised curriculum, elementary SW education emphasizes real-life problem solving based on information ethics and attitude as a field within practical subject for 17 hours a year.

3. DEVELOPMENT OF EPL CURRICULUM AND CONTENT

3.1. Curriculum Development Procedures

The EPL curriculum to improve CT was developed through the steps shown in Table 1.

Table 1. Procedures of Curriculum Development

Analysis	<ul style="list-style-type: none"> ·CT concept ·Software education direction required at elementary level ·Pre-EPL program study
↓	
Design	<ul style="list-style-type: none"> ·Extract curriculum components ·Step-by-step learning topic and content selection
↓	
Development	<ul style="list-style-type: none"> ·EPL content composition and development
↓	
Verification	<ul style="list-style-type: none"> ·Conduct validation of the curriculum and EPL contents for experts

3.2. Development of EPL Curriculum

In this study, the programming curriculum using Scratch was designed as shown in Table 2 to improve CT of elementary school students.

The elements of the CT concept consisted of sequences, loops, parallelism, events, conditionals, operators, and data using Brennan and Resnick's CT evaluation framework. The execution elements are also composed of incremental and iterative, testing and debugging, reusing and remixing, and abstracting and modularizing.

The subject was designed to allow students to access each category of Scratch sequentially, but to be as close as possible to the real life.

The learning stage was divided into three stages and the difficulty level of the learning was adjusted so as to have hierarchy of learning step by step. Each stage was composed of six phases and gradually expanding the command category of the Scratch related to CT.

Table 2. Presentation of EPL Curriculum

Level	Period	Topic	CT Concepts	Block categories	Contents of learning activity
1	1	Driving		E, M	Controlling car motion with specific keys
	2	Hide and Appear		E, M, L, C	Controlling characters with hide and show blocks
	3	Drawing shapes		E, M, P, C	Draw a shape using a pen
	4	Dancing		E, M, L, S, C	Show your dancing to your performance
	5	Catching insects		E, M, L, S, C, S, O	Use random numbers to follow random characters
	6	Paint		E, L, P, C, S	Create Paint with multiple colors as a condition
2	1	Send and receive a conversation		E, L, C	Conversation using broadcasting block
	2	animation		E, M, L, S, C, M	Representing animation effects with additional blocks
	3	Jump		E, M, D, C, S, O	Express if the condition is satisfied.
	4	Rock Paper Scissors		E, L, D, O, C	Change the shape using the value of a variable
	5	Compare the size of a number		E, L, D, O, C	Using List to Compare Numbers
	6	Running race		E, M, L, D, O, C, S	Display two levels of difficulty with two characters running
3	1	Falling apples		E, M, C, S, O	Expressing how fast you move using variables and timer
	2	clock		E, M, C, S, O	Clock representation using current time block
	3	Put a soccer ball in the goal		E, M, C, S	Using a video sensing block to move the ball
	4	Making pattern		E, M, P, D, C, O	Create patterns using variables x and y
	5	My body grows.		E, M, D, C	Use cloning blocks to express more and more appearances
	6	Walk to goal		E, M, D, C, S, O	Express sprite movement using background motion

CT Concepts						
sequences	loops	parallelism	events	conditionals	operators	data

Block Categories				
M	L	S	P	D
Motion	Looks	Sounds	Pen	Data
E	C	S	O	M
Events	Control	Sensing	Operators	More Blocks

3.3. Development of EPL Content

Students will experience Brennan and Resnick's practice exercises through CT Opening, CT Raising, and CT Experimentation so that they can expand their CT.

In CT Opening, students use example files to identify and explore the situation. In CT Raising, students learn basic contents while learning programming step by step, and expand the project by using reuse and remixing to CT Experimentation.

Table 3. Example content

Level 1 – 1 st period	
Topic	Driving
Activity Goals	Let's move the car using the motion block.
CT	Sequences, events
Plan specific activities	
Step	Teaching and Learning Activities
CT Opening	·Using the example file to understand the content ·Explore blocks in motion categories
CT Raising	·Think of a situation where you move a set value by pressing a direction key (up, down, left, and right) through a question. ·Experiment the script and check it.  ·Complete the script so the car can move in four directions by itself
CT Experimentation	·Draw a road with Paint, then write a script to allow the car to move over the road to reach its destination ·[Optional Activities] Parking in the parking lot in reverse

3.4. Expert Validity Testing

Groups participated in this study were selected from a field related to education professionals who have experience of teaching the EPL. The results of the CVR test are shown in

Table 4, and the validity of the total items satisfies the minimum value of .62 according to 10 panelists. Therefore, it can be said that the content validity is secured according to the curriculum contents and the flow of the example contents.

Table 4. Expert Review Results

Division		CVR
Curriculum development direction		1
Learning level		.9
Learning sequence	Programming	.9
	CT	1
Learning contents	Topic	.8
	CT	.9
Learning method	Programming	1
	CT	.9

4. DISCUSSIONS

As part of software education around the world, there is a strong interest in coding education, and in 2018, software education is mandatory in Korea. This is to enable students to cultivate CT through SW and to efficiently solve various complex and unexpected problems of the future society.

This study selected a Scratch as an educational programming language suitable for elementary level, and programmed it so that CT can be extended through programming education. 17 hours allocated as regular curriculum hours are planned to achieve the goal of software education by including content other than programming. So, it is difficult for the programming education to expand the CT within the regular course time.

Therefore, the program developed in this study proposes a method to secure and apply sufficient time through club activities, after - school activities, camps, gifted education, etc.

5. REFERENCES

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Paper presented at annual American Educational Research Association meeting*, Vancouver, BC, Canada.

C. H. Lee (2015). Direction and Model of Software Education in Elementary Education, *Journal of Korean Practical Arts Education*, 28(4), 207-222.

E. H. Lee, T. W. Lee (2015). Instruction Model for Elementary School on Programming Induction Education Using ENTRY, *Journal of Korean Association of Computer Education*, 19(1), 43-46.

J. H. Seo, Y. S. Kim (2016). Development and Application of Educational Contents for Software Education based on the Integrative Production for Increasing the IT Competence of Elementary Students, *Journal of Korean Association of Computer Education*, 20(4), 357-366.

W. S. Moon (2015). The Application of the Scratch2.0 and the Sensor Board to the Programming Education of Elementary School, *Journal of Korean Association of Information Education*, 19(1), 149-158.

Comparing with Scratch and Python in CT Concepts

Tae-ryeong KIM, Sun-gwan HAN

Dept. of STEAM Education, Gyeong-in National University of Education
crossallover@gmail.com, han@gin.ac.kr

ABSTRACT

The expansion of software education has given learners the opportunity to learn CT concepts related to CS through block programming such as Scratch. However, due to the nature of EPL, the concept of computer science is limited, and inevitably the text programming language is learned to expand CS thinking. In this paper, we will examine the possibility of using the concept of prior learning after block programming tools through comparison of basic grammar examples of Scratch and Python in terms of CT concepts.

KEYWORDS

EPL, TPL, Scratch, Python, CT Concepts

1. INTRODUCTION

Extensive expansion of EPL (Educational Programming Language) education has resulted in many students improving their CT competency and related areas. In this area, Brennan & Resnick (2012) divided the CT into three dimensions by analyzing the results of the Scratch outputs made by the students. One of them was CT concepts. These concepts that can be transmitted in other programming languages, and it is common in programming languages as well. If student experience a certain level of EPL training, they will inevitably go to TPL (Text based Programming Language) to improve their programming skills (Jun, 2012). Therefore, From the perspective that transfer mechanism (Schwartz & Bransford, 1998), we want to create an opportunity to summarize these concepts as TPL and to utilize the student's prior knowledge on related concepts. An example of TPL is Python, which has a high educational potential among text languages (Grandell, 2006).

2. COMPARISON

2.1. Sequences

In the Scratch, the sequence of blocks directs the operation of the object (sprite), so the concept of sequence can be learned without difficulty (Elkin et al, 2014). Likewise, Python is well suited for students to learn sequence concept because grammar itself is not only a direct language, it also provides immediate and visual information as interpreted language (Yeum, 2008). The sequence concept can be easily transmitted in text language like Figure 1.



Figure 1. Comparison in Sequences Concept

2.2. Loops

In the sequence concept, the principle of efficiency leads to repetition naturally. Instead of using many blocks one by one, students can easily configure the program with several blocks. Python can easily configure bound loops and conditional loops too. In particular, it has the advantage of being able to configure the iterators that make circuit of the data, as shown in Figure 2.

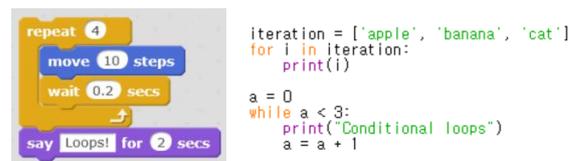


Figure 2. Comparison in Loops Concept

2.3. Events

Because Scratch is also intended to interact with the user, it uses event-driven programming. And this is a fun factor for learners. So, Scratch supports various event handlers. While Python's shell itself functions as an interactive mode with the user, creating an interactive program is possible a little later than the order of learning in Scratch. because It needs to learn how to use functions and libraries in order to create a practical program with events. Figure 3 shows how the basic library handles keyboard events.

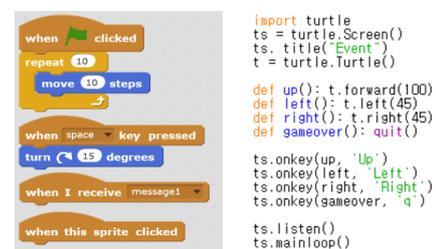


Figure 3. Comparison in producing Event

2.4. Parallelism

Similarly, the use of parallelism concept is easier in event-based programming languages. In scratch, it is possible to experience the parallel form simply by generating the event several times. However, in the interpreted language, It's not efficient. Python supports a module that handles different types of threads in being, as shown in Figure 4. It is only an example of a low-level representation of related concepts.



Figure 4. Comparison in Parallelism concept

2.5. Conditionals

Because complex algorithms can present difficulties for students, Scratch provides a various conditional block that can be combined with repetitive structure or event monitoring, operators, sensors, etc., As shown below (Dasgupta et al, 2016). In text programming languages, students can learn conditional grammars without difficulty (Milne & Rowe, 2002). In view, scratch is more configurable, Figure 5 shows that basic structure is similar.

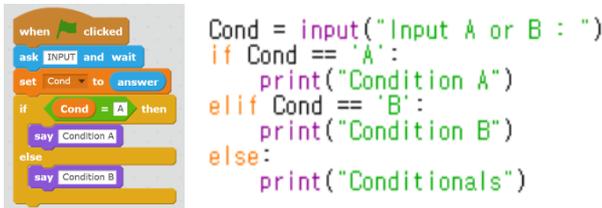


Figure 5. Comparison in Conditionals Concept

2.6. Operators

Scratch contains arithmetic (including character) operators, relational operators, and logical operators, which can be combined in various ways depending on the needs of the learner. Surely, commercial languages generally support all sorts of operations on operators. Especially in Python, almost all operator parts are easier to use because they are grammatically simpler than other text languages. If doing a number of complicated calculations, the text language can be configured more quickly and easily, if you are familiar with the grammar, as shown in Figure 6,

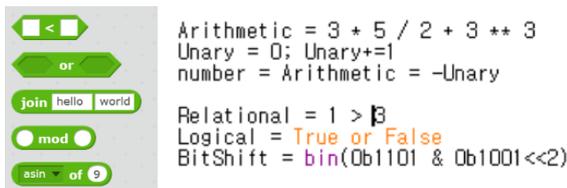


Figure 6. Comparison in Operators concept

2.7. Data

Scratch provides variable and list data types. In most of the block based programming, variables are used to implement the scoring function. Also, there is no need to define data types, which is one of the hardest parts of the students (Piteira & Costa, 2013). In Python, Because Python is a dynamic type, students do not need to set the data type like Scratch. Thus, Scratch learners can easily learn this. It's also easier to handle data than any other text language (Rashed & Ahsan, 2012). Figure 7 is one way to define and manipulate data types.

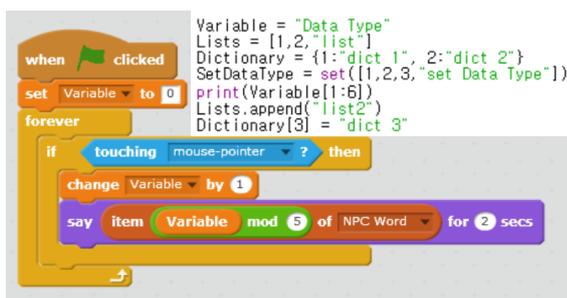


Figure 7. Comparison in Data concept

3. DISCUSSION

I compared Scratch with the Python language, focusing on the seven concepts that can be found in Scratch. As a result, it can be seen that the text language can also be structured easily in terms of Sequences, Loops, Conditionals, Operators, and Data. However, in terms of Events and Parallelism, It's hard to using precedence concepts due to difference in complexity between EPL with TPL.

Therefore, we propose to use the related computer science concepts learned in the Scratch as a precedent organizer form when continuing the learning through the text programming language education course.

4. REFERENCES

- Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association*, Vancouver, Canada (pp. 1-25).
- Dasgupta, S., Hale, W., Monroy-Hernández, A., & Hill, B. M. (2016). Remixing as a pathway to computational thinking. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing* (pp. 1438-1449). ACM.
- Elkin, M., Sullivan, A., & Bers, M. U. (2014). Implementing a robotics curriculum in an early childhood Montessori classroom. *Journal of Information Technology Education: Innovations in Practice*, 13, 153-169.
- Grandell, L., Peltomäki, M., Back, R. J., & Salakoski, T. (2006, January). Why complicate things?: introducing programming in high school using Python. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 71-80). Australian Computer Society, Inc.
- Jun. W. C. (2012). A Study on Correlation Analysis of EPL and Programming Ability for the Gifted Children in IT. *Journal of The Korean Association of Information Education*, 16(3), 353-361.
- Milne, I., & Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1), 55-66.
- Piteira, M., & Costa, C. (2013, July). Learning computer programming: study of difficulties in learning programming. In *Proceedings of the 2013 International Conference on Information Systems and Design of Communication* (pp. 75-80). ACM.
- Rashed, M. G., & Ahsan, R. (2012). Python in computational science: applications and possibilities. *International Journal of Computer Applications*, 46(20), 26-30.
- Schwartz, D. L., & Bransford, J. D. (1998). A time for telling. *Cognition and instruction*, 16(4), 475-5223.
- Yeum. Y. C. (2008). Programming Learning Environment using a Textual EPL in Informatics Education. *doctoral thesis*. Korea University.

Computational Thinking and Subject Learning and Teaching in K-12

Students' Attitude Changes through Integrating Computational Thinking into English Dialogue Learning

Xiaojing WENG

The University of Hong Kong, Hong Kong
u3532170@connect.hku.hk

ABSTRACT

Computational Thinking (CT), a problem-solving skill rooted from Computer Science (CS), is gaining attention from computer scientists and K-12 educators increasingly. Language communication skill, in the meanwhile, is one essential skill developed through the K-12 education, which will continue to attract attention in the foreseeable future globally. There is the possibility for students to learn an effective communication skill while using or improving CT, given that the interdisciplinary work of integrating language learning into a CT learning activity has not been fully discussed in existing literature, this preliminary study, which is going to be extended to a largescale work in the future, is important from the perspective of both language and computer teachers. As an initial step, this research attempted to obtain insights on developing CT in the context of English dialogue learning by investigating the attitude changes of students after they have experienced the integration of CT into English education in Scratch programming environment. To achieve this objective, pre/post-lesson surveys were distributed to nine Hong Kong primary students who attended the intervention class to study computer programming by using the graphical programming language Scratch. The preliminary results show that primary school students have positive reaction to the introduction of CT into English dialogue learning through graphical programming language. Students were more motivated to learn English dialogue after the class under study; however, their attitude towards learning graphical programming language become less positive after the intervention.

KEYWORDS

Computational Thinking, English dialogue learning, Scratch, K-12

1. INTRODUCTION

The idea of computing, which refers to all the activities that require, benefit from, or create computers (Shackelford et al., 2006), first gained attention from the public as a result of Seymour Papert's work in MIT in the 1980s (Lockwood & Mooney, 2017). However, the concept of CT became increasingly popular ever since it was refined by Jeannette M. Wing in 2006 (Grover & Pea, 2013). Wing argues that CT is a universal attitude and skill that can be applied by everyone; it is not limited to computer scientists (Wing, 2006).

A huge proportion of CT development programs for young students in schools, colleges or afterschool clubs have

been conducted in the context of CS subject. This is mainly because improving students' problem-solving thinking skills and learning programming are the major elements of CS course (Lockwood and Mooney 2017). However, this seems too limiting. The ability to think computationally has the potential to benefit students in all courses. Furthermore, while problem-solving skills and programming are perhaps the most direct approaches to cultivating CT ability, they are not the only important elements in CS. There are also still education objective confusions as well as disagreements on learning content and the issue of whether CS should be a compulsory subject in the K-12 curriculum (Armoni, 2013; Hubwieser, 2012). Taking into account these considerations, it becomes obvious that CT should and can go further than to be constrained to computing-related subjects.

Therefore, many researchers have been exploring how CT can be integrated into other subjects, including Non-CS STEM (which refers to four subjects including Science, Technology, Engineering and Mathematics) subjects and the humanities (Kafai & Burke, 2013; Lee, Martin, & Apone, 2014; Lye & Koh, 2014). As a matter of fact, language arts can be used as a springboard for the integration of CT into the K-12 curricula, like what has been proposed by Barr and Stephenson—computational skills such as abstraction, algorithm, automation and decomposition can be applied or enhanced when students are using rhetorical devices, writing instructions, conducting story reenactments or planning an outline for a composition in a language class.

Many researchers connecting CT to English start their work by utilizing models found in writing-related workshops like composition, journalism, literature or poetry (Burke & Kafai, 2012; Nesiba, Pontelli, & Staley, 2015; Wolz, Stone, Pearson, Pulimood, & Switzer, 2011). This strategy is reasonable because writing for programs is coding in CS, and since writing and coding are both types of expression but with different carriers, young people can come to learn the significance of sequence, structure and clarity of expression (Burke & Kafai, 2012) from both of them. It is inspiring to see that there are many positive outcomes of these practices in terms of students' perspective towards CT; however, the depth and breadth of this infusion in the context of English can be extended further.

Though it seems that no researcher has specifically conducted an experiment exploring CT ability in English dialogue learning, the literature on English dialogue is insightful in showing us the possible ways in which English dialogue learning could embrace CT. For example, the literature on Second Language Acquisition (SLA) indicates that when exposed to questions and answers in conversations, people understand how different parts of a sentence works as a unit and can master the vocabularies at

the same time (Hatch, 1978). Furthermore, students can be more creative when they are engaging with topics in an open-ended, free manner instead of planning the conversation ahead of time (Andersen, 1983). Besides these benefits, many other elements of CT can also be employed in English dialogue learning (as shown in figure 1).

CT Concept & Capability	Language Arts	English Dialogue
Data collection	Do linguistic analysis of sentences	Receive information
Data analysis	Identify patterns for different sentence types	Understand information
Data representation	Represent patterns of different sentence types	Express idea
Problem Decomposition	Write an outline	Listening and thinking
Abstraction	Use of simile and metaphor; write a story with branches	Define topic
Algorithms & procedures	Write instructions	Take turns
Automation	Use a spell checker	Language checking Apps
Parallelization		Digital storytelling
Simulation	Do a re-enactment from a story	Syntax and vocabulary
Creativity		Open-ended answers

Figure 1. Bridging CT, Language Arts (Barr & Stephenson, 2011) and English dialogue learning.

Our research closely connects CT with the graphical programming language Scratch in order to give answers to the following questions:

RQ1: What attitude do students hold towards using graphical programming language in English dialogue learning?

RQ2&3: Are there attitude changes of students towards both English dialogue learning and graphical programming language learning after students experience the integration of graphical programming language in English dialogue learning?

The paper will then be organized as follows. In Section 2, the research methodology will be introduced, data collection and data analysis will be presented in Section 3, in Section 4 results of the research will be given, discussion of this research will be presented in Section 5, and future research fields in infusing CT into English Education are suggested in Section 6.

2. METHODOLOGY

2.1. Constructionism as the Theoretical Framework

Constructionism describes the process of gaining knowledge as “building knowledge structures” (Papert, 1991). Among many renowned scholars in constructionism, Papert is one of the most significant representative figures in this school of thought. He stresses that people gain new knowledge by engaging in doing and making artifacts, no matter what kind of the learning circumstances and working entities. From this perspective, constructionism focuses more on people’s personal conversation with their own representations, projects and products rather than the general developmental rules (Tokoro & Steels, 2004).

According to constructionism, the participants of this study are assigned to finish a digital artifact individually by adopting the graphical programming tool Scratch. This is an example of constructionism practice since by programming in Scratch students are practitioners of the constructionism principle -- “learning by making”. In this way, students can build their CT and English language knowledge structures.

The one-session intervention class was designed under the guidance of constructionism as shown in figure 2:

Time/Mins	Teaching Activity
5	Guide students to review what they have learned in the last Scratch class, remind students of their knowledge of character sets, dialogue and sequence operations in Scratch.
5	1. Classroom leading-in. Little Squirrel Scrat is studying Scratch and English in his class. One day, his English teacher left a homework and asked them to present a set of English dialogues they have learned. His teacher promised the best presenter can get an acorn as a reward, how can Scrat obtain the reward? 2. Ask students to think about the possible ways for Scrat to display the English dialogues (perform short plays, make small videos, etc.). 3. Lead to the idea that students can use Scratch as a tool to display English dialogues.
10	Task 1: 1. 3 to 4 students form a learning group, and each group needs to write a set of English dialogues. English dialogue requirements: Between the two roles; No less than two rounds. 2. Modify the English dialogues between different groups, then the teacher confirms the dialogues and process to the next step. Dialogue checking criteria: No spelling mistakes; No grammatical errors; Meet the dialogue requirements set out in task 1.
30	Task 2: 1. Open the edit area and demonstrate how to create a character by using Scratch’s role sample library. 2. Demonstrate how to use Scratch’s background sample library as background to change the stage background. 3. Demonstrate how to drag and drop program modules into the programming area to implement conditional expressions and dialogue / question functions.
10	Task 3: Invite students to show their works through auto-play and look inside the projects, then encourage them to discuss how to improve their creations.

Figure 2. Lesson Design.

2.2. Scratch as the CT tool

As a graphical programming language, Scratch is a popular product of the Lifelong Kindergarten Group at the MIT Media Lab. It provides the platform for young children from 8 to 16 to program different forms of projects, including stories, games and animations (Resnick et al., 2009). Research has concluded that Scratch can improve students’ creativity, study outcomes and problem-solving abilities (Chang, 2014), therefore it is well accepted by the public. In 2015, Scratch welcomed its tenth birthday with more than 3,500,000 users and more than 6,000,000 shared projects (Moreno-León & Robles, 2015) from over 150 different countries and in more than 40 languages.

CT was defined as a three-dimensional framework by Brennan and Resnick with respect to Scratch (Brennan & Resnick, 2012), this framework suggests understanding CT from different angles, including computational concepts (sequences, loops, events and so on), computational practices (experimenting and iterating, testing and debugging, reusing and remixing, etc.) and computational perspectives (expressing, connecting, questioning) (Resnick et al., 2009). With this kind of supporting theories, therefore, Scratch was employed as the CT instrument to facilitate researchers’ research design (Burke & Kafai, 2012; Moreno-León & Robles, 2015; Holt, 2011; Meerbaum-Salant, Armoni, & Ben-Ari, 2013).

Gaining experience from reviewing other experimental research, the researcher in this study chose to take the graphical programming language Scratch as the CT tool as well.

An example of the artifacts in Scratch is shown in figure 3 and figure 4.



Figure 3. Project Example (Scratch visuals).

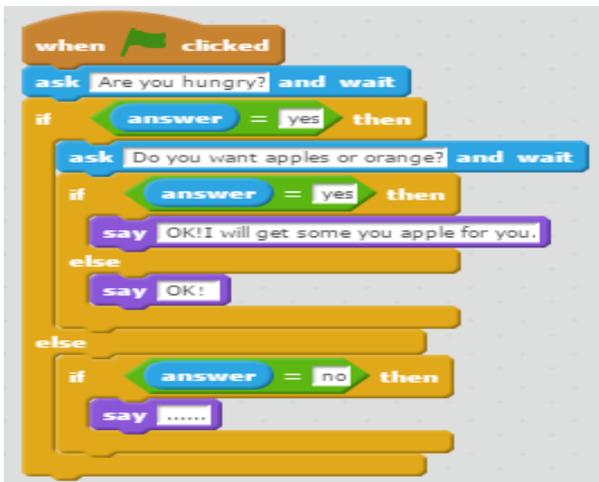


Figure 4. Project Example (Scratch coding blocks).

2.3. The Attitude Tests

In this research, the attitude tests were adapted from the 3-TUM (Three-Tier Technology Use Model) by Shu-Sheng Liaw (see figure 5) to investigate user perceptions toward information and Internet technologies. According to the 3-TUM, there are three different tiers for evaluating attitudes toward information technology: the tier of individual experience and system quality, the tier of affect and cognition, and the tier of behavioral intention (Liaw, Huang, & Chen, 2007). Therefore, the pre/post surveys cover questions addressing students' personal experiences, affects and behaviors in terms of English dialogue learning and Scratch learning. The surveys were designed in this study by using the five-point Likert scale in which respondents are asked to evaluate each statement by choosing a number from one to five, where 5 = Strongly Agree, 4 = Agree, 3 = Neutral, 2 = Disagree, 1 = Strongly Disagree.

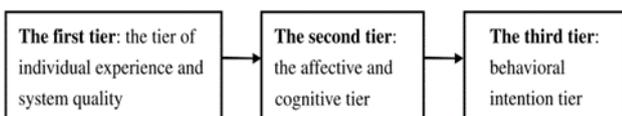


Figure 5. The three-tier use model (3-TUM).

However, there are minor differences in the pre-test and the post-test -- besides exploring students' attitudes towards English dialogue learning and Scratch learning, the post-test also explored students' attitudes towards the CT-infusing class (as illustrated in figure 6).

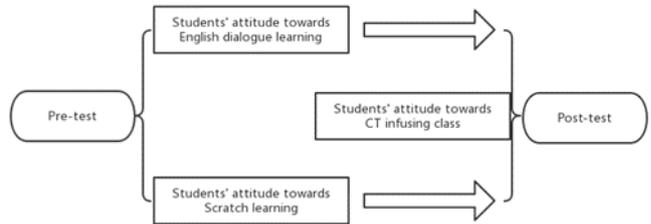


Figure 6. Differences in Pre-test and Post-test.

3. DATA COLLECTION AND ANALYSIS

3.1. Participants

A local aided whole-day co-educational primary school in Hong Kong agreed to participate in this research from February to July 2017. Students in this school can have an extra-curricular interest-oriented CS class weekly. In this class, students from grade one to grade three will start learning elementary computer operations; for students in grade four and above, the CS teacher adopts graphical programming platform Scratch (Resnick et al., 2009) to teach them how to program. There were 9 students in the interest-oriented class took part in this research, their gender, age and grade information are shown in figure 7.

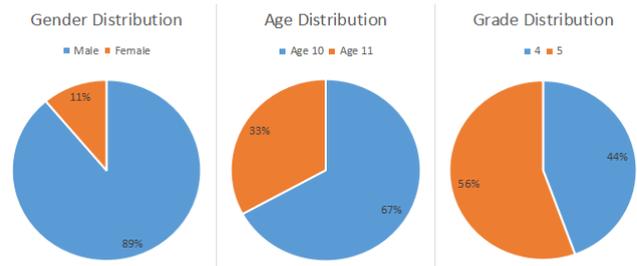


Figure 7. The gender, age and grade distribution of students.

3.2. Data Collection Process

The researcher reserved fifteen minutes with all the students taking part in this study before distributing any questionnaires. During this period, the researcher introduced the research objectives, background, and process to all the participants. Brief information about the pre-test and the post-test was provided, and students' rights as research participants were described as well. Students were then given enough time to finish the pre-test before the intervention, and the same length of time was offered to students for the post-test after the intervention. Qualitative interviews will be conducted to gain further insights in the future as the next step of our research.

3.3. Data analysis

Both the pre-test and the post-test data were gathered in Microsoft Excel to provide an overview of the research results. The data were then compared to see if the CT-infusing class caused any attitude changes among students.

4. RESULTS

As introduced in the methodology, the surveys were designed by using the five-point Likert scale. Students needed to evaluate each statement in the survey by choosing the strength of their agreement from 1 to 5, therefore each item got a total mark ranging from 5 to 45 points based on 9 students' responses.

4.1. Pre-test and Post-test Results for RQ 1

RQ1: What attitude do students hold towards using graphical programming language in English dialogue learning?

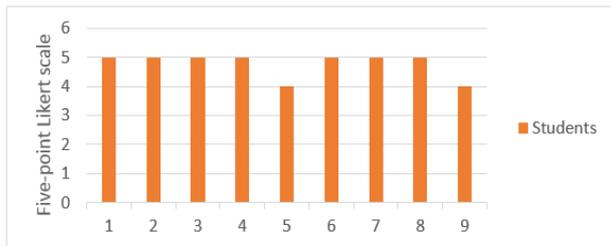


Figure 8. Student responses to the statement “I think Scratch helped me create my English dialogues.”

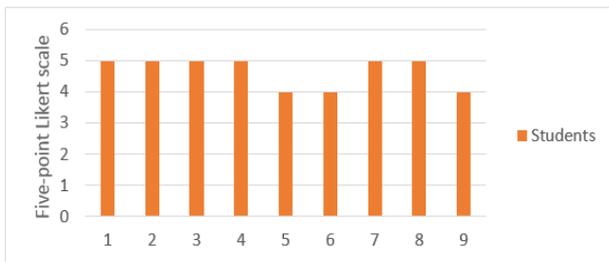


Figure 9. Student responses to the statement “I enjoyed the experience of learning English dialogue with Scratch.”

According to the collected answers from the questionnaire issued after the intervention, 78% of the students strongly agreed that Scratch helped them create English dialogues. As shown in figure 8, all the students held a positive perception towards the role of Scratch in their English dialogue learning. Furthermore, 67% of the students strongly agreed that they enjoyed the experience of learning English dialogue with Scratch. No student gave neutral or negative feedback about the experimental class experience (see figure 9). Thus, it is apparent that students held a positive attitude towards using graphical programming language in English dialogue learning.

4.2. Pre-test and Post-test Results for RQ 2

RQ2: Do students' attitudes towards English dialogue learning change after students experience a class in which graphical programming language is infused in English dialogue learning?

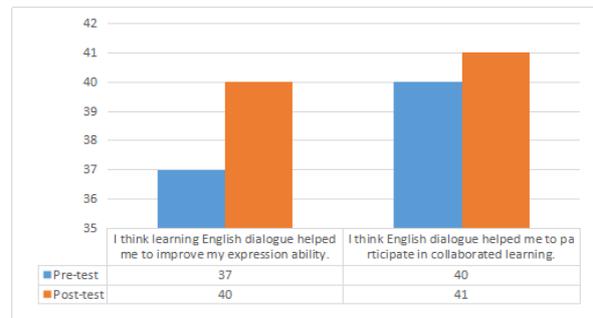


Figure 10. Total Likert score across students for the first tier of individual's attitudes (individual experience).

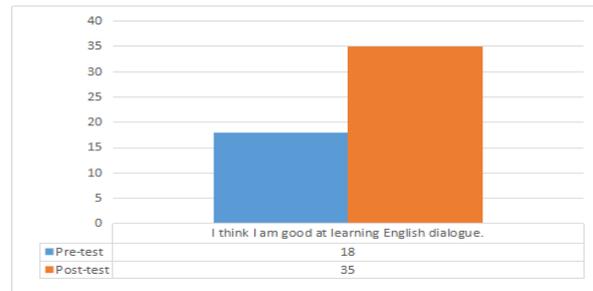


Figure 11. Total Likert score across students for the second tier of individual's attitudes (affective and cognitive).

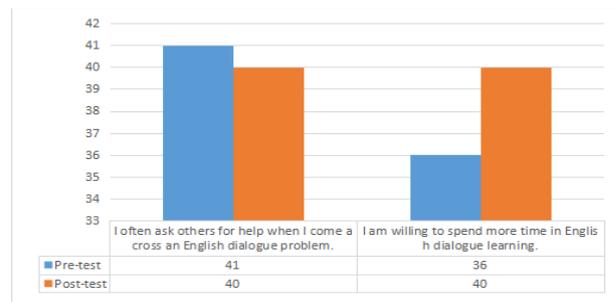


Figure 12. Total Likert score across students for the third tier of individual's attitudes (behavioral intention).

Based on students' responses, their personal experience and affection towards English dialogue learning developed in a more positive direction after the intervention class (see figures 10 and 11). What is more, they were willing to spend more time learning English dialogue than before (see figure 12) as a result of the intervention. However, it is noticeable that students became less willing to ask others for help when coming across an English dialogue problem (see figure 12). Overall, after the intervention, students' attitudes became more positive towards English dialogue learning.

4.3. Pre-test and Post-test Results for RQ 3

RQ3: Do students' attitudes towards graphical programming language learning change after students experience a class in which graphical programming language is infused in English dialogue learning?

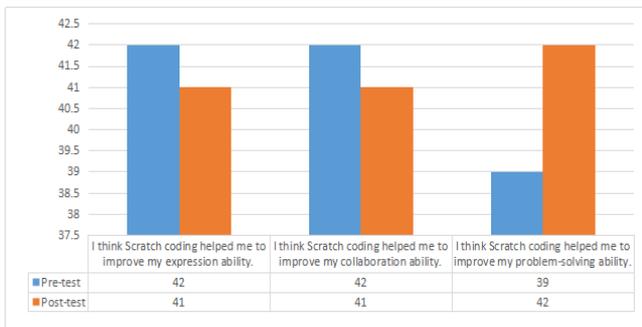


Figure 13. Total Likert score across students for the first tier of individual's attitudes (individual experience).

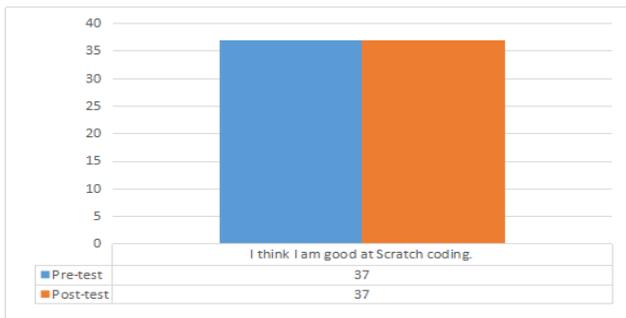


Figure 14. Total Likert score across students for the second tier of individual's attitudes (affective and cognitive).

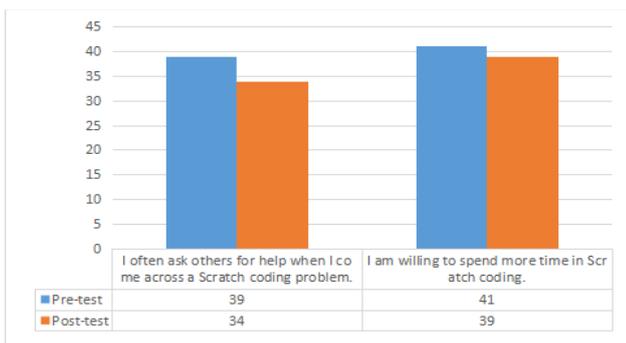


Figure 15. Total Likert score across students for the third tier of individual's attitudes (behavioral intention).

Students' attitudes towards graphical programming language were much more complicated compared to their attitudes towards English dialogue learning after the intervention. In the tier of personal experience, they were less likely to feel that graphical programming language helped them with their expression and collaboration ability after the intervention; however, they thought their problem-solving ability improved while learning graphical programming language (see figure 13). Students' affection towards graphical programming language remained at the same level before and after the intervention (see figure 14). However, in the behavioral tier, students became less willing to ask others for help when coming across programming problems and less willing to spend time coding with graphical programming language after the intervention (see figure 15). Generally speaking, students' attitudes towards graphical programming language became less positive after the intervention.

5. DISCUSSION

This research has achieved its goal to provide some initial insights on the integration of CT into English education, and therefore benefit students from CS beyond the CS class.

Students welcomed the novel practice of utilizing Scratch in other courses. One of the biggest challenges in teaching K-12 students is how to hold their attention -- since Scratch is designed to cater to students' needs and maintain children's interest, students can absorb the knowledge of other courses being taught through Scratch in a subtle way and they will not feel bored in this process. It is obvious from the pre-test and post-test comparison that students became more motivated to learn English dialogue after the intervention. Scratch enables students to do visual programming by themselves, presenting them with colorful Sprites and offering them the chance to take part in the dialogues interactively in the simulated environment. As such, Scratch makes English dialogue learning interesting and different from what students have experienced in their ordinary English dialogue learning classes. Meanwhile, students' attitudes towards graphical programming language became less positive after the intervention. Since the intervention only lasted for one session, not much differences happened in students' coding ability, it is reasonable that students' self-evaluation towards Scratch coding ability remained the same in the pre/post surveys. Students' declined initiative efforts and willingness to study more about Scratch coding after the intervention have great enlightening significance for the instructional design of the CT infusing class -- when students' attention was drawn by the appealing content of the infusing class, it is easier for them to get frustrated if the programming tool goes wrong compared with the pure programming class in which they only have one focus to concern.

However, limitations exist in this research. This research only has a sample of nine students, which makes it difficult to generalize any information collected from the pre-test and post-test to the average student. Furthermore, the researcher asked the same students almost the same questions in both the pre and post surveys (the only difference was that the post-test asked about student's perspective towards the infusing class while the pre-test did not). Without a control group, this means that some of the changed effects in attitude might not be due to the intervention but rather due to students being 'primed' by the pre-survey. What is more, the intervention class was too short to make any influential changes of students' CT and English dialogue learning capability, though this research only involves the perspective aspect of the participants, more insights would have been achieved if the intervention were longer.

6. FUTURE WORKS

Future studies on how to integrate CT into English education are needed from various perspectives. While this study focused on students' attitudes, future researchers can go one step further and apply some valid and reliable scales to assess students' learning performance in both CT and English as a result of infusing classes. With these kinds of studies, we can have a better vision of what happens to students when they undergo these infusing lessons and if students can benefit from this learning.

In this research, the researcher adopted Scratch as the CT instrument in the infusing class, however other graphical programming platforms including Alice, Game Maker, Kodu and Greenfoot can be used to promote CT development in K-12 education as well. Therefore, researches on the feasibility of different CT tools in facilitating English education can be an important branch of both CS and English education research.

Additional, integrating CT with English education is an interdisciplinary topic which only has a limited literature support. The work of putting forwards any framework to guide the following practice in this field is highly needed at this stage.

Teachers' professional development and community of practice on how to teach CT are significant factors that cannot be ignored. Research on how to better prepare teachers is bound to have great impact on the classroom effect of the CT infusing lesson therefore should be enhanced.

7. REFERENCES

- Andersen, R. W. (1983). *Pidginization and Creolization as Language Acquisition*: ERIC.
- Armoni, M. (2013). Computing K-12 curricular updates: a necessity, or an unjustified effort? *ACM Inroads*, 4(4), 20-21.
- 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.
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada.
- Burke, Q., & Kafai, Y. B. (2012). *The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms*. Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education.
- Chang, C.-K. (2014). Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction. *Journal of Educational Computing Research*, 51(2), 185-204. doi:10.2190/EC.51.2.c
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Hatch, E. M. (1978). *Second language acquisition: A book of readings*: Newbury House Pub.
- Holt, L. (2011). *Creating Digital Stories with Scratch to Promote Computational Thinking*. Paper presented at the Society for Information Technology & Teacher Education International Conference.
- Hubwieser, P. (2012). Computer science education in secondary schools--the introduction of a new compulsory subject. *ACM Transactions on Computing Education (TOCE)*, 12(4), 16.
- Kafai, Y. B., & Burke, Q. (2013). Computer programming goes back to school. *Phi Delta Kappan*, 95(1), 61-65.
- Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K--8 curriculum. *ACM Inroads*, 5(4), 64-71.
- Liaw, S.-S., Huang, H.-M., & Chen, G.-D. (2007). Surveying instructor and learner attitudes toward e-learning. *Computers & Education*, 49(4), 1066-1080.
- Lockwood, J., & Mooney, A. (2017). Computational Thinking in Education: Where does it Fit? A systematic literary review. *arXiv preprint arXiv:1703.07659*.
- 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.
- Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239-264.
- Moreno-León, J., & Robles, G. (2015). Computer programming as an educational tool in the English classroom a preliminary study. In (Vol. 2015-, pp. 961-966).
- Nesiba, N., Pontelli, E., & Staley, T. (2015). *DISSECT: Exploring the relationship between computational thinking and English literature in K-12 curricula*. Paper presented at the Frontiers in Education Conference (FIE), 2015 IEEE.
- Papert, S. (1991). *Situating Constructionism. Constructionism. I. Harel and S. Papert. Norwood. In: NJ, Ablex Publishing.*
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., . . . Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67. doi:10.1145/1592761.1592779
- Shackelford, R., McGettrick, A., Sloan, R., Topi, H., Davies, G., Kamali, R., . . . Lunt, B. (2006). Computing Curricula 2005: The Overview Report. In (pp. 456-457).
- Tokoro, M., & Steels, L. (2004). *A learning zone of one's own : sharing representations and flow in collaborative learning environments*. Amsterdam: IOS Press.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education (TOCE)*, 11(2), 9.

基于DBR的高中生计算思维的培养

——以信息技术课程为例

苏幼园^{1*}, 马秀麟¹, 毛荷², 王翠霞¹

¹北京师范大学教育技术学院

²洛阳市实验小学恒大分校

201621010214@mail.bnu.edu.cn, maxl@bnu.edu.cn, 919706512@qq.com, 201521010214@mail.bnu.edu.cn

摘要

本研究基于计算思维的维度,探究在信息技术课程中培养高中生计算思维的切入点的问题,提出了通过“设计”培养高中生抽象、建模、分解与综合等维度的计算思维的思路。通过把DBR的研究范式融入到学生的作品“设计”和教师的教学设计过程中,形成了有效的教学实践,并验证了在设计过程中培养高中生抽象、建模、分解与综合等计算思维的可行性。

关键字

计算思维;信息技术课程;设计;DBR

1. 研究背景

1.1. 计算思维的出现、定义及对人才培养的价值

计算机科学与技术对人类的生产、工作、学习和生活产生了重要影响,能够从思维模式的层次影响人们的解题方法,这促成了“计算思维”概念的提出。

2006年3月,周以真教授提出了计算思维的概念。周以真教授认为:计算思维是运用计算机科学的基础概念进行“问题求解、系统设计以及人类行为理解”等涵盖计算机科学之广度的一系列思维活动(Wing J M, 2006)。2015年,张学军等学者在深入探究这些问题的基础上,指出计算思维的培养主要为八个方面:计算、抽象、自动化、设计、通信、协作、记忆和评估(张学军, 2015)。

2016年的《高中信息技术课程标准修订稿》中指出:高中信息技术课程的培养目标是使学生学会运用计算思维对问题进行识别、分析、抽象、建模并设计系统解决方案,在数字化学习的过程中形成对人与世界的多元理解力,成为数字化时代的合格公民(任友群, 2016)。

1.2. 高中生计算思维能力培养存在的问题

尽管计算思维对中学生个人能力的发展非常重要,但从基础教育的现状来看,中学生的信息技术相关操作技能学习得较好,但计算思维能力仍存在严重欠缺。

为计算思维能力开设专门的课程也存在着困难。第一,若以纯理论课程的形式讲授计算思维的概念、规则和内涵,只能促使学生背诵和记忆抽象概念,并不能真正地实现深层次的理解和掌握;第二,让中小學生减负、把更多自由的时间还给学生是当前教育的主流思

想。因此,专门开设计算思维培养类课程并不是最佳选择。

2. 研究问题

高中信息技术课程的很多内容都脱胎于计算机和网络技术,其本身就蕴含着计算思维的思想,因此,把计算思维的培养融入到信息技术类课程之中,进而促使学生掌握计算思维的相关理论、策略和技术的思路是可行的。本研究旨在探索以信息技术课程为载体,思考计算思维的哪些维度易于培养以及如何培养学生计算思维的问题。

- (1)计算思维的维度很多,对高中生计算思维能力培养应着重哪些方面?
- (2)计算思维可以借助何种载体进行培养?
- (3)计算思维培养的具体策略有哪些?
- (4)应用此种策略培养学习者的计算思维,效果如何?

3. 研究设计

3.1. 确立培养内容——选择抽象、建模、分解与综合维度的技能作为计算思维素养培养的首批任务

本学期笔者为高二学生开设《多媒体技术》课程,其中涉及了大量设计能力培养的内容。而设计能力的核心就是分析和综合,归根到底就是对现实问题的解析与抽象的能力,这也是计算思维培养的核心内容。鉴于《多媒体技术》课程的特点及其蕴含的计算思维特性,本研究选择抽象、建模、分解与综合维度的技能作为计算思维素养培养的首批任务。

3.2. 确立培养载体——以作品设计为载体,让计算思维融合于设计过程中

计算思维中的抽象、建模、分解与综合思想本就是“设计学”中的核心内容,设计的过程可以较为直观地体现一个人的思维过程。

《多媒体技术》是一门操作性、实践性较强的学科,学习者对其的掌握大多都是通过作品设计来进行实现。所以,课程任务的实现和设计的一般过程不谋而合,即通过学习者的作品设计过程来体现或培养学习者的计算思维是可行的。

3.3. 确立培养策略——计算思维培养的具体策略

本研究中设计能力的培养不仅仅是学生层面,还包括教师层面。于学生而言,培养其作品设计能力,注重其作品设计过程;于教师而言,为促进学生作品设计能力的发展,提升教师的教学设计能力。另外,无论

是学生设计能力的发展,还是教师为实现此目标而开展的教学设计,都是一个在实践中不断发现问题并完善的过程。

3.3.1. 基于设计的研究范式

基于设计的研究(DBR),其目的是在真实情境中,以研究者与实践者的协作为基础,通过分析、设计、开发和实施的反复循环,来改进教育实践,并提炼对情境敏感的设计原则和理论(焦建利,2008)。目前主要有理论主义、实用主义、干预主义等取向。本研究主要依据张文兰学者的教育干预取向(张文兰,2007),对教师教学活动、学生学习活动进行设计。首先,教师的教学设计应遵循 DBR 的理论,在设计及不断的迭代中成长。再者,学生对作品的设计和计算思维能力的形成也是一个 DBR 的过程,借助“设计”,使其作品不断优化与完善,促使其抽象、建模能力不断提升,逐步形成良好的思维品质,最终达到“加强计算思维意识,掌握计算思维方法,提升计算思维能力”的学习目标。

3.3.2. 教师的教学活动设计

教师在明确章节内容所蕴含的计算思维并完成教学内容设计的基础上,主要采取以下策略组织教学活动。

(1)借助任务驱动,融入任务分解与综合的思维方式

通过“任务分解—任务设计—任务反思—任务优化”四个环节来开展教学,使学生在设计过程、问题解决与任务完成中不断增强计算思维的培养。

(2)促使学生思考如何把现实问题计算机化,形成抽象、建模思想

教师不仅要讲解学科内容与完成任务的具体步骤,更要着重讲解从现实问题转化为计算机可操作模型的过程,向学生呈现教师对作品设计的思考,促使学生思考如何将现实社会的复杂问题一步步地抽象化、模型化,并通过计算机来实现。

(3)要求学生提交设计报告,跟踪其思维过程

学生通过填写作品设计报告,可以对作品的设计思路和过程有更加清晰的认识;教师通过学生的作品设计报告,可以了解其思维过程并进行过程性评价,使评价方式多元化。作品设计报告见表1。

表1 作品设计报告

作品名称	
作者	
创意来源	
设计思路	
具体制作过程	
收获	

(4)把作品设计报告纳入作品质量评价标准

教师对学生最终作品的评价不局限于最终结果的呈现,同时将学生作品设计过程报告列入考察指标,作为衡量其作品质量的重要因素,具体的作品评价标准见表2。

表2 作品质量评价标准

评价指标	要求与分数
作品内容	主题突出。(10分)
	美观和谐、构图完整。(10分)
创造性	作品形式新颖,设计巧妙。(10分)
技术性	正确并灵活运用所讲课程内容。(20分)
思维素养	设计思路与制作过程中思维清晰,逻辑顺畅,是否高效地解决问题。(50分)

(5)设计调查问卷,掌握学情并及时评价

本研究在参考国内外有关计算思维测评量表的基础上,结合高中信息技术课程内容,设计了中学生计算思维(主要包括计算思维意识、方法、能力三大类)调查问卷,以便监测学生在实验前后的计算思维变化状态。

在预测阶段,随机抽取了人大附中高二年级的50名学生进行问卷发放与回收,其中有效问卷50份,有效率100%。对于问卷的信度检验采用的是克朗巴哈系数,结果表明整体与各维度的克朗巴哈系数均大于0.6,所以此问卷有较好的信度。对于问卷的结构效度检验采用的是“KMO和Bartlett的球形检验”,其KMO值为0.825,大于0.7,故问卷具有较好的结构效度。

3.3.3. 让学生在作品设计过程中形成计算思维思想

根据设计的一般流程,要求学生遵循“明确目标、任务分解、抽象建模”的一般流程来完成作品设计。主要包括以下环节。①明确任务。②分析任务。主要借助任务分解的策略把综合任务分解为若干便于操作的子任务。③方案形成。要求学生思考如何实现每一个子任务的计算机化,形成在计算机技术范畴内具备可操作性的技术方案,并最终形成自己拟完成作品的抽象模型,形成方案。④反思与迭代。针对已经形成的方案,根据课程要点思考任务完成过程中需注意的地方,进一步反思并迭代。⑤技术实现并形成作品。搜集相应素材,进行作品设计与制作,并及时记录作品设计报告。⑥分享、完善与提交。进行作品的相互分享、完善与提交。

4. 教学实践

4.1. 教学模块选择与教学对象分析

本研究中高中信息技术课程中所使用的教材为《多媒体技术》,本教材中含有Photoshop设计模块。本轮教学实践讲授的是PS中的分图层操作。其所蕴含的计算思维是分解、模块化、设计。本研究教学实践选取的教学对象为中国人民大学附属中学高二年级学生,共112名。

4.2. 第一轮教学实践

4.2.1. 教学内容设计

(1)学科内容

学科内容包括图层的概念;图层的主要操作;利用图层完成复杂图像的设计三方面。

(2)计算思维内容

图层概念中蕴含着“分解”与“综合”的计算思维思想。在本教学过程中，应该体现出如何把整个综合性大任务分解为若干个小任务的任务分解的思想，并在任务分解中，遵循“自上而下，逐层分解，由繁到简”的方式完成复杂任务的解决。

4.2.2. 教学活动与任务设计

基于以上的分析和设计，此章节的演示讲解与任务实施过程如表3。

表3 多图层操作的演示与讲解过程

教师活动	学生活动
呈现反面案例： 将所有内容都放到一个图层中的图像，并提问如果想修改图像中的某一部分(如背景色)该如何操作？	思考并尝试解决
引入图层概念： 图层的概念	了解概念及作用
呈现简单任务： 利用图层分别存储复杂图像的局部内容。特别正式且严肃地向学生强调“分解”的重要性及其对未来作品修正的便捷性。	形成分图层处理图像的计算思维，并掌握图层的新建、删除等内容要点
演示操作过程： 以简单案例进行举例，并进行操作演示	更进一步地体会分图层处理的重要性
师生交流互动： 总结知识要点	提问疑问及想法
发布任务： 设计防霾海报，呼吁人们重视空气污染问题	明确任务主题
明确任务要求： ①使用的素材数量不少于8个；②作品应存储为PSD格式，保留操作过程中的图层，图层不少于5层；③记录作品的设计思路和过程，并与作品文件一起打包上传至平台。	明确任务要求，构思作品：怎么做，需要什么，应该注意什么等
作品设计： 巡视学生作品设计情况，并给予适当指导	搜集素材，制作作品，记录作品设计报告
作品分享： 学生之间进行作品的互相分享	作品的完善与优化
作品提交： 提醒作品上传	将作品、设计报告等文件上传至平台

4.2.3. 教学评价与存在问题

(1)教学评价

从多媒体作品设计报告的角度来看，学生的作品契合主题，构思各异，形式也别具一格，并在制作过程中体现出了其理解任务、一步一步分解并逐步设计完成的过程。当然，也有一部分学生指出自己面对复杂图像分层完成的能力还不够，起初还是没有意识到任务分解、图像分层的重要性，直到修改局部内容的时候才发现操作十分复杂，才更加深刻理解任务分解的重要性。

从多媒体作品质量来看，具体如表4，70分以下的学生占比62.5%，80分以下的占比94.64%，作品的质量还

不甚理想。学生们在课堂上的积极表现和问题抽象与任务分解能力的提高并未显著反映到作品质量的提高上来。

表4 “抗霾公益海报”作品成绩

成绩	60分以下	60-70	70-80	80-90	90-100
人数	14	56	36	4	2
百分比	12.5%	50%	32.14%	3.6%	1.79%

(2)存在问题

基于课堂观察及对部分教师和学生的访谈，发现本次教学实践中突出的问题有：①学生用于填写作品设计报告的时间过短，大部分学生拿到任务后会一头扎进实际制作中，而不是进行认真梳理和缜密规划，由于缺乏系统的设计，导致最终的作品缺乏整体的设计感和可修改性。②在任务实现过程中，师生都有些过于关注作品自身而忽略了学生思维的形成过程。虽然在开展教学实践前已经和教师沟通过要充分体现学生的主体地位和关注作品实现的思维过程，但教师习惯于传统的讲授型教学方式，并不能真正放手让学生去体验设计的过程。

4.3. 三轮教学实践及其效果

4.3.1. 以DBR为指导的持续教学实践

基于第一轮教学实践中出现的问题，笔者参考DBR中对“设计”及其规范的要求，对教学实践和活动组织进行了调整，然后组织了第二轮和第三轮教学实践。

(1)针对第一轮的教学实践的改进与设计

①针对学生拿到任务不认真进行设计和规划的情况，教师在发布任务前会告知学生作品的评价标准和细则，并强调作品设计报告所占的分值及重要性，在完成任务过程中也会及时督促其记录作品的设计思路和过程。②针对教师过度关注学生的情况，需注意要留给学生适当的思考和尝试的空间，要大胆地启发学生去尝试和探索，鼓励他们举一反三，综合运用各种方法来解决问题，实施以学生思维活动为主的教学过程。

(2)第二轮和第三轮教学实践的实施

为了推进基于分解、模块化、设计等计算思维思想的形成，基于第一轮教学实践经验及其存在的问题，笔者还在动态画笔模块、动作模块和小动画制作模块继续推行“学科内容+计算思维培养”教学模式，基于抽象、建模、分解和综合思想组织教学活动，持续进行了3轮教学实践，每轮实践均在上一轮实践反思优化的基础上进行。

4.3.2. 三轮教学实践的成效及分析

(1)从计算思维的意识、方法、能力来看

因问卷中的题目选择项均为3项，收集的数据为低测度的定序变量，所以对其差异性的检验使用的是基于交叉表的卡方检验。

从卡方检验来看，具体见表5，计算思维意识、方法、能力上前后存在显著差异，说明从设计角度提升学生的计算思维是有效的。

表5 计算思维前后测问卷卡方检验

计算思维素质	题号	渐进 Sig.(双侧)
意识	a1	0.000**
	a2	0.001**
	a3	0.004**
	a4	0.001**
	a5	0.000**
方法	b1	0.000**
	b2	0.035**
	b3	0.000**
	b4	0.000**
能力	c1	0.000**
	c2	0.650**
	c3	0.000**
	c4	0.225**
	c5	0.014**

**在 0.05 水平(双侧)上显著相关

(2)从作品设计报告来看

针对第一轮教学实践中存在的问题，第二轮与第三教学实践中均进行了改进。经过训练，在第三轮教学实践中，学生的设计报告日益规范和完善。于学生而言，其作品设计与制作过程思路更加清晰，任务完成更加高效；于教师而言，在对设计报告指出问题的同时，也在改变着学生的思维逻辑，教师关注的不仅仅是最终的作品，而是其设计的过程，真正实现了计算思维培养的教学。

(3)从作品质量来看

在思维意识、方法、能力不断掌握的过程中，学生的设计过程不断清晰明了，作品质量也有显著提升。第

三轮教学实践发现：在 70 分以下的作品急剧减少，70-90 分作品逐渐增多，90 分以上的作品逐渐增多，且占比将近 50%。

5. 研究总结

(1)借助信息技术课程中的设计章节，从教师教学设计、学生作品设计的角度进行抽象、建模、设计、分解与综合等维度的计算思维的培养是可行的。

(2)教师对于课程的讲解应更多地在于思维逻辑能力的逐步渗入，对于任务的完成应体现以学生为主体的理念，当然，对于基础较弱的学生应给予主动的询问和指导，并提倡生生互助。

(3)作品设计报告的实时、认真、规范撰写对学生的计算思维培养有着非常重要的作用，教师需认真设计作品设计报告的内容和格式，以便体现学生们的思维逻辑变化及存在的问题，以便进行及时的提升和帮助。

(4)计算思维的培养不是一蹴而就，而是不断改善、不断实践的迭代过程。

参考文献

Wing J M(2006). Computational thinking[J]. *Acm Sigcse Bulletin*, 49(3):3-3.

张学军、郭梦婷和李华(2015). 高中信息技术课程蕴含的计算思维分析[J]. *电化教育研究*, (8):80-86.

任友群和黄荣怀(2016). 高中信息技术课程标准修订说明 高中信息技术课程标准修订组[J]. *中国电化教育*, (12):1-3.

焦建利(2008). 基于设计的研究:教育技术学研究的新取向[J]. *现代教育技术*, 18(5):5-11.

张文兰和刘俊生(2007). 基于设计的研究——教育技术研究的一种新范式[J]. *电化教育研究*, (10):13-17.

Promoting Computational Thinking and Collaborative Skills in Primary Robotics Classes

Hyungshin CHOI^{1*}, Jeongmin LEE²

¹Chuncheon National University of Education

²Ewha Womans University

hschoi@cnue.ac.kr, jeongmin@ewha.ac.kr

ABSTRACT

This current study reports our attempt to design and implement a course to promote computational thinking and collaborative skills for primary school students in Korea. We have incorporated Wedo 2.0 into fourth graders' curriculum in various real world problem solving contexts. This paper reports the students' activities, learning outcomes in terms of computational thinking and collaborative/communication skills.

KEYWORDS

Computational thinking, Bebras tasks, Collaborative skills, Robotics Classes, Primary Education

1. INTRODUCTION

As computational thinking(Wing, 2006) is widely recognized as the core competency in software-embedded society, various educational attempts are being made to promote primary students' computational thinking skills in Korea. These efforts include educational programming such as Scratch programming, physical computing with robotics and microcontrollers, and unplugged activities.

It is claimed that computational thinking can be promoted by computer programming because it provides kids with debugging and troubleshooting chances where they receive quick feedback (Bers, 2018). Furthermore designing and programming robots to function offer tangible objects for kids to play with and observe so that debugging becomes more visible. It is often neglected that, however, computational thinking is a problem solving skill and therefore students should apply computational thinking skills in authentic problem situations while collaborating with their peers.

This current study reports our attempt to design and implement a course in primary education in Korea. Specifically we aimed to investigate how the current course design of primary robotics activities impacts on students' computational thinking and collaborative/communication skills.

2. CONTEXT & METHODOLOGY

2.1. Participants and Research Procedure

In the current study, 75 Korean fourth grade school students participated in the robotics classes. Robots programming classes were designed as a subject integration project. The same modules were carried out in four different classes from the 2nd week of September to the 1st week of December 2017. The pre-CT Bebras tests and pre-tests of questionnaires were given before the first module and post-tests were given after the 7th module. We have selected 59

students as final research subjects after removing incomplete responses. The participants were 31(52%) boys and 28(48%) girls.

2.2. Measuring Instruments

In order to investigate the effects of the robotics class we look into two main areas: cognitive and social. To measure cognitive skills we focused on students' computational thinking and incorporated Bebras tasks(www.bebas.org). The Bebras tasks consisted of authentic problems used to measure students' CT transfer. We selected 6 items from the Korean Bebras pilot test conducted in 2016 (Park & Jeong, 2017). In addition, to measure social skills we concentrated on collaboration and communication skills. Collaboration skills were measured using the 5-Likert scale by Yoon and Kim (2011). The instrument consisted of 9 items and coefficient alpha is .780. Communication skills were measured using communication the 5-Likert scale questionnaires by Choi and et al. (2013). The instrument consisted of 5 items and coefficient alpha is .845.

2.3. Data Analysis

SPSS was used for the data analysis. First, we conducted a matched pair *t*-test to discover if robotics programming education improved students' computational thinking, collaboration, and communication skill.

2.4. Robotics Class Design

As Table 1 indicates, we designed the robotics class including 7 modules, and each module took 2 hours. The modules were designed to help 4th graders solve problems in authentic scenarios such as earthquake, rescuing people, recycling, and food deficiency situations. The primary students act as researchers in a future disaster research center who need to solve the incurring 'real-world' problems. For each module, students were urged to work together as a team of two acting one as a designer and the other as an engineer. The designer designs the robot and the engineer creates programs to solve the problems. The team members were encouraged to switch the roles back and forth allowing them to be able to perform two roles. In addition, as students build collaborative robots they work as a team of four members, and combine two robots into one or synchronize robots' behaviors.

Table 1. Robotics class modules' themes

Module	Themes
1	Disaster Robots Introduction
2	Designing Rescue Robots
3	Future Food Problem Solving Robots
4	Building Earthquake-resistant Houses
5	Recycling Helper Robots
6	Collaborative Robots
7	Designing Future Robots

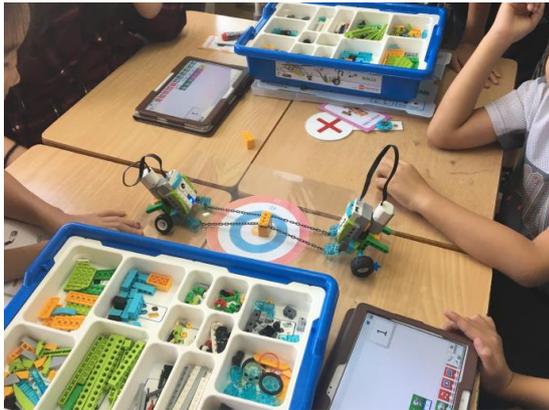


Figure 1. Collaborative rescue robots

3. RESULTS

3.1. Cognitive : Computational Thinking skill

A paired samples t-test showed a statistically significant increase in computational thinking from pre-test $M= 1.85$, $SD= 1.06$ to post-test ($M= 2.47$, $SD= 1.24$), $t(58)=-3.636$, $p<.05$.

3.2. Social : Collaboration/Communication skill

Collaboration skills significantly increased from pre-test ($M= 3.79$, $SD= .51$) to post-test ($M= 4.03$, $SD= .48$), $t(58)=-4.247$, $p<.05$. In addition, communication skills also significantly increased from pre-test ($M= 3.46$, $SD= .61$) to post-test ($M= 3.76$, $SD= .58$), $t(58)= -4.425$, $p<.05$.

4. CONCLUSIONS & FUTURE STUDY

This research reports our design and implementation of fourth graders' robotics classes to promote their computational thinking and social skills. As our findings indicate, the robotics programming classes positively impacted primary students' computational thinking skills. Although the pre-test results of Bebras tasks were relatively low the post-test scores were significantly improved. In order to investigate students' persistent development of CT skills, a series of design-based research will be conducted.

In addition, the robotics programming classes positively impacted on primary students' perceived collaborative/communication skills. The robotics modules were designed for students to collaborate in a group of two (module 2-4) and four (module 6-7) to solve authentic problems. This provided the students with opportunities to work together and communicate to achieve the goal.

5. ACKNOWLEDGMENT

This work was supported by the Ministry of Education of the Republic of Korea and the National Research Foundation of Korea. (NRF-2016S1A5A2A0392687)

6. REFERENCES

- Bers, M. U. (2018). *Coding as a playground*. New York: Routledge.
- Choi, Y., Noh, J., Lim, Y., Lee, D., Lee, E., & Noh, J. (2013). The Development of the STEAM literacy measurement instrument for elementary, junior-high, and high school students. *Journal of Korean Technology Education*, 13(2), 177-198.
- Park, Y., & Jeong, I. (2017). Assessing elementary school students' computational thinking skills on Bebras tasks. *The Korean Association of Information Education Research Journal*, 8(1), 27-31.
- Yoon, H., & Kim, S. (2001). The effects of cooperative learning applying Jigsaw II on learners' self-regulated learning, achievement, self-esteem & cooperation. *Studies on Education of Fisheries and Marine Sciences*, 13(2), 194-211.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 19(3), 33-35.

Computational Thinking and IoT

A Design-based Approach to Implementing a Computational Thinking

Curriculum with App Inventor and the Internet of Things

Chi-hung TSENG^{1*}, Mike TISSENBAUM^{1*}, Wen-hsuan KUAN², Feng-chih HSU³, Ching-chang WONG⁴

¹Massachusetts Institute for Technology

²University of Taipei

³CAVEDU Education, Taiwan

⁴Tamkang University, Taiwan

chihung@mit.edu, mtissen@mit.edu, whkuan@utaiei.edu.tw, jesuvsictory777@cavedu.com, wong@ee.tku.edu.tw

ABSTRACT

The growing ubiquity of everyday devices connected over the Internet, known generally as the Internet of Things (IoT), has opened up new avenues for students to explore their worlds and think and create computationally. Combining IoT with mobile technologies (such as smartphones), enables students to move their designs and computational thinking out of traditional classroom settings and into the real world. This article outlines a design-based IoT curriculum that connects Taiwanese students with the personally-relevant issue of air pollution. The curriculum employs student-driven smartphone application design, using MIT's App Inventor, with Wi-Fi enabled IoT devices (LinkIt 7697 Wi-Fi/BLE MCU board). This paper reports on changes to the curriculum based on a preliminary pilot and observations of student engagement during the most recent enactment.

KEYWORDS

Computational Thinking, App Inventor, Internet of things, Curriculum design, air pollution

1. INTRODUCTION

1.1. *Internet of Things: The Next Sphere of Digital Empowerment for Learners*

As Asoton (2009) so clearly highlighted, the increasing ubiquity of our everyday objects connected through the Internet, commonly termed the Internet of Things (IoT), is changing our daily lives in profound ways. This persistent connectivity is even reaching into our home. From our refrigerators, to our lightbulbs and thermostats, even our home entertainment systems are all increasingly connected to the Internet and controllable through mobile applications. Smart hubs like Google Home or Amazon Echo, are acting as "digital assistants" that allow you control your home appliances by simply talking to them. However, most of these systems are black boxes to users. We do not know how they work, what they do with our data, and generally cannot customize them for our own needs. Similar to the call for computing education to embrace mobile computing as a means for empowering students as creators and not mere consumers of our digital futures (Tissenbaum, Lee, et al., 2017), there is a growing need to consider how to effectively integrate IoT into educational designs.

1.2. *Making Computational Thinking Meaningful*

The continued focus of computing education with learning the fundamentals of computing (e.g., loops, conditionals,

functions, variables, and data handling) first proposed by Wing (2006) and others, risks disconnecting what students learn from how they might apply it in their own daily lives. This separation of learning from contexts threatens to make learners feel that they do not need to learn computing, because they cannot see how it will apply to lives or their futures, a challenge commonly faced in math and physics education (Williams et al., 2003; Flegg et al., 2012). It therefore becomes critical for education designers to understand what current issues may be of interest to learners, and how they can develop educational interventions that can connect them to computing education.

For instance, air pollution has become a rising problem in Taiwan recent years. This problem is keenly understood by everyone in Taiwan and is the topic of science and other disciplines within K-12 education. One source of data for understanding the air pollution status in Taiwan is the readings provided by government air quality stations. However, even within a small geographical region like Taiwan, the air quality can vary significantly, and the government stations alone are not robust enough to capture the variances. The increased availability of low cost IoT devices and peripherals, coupled with Internet connectivity offers new opportunities for the public to design and build their own sensors, and to collectively share that data to public or private servers (Chen et al., 2017). The convergence of personally meaningful context and low-cost technology provides the ideal context for designing computational curriculum that can engage students in a personally meaningful way.

1.3. *Reducing Barriers for Computational Thinking*

While engaging students in IoT-focused computing curriculum may provide new ways for making computing personally meaningful, it is not without challenges. Most programming languages require arcane syntax and grammar, which is a significant barrier for young learners wishing to engage in computational practices (Maloney et al., 2004). If our goal is to have students feel empowered to develop computational solutions to real-world problems and become excited about their ability to do so, we need to reduce these barriers to entry. In response, researchers have developed block-based programming environments, in which users assemble programs by snapping "blocks" of code together. These blocks-based languages have been shown to support novice programmers to more easily develop relatively complex programs in the domains of games (Brennan and

Resnick, 2012), 3D animations (Dann, Cooper, Pausch, 2011), and computational models (Begel & Klopfer, 2007).

MIT's App Inventor is an example of a blocks-based programming environment that allows users to build fully functional native apps for Android phones and tablets. Because App Inventor is focused on mobile applications, it allows the programs that young learners build to move off their computer screens and into their lived lives. When coupled with sensors and other IoT devices, App Inventor can open exciting new possibilities for students to experience, understand, and interact with their physical worlds. While the promise of youth developing transformational interventions using IoT is exciting, the technical complexity required to actually develop these interventions is a clear barrier.

2. METHODS

2.1. *Developing a Low-barrier IoT Curriculum for Taiwan through App Inventor*

In response to the challenges of designing a personally meaningful computational thinking curriculum for Taiwanese students that does not require complicated pre-existing programming knowledge, we developed new IoT extensions for App Inventor. Below we discuss the development of the new IoT functionalities for App Inventor and the successive iterations of our air quality curriculum.

2.2. *Why Wi-Fi not Bluetooth*

There has been significant prior work focused on using Bluetooth to control robots or devices (AlHumoud et al., 2014). However, compared with Wi-Fi, the range Bluetooth can cover is much smaller, making it mainly suitable for spaces about the area of a classroom. When it comes to larger spaces, such as a playground or even a campus, Bluetooth does not have the range to support communication between devices. In this curriculum design, students' air quality monitor systems could be 50 to 500 meters away from each other, well beyond the range of Bluetooth or Ethernet cables. In these cases, we recommend using development boards that are Wi-Fi enabled (such as the LinkIt 7697 used here).

While there are many prototyping boards options, embedded boards - boards that have all the necessary parts for controlling other devices already built into them (Barr & Massa, 2006) - are particularly useful for educational purposes. Embedded boards are significantly cheaper, smaller, more portable, and have lower power consumption than full-fledged computers. It is also fairly easy to power prototypes developed using embedded boards using small portable power sources (e.g., AA batteries or power banks) (Tseng et al., 2017).

By coupling these embedded boards with mobile technologies, we can extend their capabilities in ways that would be prohibitively complex on their own. For instance, voice recognition is relatively simple to implement with smartphones (similar to Google Assistant or Apple Siri). However, this kind of functionality is extremely difficult to implement on embedded boards alone. Combining the two naturally complements the affordances of each and allows

us to envision more complex and engaging educational designs.

2.3. *An Authentic Problem: Taiwan's Air Pollution*

The design of this camp is focused on the current air pollution problem in Taipei, which has become an increasingly serious health threat to everyone living there. Among all pollutants, fine particulate matter (PM2.5 - particles that are less than 2.5 micrometers in diameter), are particularly serious as they can penetrate the alveoli (the gas exchange regions of the lungs) and even pass through the lungs to affect other organs. PM2.5 have been shown to cause serious illness and increase cancer rates and is directly related to a range of serious health problems, such as asthma, cardiovascular disease, respiratory diseases, lung cancer, and premature death (Chen, 2017). According to Taiwan Environmental Protection Administration (2018), a person's respiratory system can be seriously affected when the PM2.5 level is above 50 $\mu\text{g}/\text{m}^3$. Given the seriousness of the problem, and its direct connection to the population of Taiwan, the subject matter was one we believed participants would be able to directly connect to.

2.4. *Participants and Setting*

This work was designed as a summer camp taking at three different high schools in the same week. Each camp has 30 students randomly separated into 12 to 15 groups disregarding gender or prior programming experiences. The camp took place over five 6-hour days (9:30 to 16:30 each day). In each camp, one expert instructor conducts the curriculum and three TAs are present to work with students and collect observations. At the end of each day, the instructor and TAs debriefed together to exchange information about interesting and unexpected events.

2.5. *Data Collection*

Observations during camp sessions were collected by the instructors and TAs, debrief sessions with the instructors and TAs, and the students' final products.

3. CURRICULUM DESIGN

This curriculum was implemented using a design-based research approach, which employs iterative cycles of design, deployment, observation, and redesign (Barab & Squire, 2004).

3.1. *LinkIt 7697 Wi-Fi/Bluetooth MCS Board*

For IoT connectivity we used the LinkIt 7697, which is an Arduino compatible development board (2018). It supports Wi-Fi and Bluetooth Low Energy connectivity. With its relative affordable price (about 15 USD) and the support of the open source community, LinkIt 7697 board is relatively easy for beginners to get started with. Students can quickly build and test their designs without any complicated setup.

In this camp, we combined the LinkIt 7697 to a PM2.5 sensor as a prototype for students to collect PM2.5 data and to further explore the physical world.

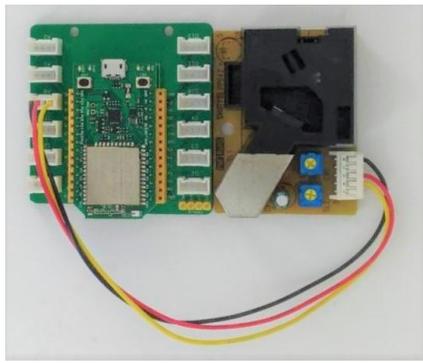


Figure 1. Prototype of the air pollution monitor system, including LinkIt 7697

To make the hardware setup easier, we used an extension board to connect the air quality sensor, removing the need for students to use a breadboard and messy wires. This was a suitable approach for a camp, but for a longer-term intervention, it might be better for students to have more hands-on experiences building breadboard circuits.

3.2. Curriculum Content

Below we describe each day of the five-day camp.

Day 1. Basic Understanding of Mobile Programming.

The camp started with an introduction to the App Inventor platform and having students install the Android emulator on their PCs. Using a set of tutorials, students developed a basic understanding of the AIA environment and its components (e.g., Buttons, Textboxes, Images, Webview), and how to build apps to complete certain tasks (e.g., to have users input two numbers into Textboxes to calculate the area of a rectangle, and how to show error message if either one Textbox is empty).

Day 2. Get into Mobile Phone's Functionalities. On Day 2, using the premise of game design, students learned how to integrate multimedia, and sensing functions in their apps. Students had to make a virtual ball on the screen roll according to the phone's orientation (utilizing the orientation/accelerometer sensor). We also introduced the Map and location sensor components, having students build a location-based app to detect their location.

Day 3. Basic Understanding of Circuits and MCU boards. On day 3, students began working with the LinkIt 7697 MCU board to control several electrical components, such as LEDs, potentiometers and buttons. In the afternoon, students built a light-controlled LED, in which the LED intensity was affected by the ambient light condition using the photoresistor.

Day 4. Receive Data from MCU Board to Show on the App Screen. On day 4, students built the main components of their air quality monitoring systems. Students learned how to control the MCU board through their mobile phone through Wi-Fi, how IP connections work, how to send their air quality sensor data to the server, and how to retrieve data from other devices and show it on their phone's screen.

Day 5. Demo, Share and Feedback. On the fifth and last day, students finished their monitor system and tried to add more functions to it, such voice control or using different

color LEDs to indicate the air quality. In the afternoon, they presented what they had built and learned during the week to the larger group. As a follow-up reflection, students were asked to discuss and write down what they could add to make their projects to make them better. Some of this discussion was/will be used to improve subsequent iterations of the camp.

The prototype of our air pollution monitor system uses a LinkIt 7697 MCU board, extension board and an air quality sensor (can detect PM10, PM2.5) (Figure 1).

Students then designed their own interfaces based on what they learned over the first three days. Each group came up with different ways to present their data: one team used Google Chart API to visualize the hourly PM2.5 status updates, while others simply showed the readings on the screen (Figure 3).



Figure 2. Examples of student representations of air quality

4. OBSERVATIONAL STUDY

During the one-week camp, almost every group of students from the three high schools finished building their air pollution monitor systems, sent data to the server and reviewed all the air quality sensor data installed within the campus. From our observations, about half of the students resisted or paid less attention at first because they felt that mobile and IoT programming was too difficult to learn. However, with the help of AIA and the easy-to-connect hardware, these students began to explore more of the AIA functionality on their own. This exploration was clearly seen in their final project presentations. All the students were visibly excited when they successfully sent data to the server and were able to view their surrounding campus' overall air quality. Many students expressed the idea that this data was really meaningful and impacted their perceptions of the need to understand and care for the environment in the future. After the camps were over, we randomly selected several students from each camp and quoted their feedback below:

"It's really exciting when I see the data jumping on the screen."

"Now I know what the difference is between Wi-Fi and Bluetooth."

"I can prepare a mask before I go to school if the app tells me today's air condition is not good."

The results of this first full implementation of the camp was extremely encouraging and will further design iterations for year two.

5. DISCUSSION AND FUTURE WORK

This paper described the design and development of a one-week IoT curriculum with high school students which aimed help them in developing their identities as computational thinkers. This iterative work built on a previously piloted version of this camp held across three high schools in Taipei city, Taiwan. By reviewing the observatory results and student's projects, we saw that overall, students were motivated and connected to the work because the topic was connected to their daily lives.

Building off our current run of the camp, we have some thoughts on how to extend and improve the curriculum. For students who want to explore further, it might be fruitful to provide them with opportunities to try out other making skills that connect to this topic, such as how to use 3D printing or laser cutting to fabricate an exterior case for their air pollution monitor system to provide better protection. Another option for student exploration could include opportunities for students to add additional sensors to expand their device's functionality. For instance, they could add temperature, humidity and wind direction sensors to provide a more comprehensive analytics results. They could also add multi-color LEDs to indicate different air quality conditions or extend their mobile apps to provide a pop-up notification when the air quality condition is bad.

Building off these insights, in future camps we will design more inquiry activities for students to design and build with AIA and IoT devices, allowing them to explore their surrounding environments through computational means. Future iterations of this camp will also involve more students and will employ a pre/post survey to collect more detailed quantitative and qualitative findings. Through this work, expect to have a more comprehensive understanding of how students become computational thinkers through participation in this camp, and how it may affect students' computational thinking skills, eventually their future study and career pathway choices.

6. REFERENCES

AlHumoud, S., Al-Khalifa, H. S., Al-Razgan, M. & Alfaries, A. (2014). Using App Inventor and LEGO mindstorm NXT in a summer camp to attract high school girls to computing fields, *2014 IEEE Global Engineering Education Conference (EDUCON)*, Istanbul, 2014, pp. 173-177.

Barab, S. & Squire, K. (2004). Design-Based Research: Putting a Stake in the Ground. *Journal of the Learning Sciences*, 13(1), 1-14.

Barr, M. & Massa A. J. (2006). "Introduction". *Programming embedded systems: with C and GNU development tools*. O'Reilly. pp. 1-2. ISBN 978-0-596-00983-0.

Begel, A. & Klopfer, E. 2007. Starlogo TNG: An introduction to game development. *Journal of ELearning*.

Chen, L.J., Ho, Y.H., Hsieh, H.H., Huang, S.T., Lee, H.C., & Mahajan, S. (2017). ADF: an Anomaly Detection Framework for Large-scale PM2.5 Sensing Systems. Accepted to *IEEE Internet of Things Journal*.

Chen, L.J., Ho, Y.H., Lee, H.C., Wu, H.C., Liu, H.M, Hsieh, H.H., Huang, Y.T., & Lung, S.C. (2017). An Open Framework for Participatory PM2.5 Monitoring in Smart Cities. *IEEE Access Journal*, 5, 14441-14454.

Flegg, J., Mallet, D., & Lupton, M. (2012). Students' perceptions of the relevance of mathematics in engineering. *Intl. Journal of Mathematical Education in Science and Technology*, 43(6), 717-732.

Lee, C. H., & Soep, E. (2016). None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*, 49(4), 480-492.

LinkIt 7697 Development platform. Retrieved from <https://docs.labs.mediatek.com/resource/linkit7697-arduino/en>

Location Aware Sensing System. Retrieved from <https://pm25.lass-net.org/>

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004, January). Scratch: a sneak preview [education]. In *Creating, connecting and collaborating through computing, 2004. Proceedings. Second International Conference on* (pp. 104-109).

PM2.5 concentration indexes and activity advices. Retrieved from <http://taqm.epa.gov.tw/taqm/tw/fpmi.aspx>

Tseng, C.H., Wong, C.C. & Kuan, W.H. (2017). Implementation of a map route analysis robot: combining an Android smart device and differential-drive robotic platform. *MATEC Web Conf*, 95 (2017) 08005.

Wagner, A., Corley, G. J., & Wolber, D. (2013). Using app inventor in a K-12 summer camp. *Proceedings of the 44th ACM technical symposium on Computer science education*, pp. 621-626.

Williams, C., Stanisstreet, M., Spall, K., Boyes, E., & Dickson, D. (2003). Why aren't secondary students interested in physics? *Physics Education*, 38(4), 324.

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

Computational Thinking Development in Higher Education

The Use of Computational Thinking to Advance Learning in the Pre-university Subject of Digital Literacies

Ildiko VOLCZ
University of Technology, Sydney, Insearch
ildiko.volcz@uts.edu.au

ABSTRACT

This paper describes how elements of computational thinking are employed to advance student learning and engagement in the Digital Literacies subject of the Foundations Studies of the University of Technology, Sydney. This study does not focus on enhancing technical coding skills, rather it takes the conceptual view on computational thinking and investigates how aspects of it can be used to further students' academic skills and develop their abilities to solve complex problems in collaboration and with the use of technology.

This paper will also reveal 115 students' responses of the use of computational thinking elements in project-based assessment.

KEYWORDS

Computational thinking, Digital Literacies, project-based learning.

1. BACKGROUND

The UTS Foundation Studies Program is offered in Australia by Insearch to international students; the academic entry level requirement is completion of Year 11, with an English language requirement of IELTS overall 5.5 and a minimum of 5.0 in writing. The course is designed to develop students' academic and language skills to prepare them for a university education.

The student body represents a diverse population in terms of gender, country of origin and also use of technology. There is great fluctuation in the composition of nationalities from semester to semester: in some semester there is approximately 40% Chinese and 40% Nepali students, in other semesters, 75% of the students are from China. The rest of the student population is generally from Indonesia, India, Korea, Laos, Malaysia, and Vietnam. Therefore, the curriculum has to be flexible enough to cater for students with different skill levels while aiming for consistent learning outcomes.

The composition of the cohort that provided information for this paper consisted of 75% students from China, 7% from Indonesia as the second biggest group of students, 2% from Korea and 2% from Hong Kong; the rest of the students were from Nepal, Myanmar, Pakistan, Nepal, Malaysia.

The cohort represented 55% male, 45% female students; 55% studying Business, 13% Communication, 10% Design, 6% Engineering.

2. INTRODUCTION

The aim of the UTS Foundation Studies is to provide preparatory education to international students for university

courses. The role of the Digital Literacies subject in the program is to equip students with the technical conceptions and skills to become efficient users of digital and online resources. The subject employs a number of computational thinking principles and elements ranging from simple coding activities to complex project-based collaborative learning assessments. This paper will describe how computational thinking is applied in this subject.

3. COMPUTATIONAL THINKING

3.1. Definition of Computational Thinking

There are numerous definitions of computational thinking (CT); one of the widely applied description by Jeanette M. Wing (2011) states: "computational thinking is the thought processes involved in formulating problems and their solutions so that solutions are presented in a form that can be effectively carried out by an information-processing agent". Hemmendinger (2010) describes it as: "Teaching computational thinking, however is something else; not to lead people to think like us — which is pretty varied anyway. Instead, it is 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." These definitions suggest that computational thinking skills go beyond Computer Science and can be applied to non-computing subjects. DeSchryver and Yadav (2015) argue that computational thinking skills (as strategies for problem solving in data-mediated, technology-rich learning and work environments) coupled with the use of new literacies skills (strategies to negotiate, generate, and communicate meaning among myriad encoded digital forms) enhance creative thinking skills (cognitive activity comprising various subsets of these component thinking skills that are mediated by the more aesthetic components of traditional creativity). There are various calls for opening up CT elements to learning that is not computer dependent.

The heightened need for including computational thinking in K-12 curriculum that is supported by government bodies: England added Computational Thinking and Computer Programming in the national curriculum of primary and secondary education (Department for Education England, 2013), while Australia puts emphasis on STEM (Science, Technology, Engineering and Mathematics) subjects in their curriculum, as Chang (2015) states: "The new curriculum echoes successful programs implemented in the United States such as Code.org and "Hour of Code", with the support of Google and Microsoft, including the United Kingdom who introduced coding in primary schools last year." As CT skills will advance in K-12 education,

university preparatory courses will also need to incorporate elements of CT.

3.2. Elements of Computational Thinking

Just as there are numerous definitions for Computational Thinking, there are also a wide range of ideas on the components of it. Gouws, Bradshaw & Wentworth, (2013) developed a Computational Thinking Framework (CTF) to assist in developing educational materials. According to their framework, the followings skill sets are required for computational thinking: Processes and Transformation, Models and Abstraction, Patterns and Algorithm, Tools and Resources, Inference and Logic, Evaluations and Improvements. The levels of these skills are described as: Recognise, Understand, Apply and Assimilate.

Weese (2017) describes the elements of computational thinking as: Algorithmic Thinking, Abstraction, Problem Decomposition and Control Flow.

According to Barr and Stephenson (2011) the core computational concepts are: data collection, data analysis, data representation, problem decomposition, abstraction, algorithm and procedures, automation, parallelization and simulation.

García-Peñalvo and Mendes (2018) describes computational thinking as “an active problem solving methodology where the students should use a set of concepts, such as abstraction, patterns matching, etc., to process and analyze data, and to create real or virtual artefacts”.

Code.org (2014) illustrates the process of computational thinking in 4 steps: step 1 - decompose, step 2 - patterns, step 3 - abstraction, step 4 - algorithm.

Reviewing the definitions and descriptions of the components of CT skills one can deduct that some common ingredients are: problem decomposition, patterns matching, abstraction and algorithm creation.

4. COMPUTATIONAL THINKING IN THE SUBJECT OF DIGITAL LITERACIES

The aim of the UTS Foundation Studies’ Digital Literacies subject is to equip students with the skills and knowledge to become efficient users of digital devices and applications for academic purposes. The subject employs the components of computational thinking to enhance students’ abilities to tackle complex issues and solve project-based problems in collaboration with others. Other than technical capabilities, the main skills that pre-university students develop in this course are conceptual skills and problem solving abilities.

4.1. The development of CT skills

In the Digital Literacies subject students first complete an unplugged activity developed around the four elements of computational thinking (problem decomposition, abstraction, patterns matching and algorithm creation). The main aim of this learning activity is to enable students to become familiar with the concepts of CT. Through this exercise students develop their knowledge to recognise and understand the process of computational thinking. Following this unplugged activity, in the next two CT activities students work individually to apply their knowledge to practical exercises. They complete two “Hour

of Code” tasks: Minecraft as a beginner coding activity on code.org and to understand programming logic they use the lightbot.com mission. The Hour of Code game is used in the subject to inspire students and raise awareness of the importance of programming, but not to teach students technical coding as concluded by Du, Wimmer and Rada (2016). Light-bot.com is applied in the course as a sound resource to teach students conceptual skills and abstraction, as assessed by Gouws, Bradshaw & Wentworth, (2013) Lightbot has a 74% overall CT score, although, the program is weaker in the areas of recognising patterns and creating algorithm.

4.1.1. Feedback from the students on these activities

55 students completed an evaluation survey of the lesson and the majority of the students (83%) enjoyed the activities and rate them 4+ stars.

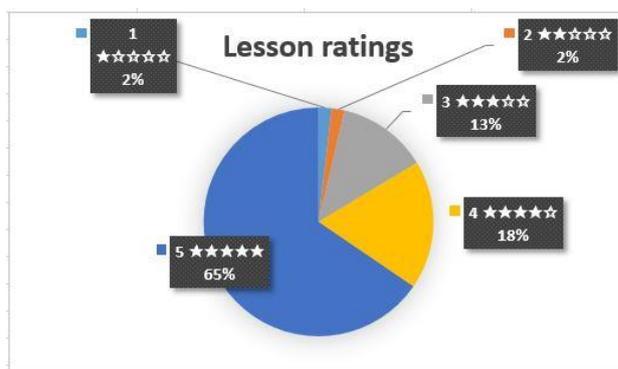


Figure 1. Student rating of Hour-of-Code lesson

The word cloud below serves as a summary of the students’ comment on the lesson.

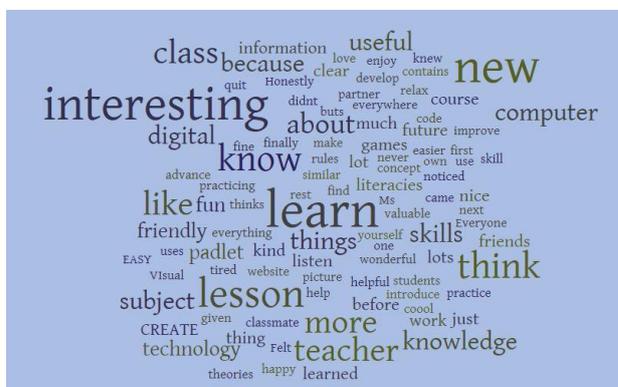


Figure 2. Student comments of Hour-of-Code lesson

4.2. Project-based assessment details

Once students have completed these activities, they progress to the assimilation level of the CT skills and are engaged in a 6-week-long project-based learning assessment. In this assessment students need to employ their CT skills to decompose the complexity of the task, recognise patterns to find similarities within and among tasks, use abstraction to focus on the main issue and create an algorithm to identify details of the process.

The assessment is a collaborative task where students create a 5-minute movie that tells an inspiring story around a digital literacies topic. The project has three deadlines that are set in order to assist students to keep on track with the project:

in week 2 they need to submit a story brief that includes the elements of their short story, in week 4 they do a movie pitch to build anticipation and create excitement about their film in class and in week 6 they submit the final, edited movie. Concurrently, students will acquire skills in the areas of visual literacy incorporating camera angles, shots and staging; in audio editing to record and combine audio tracks and in movie editing to comply a short video.

4.3. Computational Thinking Process in Project-based assessment

Students are required to use the process of CT to complete the assessment. They need to apply the components of CT to create the collaborative project. This experience will assist them in developing their conceptual understanding and team-based problem solving skills.

4.3.1. Decomposing the Problem

The first step when starting an assignment is to break it down into smaller, more manageable tasks where each part can be solved independently of each other. Consistently, that is the very definition of decomposing the problem (Weese, 2017). To provide scaffolding to students in the decomposing process, the project-based assessment has defined deadlines to meet, such as the story brief and the movie pitch. These deadlines create the skeleton for the project and assist students with breaking the assessment into smaller tasks and allocate those to group members. Students need to decide on activities that need to be completed as a group and assign tasks to individuals to contribute to the final product.

This part of the assessment provides the students with great learning value in developing their conceptual skills and understanding the steps in starting an assignment.

4.3.2. Recognising Patterns

During the decomposition stage students will come across tasks that are similar in nature. According to Code.org (2018) pattern is a theme that is repeated many times. Students in this assessment are particularly encouraged to look for patterns when they create their shot list, so scenes that are similar can be organised accordingly. Identifying patterns can be applied to most of the technical parts of the project, such as audio and video recording and editing.

Another example for pattern recognition in the project-based assessment is the use of camera angles and shots to underpin the tone of a scene. For example, to convey emotions students use close up shots to show character's facial expressions; or to illustrate that a character is inferior they use high camera angles. This kind of pattern of shots and angles are used throughout the movie to support the story.

4.3.3. Abstraction

Abstraction refers to the general representation of a complex problem. According to Wing (2008) "The abstraction process - deciding what details we need to highlight and what details we can ignore - underlies computational thinking". The abstraction process allows students to gain a better understanding of the problem they are faced with. It allows them to investigate the core of it without focusing on unnecessary details. It helps them to concentrate on the main idea see what the more important parts of the project are.

In the movie assessment students are required to use abstraction for their movie brief and pitch. In the movie brief they need to outline the main parts of the story, using the five elements of a short story: setting, characters, conflict, plot and theme. They are not required to work out the plot structure in details, rather to provide an overall impression of their story. This abstraction provides the students with two main benefits: form the main idea for their movie and to be able to express themselves in a prompt format.

In the movie pitch the group needs to present their movie idea and a trailer (advertisement of their movie) to the class. The aim of the movie pitch is to generate anticipation and interest in their movie. The groups are required to use abstraction and present their idea in a way that provides enough information for the audience to understand the story without getting into lot of details. Creating a trailer for their movie is an excellent example for students to develop an understanding of abstraction.

Abstraction as well as problem decomposition teaches students to gain an overall view of an issue and develops their conceptual thinking skills. However, these two elements of CT differ. With problem decomposition one breaks through the complexity of a project and creates smaller, more manageable tasks, while with abstraction one gains understanding by removing unnecessary details. For example, when a student is assigned the role of an actor in the group that is problem decomposition, but when that student is trying to understand the personality of that character is abstraction. By reflecting consciously of the similarities and differences between problem decomposition and abstraction students are supported in developing a better understanding of both.

4.3.4. Algorithm

Algorithm refers to solving a problem by developing a set of steps taken in a sequence to achieve the desired outcome (Katai, 2014). The project-based assessment does not focus on developing technical step-by-step instructions for creating a movie, rather it aims to provide the students with the steps to confidently undertake any future projects in their university studies.

At the end of the project, students are required to reflect on the process, identify and evaluate steps that they took in creating their final movie. This assessment hopes to provide the students with the algorithm of successfully solving the challenges of complex, group assignments.

4.4. Feedback from students on the use of CT elements

115 students filled in the final survey after the completion of the project. Their response on the use of CT skills in the project-based learning was very positive, approx. 85% of the students agreed or strongly agreed that they have acquired skills that they will use at university, skills that will help them in to do well in other subjects, and skills in problem decomposition.

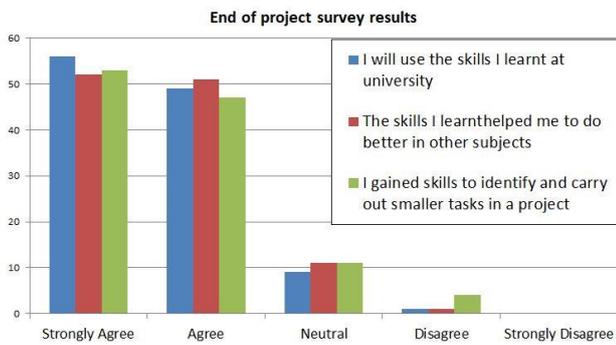


Figure 3. Student feedback on the use of CT elements in project-based assessment

5. CONCLUSION

Computational thinking skill development exercises being incorporated into K-12 Australian curriculum. Therefore, university preparatory courses need to provide similar opportunities for international students to gain knowledge and skills in CT.

Computational thinking components are being used in non-STEM subjects. In this paper, a case of using CT elements in a project-based assessment is presented. It is found that many elements of the CT process can also be applied for project-based learning. Feedback from the students who completed the assessment favours the incorporation of computational thinking into curriculum.

6. REFERENCES

- Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads*, 2(1), 48. <http://dx.doi.org/10.1145/1929887.1929905>
- Chang, O. (2015). *Australian schools are scrapping history and geography and replacing them with coding classes*. *Business Insider Australia*. Retrieved 24 January 2018, from <https://www.businessinsider.com.au/australian-schools-are-scrapping-history-and-geography-and-replacing-them-with-coding-classes-2015-9/>
- Code.org. (2014). *Code.org: Anybody can learn* [online]. Available at: <https://studio.code.org/unplugged/unplug2.pdf> [Accessed 24 Jan. 2018].
- Department for Education England. (2013). *National curriculum in England: Computing programmes of study - key stages 1 and 2*. Ref: DFE-00171e2013. Retrieved from: <https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study>
- Deschryver M. D., Yadav A. (2015). *Creative and Computational Thinking in the Context of New Literacies: Working with Teachers to Scaffold Complex Technology-Mediated Approaches to Teaching and Learning*. *Jl. of Technology and Teacher Education* (23), p411-431.
- Du, J., Wimmer, H., & Rada, R. (2016). "Hour of Code": Can it change students' attitudes toward programming? *Journal of Information Technology Education: Innovations in Practice*, 15, 52-73. Retrieved from <http://www.jite.org/documents/Vol15/JITEv15IIPp053-073Du1950.pdf>
- García-Peñalvo, F. and Mendes, A. (2018). Exploring the computational thinking effects in pre-university education. *Computers in Human Behavior*, 80, pp.407-411.
- Gouws, L., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities. *Proceedings Of The 18Th ACM Conference On Innovation And Technology In Computer Science Education - Iticse '13*. <http://dx.doi.org/10.1145/2462476.2466518>
- Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads*, 1(2), 4. <http://dx.doi.org/10.1145/1805724.1805725>
- Katai, Z. (2014). The challenge of promoting algorithmic thinking of both sciences- and humanities-oriented learners. *Journal Of Computer Assisted Learning*, 31(4), 287-299. <http://dx.doi.org/10.1111/jcal.12070>
- Weese, J. L. (2017). *Bringing computational thinking to K-12 and higher education* (Order No. 10271485). Available from ProQuest Dissertations & Theses Global. (1925537965). Retrieved from <http://ezproxy.lib.uts.edu.au/login?url=https://search-proquest-com.ezproxy.lib.uts.edu.au/docview/1925537965?accountid=17095>
- Wing, J. (2011). Computational thinking. *2011 IEEE Symposium On Visual Languages And Human-Centric Computing (VL/HCC)*, p3, <http://dx.doi.org/10.1109/vlhcc.2011.6070404>
- Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions Of The Royal Society A: Mathematical, Physical And Engineering Sciences*, 366(1881), 3717-3725. <http://dx.doi.org/10.1098/rsta.2008.0118>

應用 Scratch 之運算思維教材設計與教學成效分析

楊智為^{1*}，李政軒¹，郭伯臣¹，謝承晏²

¹國立臺中教育大學教育資訊與測驗統計研究所

²葳弘數位教育

yangcw@mail.ntcu.edu.tw, chenghsuanli@gmail.com, kbc@mail.ntcu.edu.tw, ian@wedtech.com.tw

摘要

本研究主要目的為設計一套應用 Scratch 之運算思維教材，應用在大學課堂中，探討教材實施後的教學成效。實施過程中使用的教學工具為 Scratch，研究結果顯示，經過四週的單元式教學活動，可以顯著的提升學生在運算思維的學習成效。

關鍵字

運算思維；Scratch；教材設計；高等教育。

1. 前言

因科技日益進步，為因應其快速變遷，各國政府在課程的推動上，資訊教育已是其中一項重要的主軸，其中「運算思維 Computational Thinking」是現今各國推動資訊教育的一項重點。電腦運算思考的技巧，並非電腦科學家的專利，而是每個人都應該具備的 (Wing, 2006)，指出像電腦一樣的思考，是所有人都應具備的基本能力，這裡所指的「電腦運算思考的技巧」就是運算思維 (Computational Thinking) 的概念。雖然 Wing 所指的運算思維不僅侷限在電腦相關課程或技能，但目前用以培訓學生運算思維能力的教學模式，編寫程式課程是用來提升運算思維能力的良好方式 (Lye & Koh, 2014)，程式設計教學可以培養學習者之邏輯思考和問題解決的能力 (洪駿命、黃國禎、黃意雯, 2012)，可以透過程式設計課程，建構學生的運算思維概念，已有許多國家正將程式設計課程納入課綱中，如臺灣 107 年課綱、美國和其他如澳洲、英國、法國、愛爾蘭等 16 個國家 (Schoolnet, 2015)。

可見運算思維在未來資訊教育中具相當重要性，學習運算思維較佳方式為程式設計教學，雖目前坊間已有許多程式設計教學的教材或是相關的學習機構，但其設計者的教學方向與預期的教學結果並非透過程式設計方式學習運算思維技能，教學內容多以遊戲設計、程式設計為主題。本研究將以運算思維教學為基礎設計教材，教學工具使用 MIT 開發之 Scratch，並進行教學實驗，探討應用 Scratch 之運算思維教材實施後的教學成效。

2. 文獻探討

2.1. 運算思維內涵

Google (2010) 認為運算思維是問題解決的過程，其包括邏輯排序和資料分析、透過有序的步驟 (或是演算法) 找出問題解答，可應用於所有的學科。Google 並將運算思維分為 11 個概念：Abstraction (抽象化)、Algorithm Design (演算法設計)、Automation (自動化)、Data Analysis (資料分析)、Data Collection (資料蒐集)、

Data Representation (資料表示)、Decomposition (分解)、Parallelization (平行計算)、Pattern Generalization (一般化)、Pattern Recognition (模式辨識)、Simulation (模擬)。

Brennan & Resnick (2012) 提出 TDIA (three-dimensional intergrated)，分為三個向度，分別為 1. 運算概念：序列、迴圈、平行、事件、條件、運算子、資料。2. 運算實踐：增值與迭代、測試與除錯、再用與混合、測試與除錯、再用與混合、抽象化與模組化。3. 運算視野：表達、連接、質疑；Zhong、Wang、Chen & Li (2016) 則提出改自 TDIA 的架構，與 TDIA 一樣為三個向度，但內涵略有調整，分別為 1. 運算概念：物件、指令、序列、迴圈、平行、事件、條件、運算子、資料。2. 運算實踐：計畫與設計、抽象化與建模、模組化與再用、迭代與最佳化、測試與除錯。3. 運算視野：創造與表達、溝通與合作、瞭解與質疑。

綜上所述，運算思維是一種電腦化思考的問題解決過程，各機構或學者所提出的內容有很多相同的部份，本研究礙於授課時數有限及學生程度，因此僅挑選基礎且共通的重要向度發展教學教材，並進行教學實驗。

2.2. 圖形化程式設計軟體 Scratch

Scratch 是由麻省理工學院媒體實驗室 (MIT Media Lab) 的終身幼兒園團隊 (Lifelong Kindergarten Group) 所開發的視覺化程式設計軟體，開發團隊也認為，Scratch 可以幫助使用者創造性地思考，有系統的進行推理並且可協同合作 (MIT, 2007)。目前已有許多應用 Scratch 進行教學的研究，如楊書銘 (2008) 研究結果發現利用 Scratch 教學，對於學生的問題解決能力和部分創造力有顯著提升。Calder (2010) 認為 Scratch 能提升問題解決能力，加強數理概念。Tanrikulu 與 Schaefer (2011) 的研究結果顯示：scratch 具備直觀的介面、易學易用；指令積木化，減少語法錯誤問題 (王秀鶯, 2013)。本研究透過 Scratch 進行程式設計教學，幫助學生在程式設計過程中學習運算思維概念。

3. 研究方法

3.1. 實驗設計

本研究之實驗方法為單組前後測設計，實驗對象為臺灣中部某大學二年級學生，樣本數為 49 人，其中有效樣本 45 人，無效樣本 4 人，教學課程實施四週，每週二節課的實驗教學 (100 分鐘)。實驗前後各進行一週 (60 分鐘) 的測驗 (前測/後測)。

3.2. 研究工具

本研究採用的統計分析軟體為 SPSS (Statistical Product and Service Solutions) 18.0 套裝版本。以成對樣本 T 檢定作為統計方法，用以分析教學成效。

3.2.1. Scratch

本研究採用麻省理工學院開發之 Scratch，Scratch 有網頁版的編輯器和離線版的編輯器，是一種圖形化介面的程式設計軟體，可以使用 Scratch 創造出問答方式或互動式的故事、動畫、遊戲等內容，也可將設計的作品分享至全世界(鄭苑鳳，2014)。

Scratch 的積木功能共分為 10 個類別：動作、事件、外觀、控制、聲音、偵測、畫筆、運算、資料、更多積木等，本研究會透過 Scratch 的這些積木功能進行程式設計教學，並從中進行運算思維教學。

3.2.2. 運算思維教材設計

本研究使用自編的運算思維教材，教材內容使用 Scratch 作為教學工具，共計五個單元，其教學主軸包含六種運算思維概念：物件、序列、迴圈、條件、運算子及演算法設計。期藉由圖像化的 Scratch 教學，讓學生可從教學過程中，習得運算思維的概念。

教材為單元式的課程設計，每一個單元皆以以下教學流程作為設計，課程時間為 100 分鐘：

1. 主題式設計的不插電活動：以運算思維概念為中心，以貼近學生生活經驗之案例分享與思考，用以引起學生學習動機。(10 min)
2. 介紹該單元的運算思維概念，以及在 scratch 中對應的程式積木。(10 min)
3. Scratch 任務設計：基本包含三個任務，基礎任務包含環境設定與主要積木應用，再由基礎逐漸增加任務難度（增加條件），任務內容皆環繞主題設計，最後則附加挑戰的任務，提供給高能力或進階的使用者。(60 min，講解任務與實作)
4. 測驗題是以該單元主題進行設計的診斷試題。(10 min)
5. 總結一回顧與反思。(10 min)

以單元二時尚秀為例，此單元以迴圈（Loops）為運算思維主題，在不插電活動以歌詞中的副歌重複片段、舞蹈中重複的動作、規律圖形中重複的部份來引入概念。

第二部份即以迴圈指令繪出基本圖形，藉以展示 scratch 的功能應用。

而後則進行本單元的任務，分為時尚秀、變換造型、添加任務、挑戰任務。

1. 時尚秀：透過指令讓角色（物件）移動與重複出現（迴圈），如圖 1 所示。

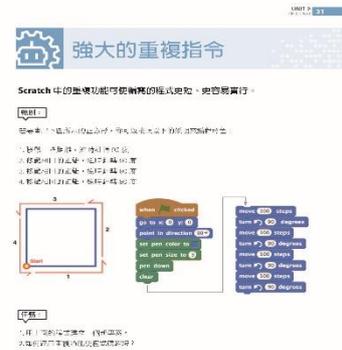


圖 1 重複指令教學活動

2. 變換造型：透過事件執行指令不斷地變換角色造型（迴圈），並讓角色中途暫停。
3. 添加任務：共有三個子任務，分別為添加音樂、變換角色的動作與拍照效果，任務內會使用到物件、指令、迴圈、運算子、事件、演算法設計等。主要是增加主要任務的複雜性，也做為控制教學時間的彈性。



圖 2 添加任務教學活動

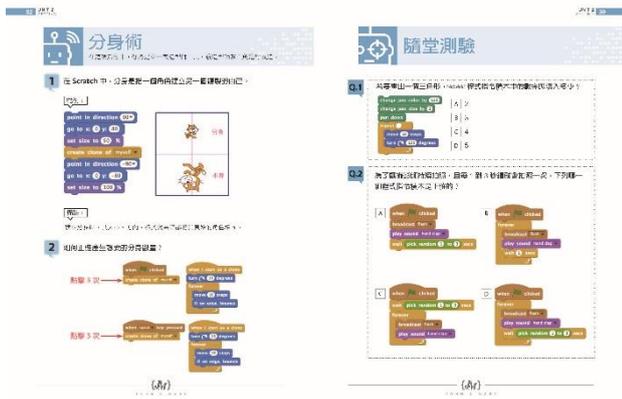


圖 3 挑戰任務與運算思維隨堂測驗

- 挑戰任務：在時尚秀的活動中，利用複製的功能事件，產生角色的分身縮影，並透過迴圈讓此特殊功能可以重複被使用，如圖 3 所示。

完成任務後，提供三題測驗題檢測學生本單元的學習成果，最終為單元總結，進行本單元的回顧與反思。

3.2.3. 運算思維評量工具

本研究使用的評量工具為擷取國際運算思維挑戰賽 (International Challenge on Informatics and Computational Thinking) 試題與自編試題，前測與後測卷各 25 題，為複本試題，試卷涵括了四週授課內容中的運算思維概念及其他的基礎概念。

4. 研究結果與討論

- 本研究的實驗結果，使用成對樣本 T 檢定分析後如表 1 及表 2，由表 2 可見顯著性為 0.014，已達顯著差異，可得知該實驗班級經過四週的 Scratch 運算思維教材教學，已具有教學成效。

表 1 測驗資料敘述統計量

	平均數	個數	標準差	平均數的標準誤
前測分數	51.378	45	18.586	2.771
後測分數	55.911	45	16.080	2.397

表 2 成對樣本 t 檢定表

前測-後測	平均數	標準差	t	自由度	顯著性 (雙尾)
	-4.533	11.927	-2.550	44	.014*

顯著性 (P) <0.05*

- 運用 Scratch 為實作工具，透過自編的運算思維教材，建構學生的運算思維概念，如物件、序列、演算法等，由實驗的成果可知，此教材設計方式應用於運算思維的教學可提升學習成效。
- 對參加的學生進行背景調查，大約一半的學生對於程式設計方面是完全的新手，從未接觸過

程式設計課程，但於後測後，所有的學生皆需繳交一份創意成果報告，全體同學皆已可自行設計簡單的遊戲並具備創意表達能力，多數的學生亦表示希望繼續此類學習課程。

- 在研究限制方面，本次研究受限於教學時間僅有四週，在課程設計上僅能設計較基礎的課程，故在運算思維概念學習成效上，進步較為有限，建議未來可以完整的運算思維架構來設計整體課程，可能會具有更顯著的成效。

5. 致謝

感謝科技部計畫編號 106-2511-S-142 -005 -MY3 給予本研究補助支持，以及感謝伊德文教授權本研究使用與本校產學合作之成果。

6. 參考文獻

- 王秀鶯 (2013)。導入 Scratch 程式教學對國中生自我效能與學習成就之探究～以程式設計課程為例。人文社會學報，9(1)，1-15。
- 洪駿命、黃國禎、黃意雯 (2012)。遊戲創作學習模式對於國小學童在學習動機、問題解決及學習成就之影響。莊紹勇 (主持人)，悅趣化學習與社會，第十六屆全球華人計算機教育應用大會 (GCCCE 2012)，臺灣墾丁福華渡假飯店。
- 楊書銘 (2008)。Scratch 程式設計對六年級學童邏輯推理能力、問題解決能力及創造力的影響。臺北市立教育大學數學資訊教育教學碩士學位班資訊組碩士論文。
- 鄭苑鳳 (2014)。Scratch 遊戲創意設計應用範例集。臺北市：上奇。
- 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.
- Calder, N (2010). Using Scratch: An Integrated Problem-Solving approach to mathematical thinking. APMC, 2010, 15 (4), 9-14
- Google(2010). Exploring Computational Thinking [Web message]. Retrieved from <https://www.google.com/edu/computational-thinking/>
- 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.
- Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books, Inc. Wing, J. M.. Computational Thinking [J]. Communication of the ACM, March 2006/Vol.49, No.3: 33-34
- Schoolnet (2015) Computing our future: Computer programming and coding Priorities, school curricula and initiatives across Europe. Retrieved from <http://fcl.eun.org/documents/10180/14689/Computing+ou>

r+future_final.pdf/746e36b1-e1a6-4bf1-8105-
ea27c0d2bbe0

Tanrikulu, E., & Schaefer, B. C. (2011). The users who touched the ceiling of scratch. *Procedia-Social and Behavioral Sciences*, 28, 764-769.

Wing, J. M. (2006) Computational thinking. *Commun. ACM* 49, 33–35.

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

MIT (2007).Scratch: imagine, program, share. Retrieved from <https://scratch.mit.edu/>

Computational Thinking and STEM/STEAM Education

A DSML for a Robotics Environment to Support

Synergistic Learning of CT and Geometry

Nicole HUTCHINS*, Timothy DARRAH*, Hamid ZARE, Gautam BISWAS*

Institute for Software Integrated Systems, Vanderbilt University

Nashville, TN USA

nicole.m.hutchins@vanderbilt.edu, timothy.s.darrah@vanderbilt.edu, hamid.zare@vanderbilt.edu, gautam.biswas@vanderbilt.edu

ABSTRACT

Synergistic learning of computational thinking (CT) and STEM has proven to be an effective method for enhancing CT education as well as advancing learning in many STEM domains. Domain Specific Modeling Languages (DSML) facilitate the building of computational modeling frameworks that are directly linked to STEM content, thus making it easier for students to focus on concepts and practices. At the same time, teachers can more easily relate curricular content to the model building tasks. This paper discusses the design, development, and implementation of a robotics DSML to support a middle school geometry curriculum.

KEYWORDS

DSML, robotics, STEM, geometry

1. INTRODUCTION

Recent developments show how computational tools have influenced research and practices in mathematics and science education (National Research Council, 2012). In parallel, rapidly evolving educational technologies have influenced pedagogy and curriculum development, primarily by integrating computational tools into the study of STEM disciplines (Grover & Pea, 2013, Hutchins, Zhang, & Biswas, 2017). While the limited availability of skilled teachers, financial constraints on educational institutions, and the inertia in changing current curricular practices has impeded the introduction of Computer Science (CS) courses into middle and high school classrooms, curricula supported by educational software that exploit the synergies between STEM and CT and integrate with current K-12 curricula have found success (Basu, Biswas, & Kinnebrew, 2017; Jona et. al., 2014, Sengupta, et. al., 2013; Weintrop, et. al., 2016).

In the past, model-based design has been employed to facilitate a necessary convergence among physical processes and software control design, thus supporting many Cyber Physical System (CPS) applications (Jackson & Sztipanovits, 2008; Jensen, Chang, & Lee, 2011). In this paper, we extend this design process to Open Ended Learning Environments (OELEs) and focus on the design and integration of curricular scaffolding in OELEs to support student learning in STEM and CS domains.

This paper outlines the development of a WebGME design studio centered on the application of a domain specific modeling language (DSML) for robotics to support a middle school mathematics curriculum. To do so, we analyze the literature and establish curricular and software requirements, describe the design and development of our WebGME

design studio, and conclude with case studies from a usability study.

2. BACKGROUND

To implement a set of learning tasks, while assuring well-formed model realizations (Jackson & Sztipanovits, 2008), we conducted a thorough analysis on the DSML design requirements in combination with the curricular needs of a middle school mathematics classroom. Here we cover four topic areas that directly relate to our research.

2.1. Computational Thinking (CT)

Following Wing's call for the increased introduction of CT in classrooms (2006), significant work was completed towards an applicable definition as well as an outline of key concepts and practices that can be used to assess learning gains in CT. The Royal Society defined CT as "*the process of recognizing aspects of computation in the world that surrounds us and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes*" (Royal Society, 2012). In Grover and Pea's systematic review (2013), the authors listed essential CT constructs and, for the purposes of our work, we have focused on flow of control, decomposition, efficiency and performance constraints, and debugging.

To facilitate CT and the acquisition of basic geometry skills, appropriate scaffolding must be incorporated into the design of the DSML. Significant success with synergistic learning of CT and STEM disciplines through the use of block-based DSMLs (Hasan & Biswas, 2017) has supported increased integration of this style of programming at the K-12 level and we seek to extend this effort through the use of a DSML created in a model-based design environment such as WebGME. In our platform, CT provides the framework for building computational models or algorithms to define and debug the movement of robots. The metamodel and model building visualizer described in Section 5 provide a level of curricular abstraction that eliminates many of the burdens of text-based programming. In addition, our model-based design environment is supported by a necessary utilization of CT constructs, such as debugging and problem decomposition.

Furthermore, our robotics platform provides multiple representations with the utilization of a physical robot (as opposed to a virtual sprite), a physical coordinate plane, and a bird's eye view of the grid space with several overlays (e.g., movement traces, lines, points, etc.). Abstraction is provided in the model building visualizer that the student uses to construct their command sequence. As pointed out above this combination of representations and abstractions

is desired so that a student is fully capable of systematically processing their solution or debugging a problem utilizing a CT approach (Basu, Biswas, & Kinnebrew, 2016).

2.2. General Robotics Courses

Many schools offer after school programs or summer camps using VEX® or LEGO Mindstorms® robotic kits. These kits come with a substantial amount of supporting information and resources including forums, tutorials, and fully executable curriculum sets. Hendricks et al. (2012) and Panadero et al. (2010) report an increase in computational thinking activities and learning outcomes when students use these kits. Other robotics courses offered as summer camps have been successful in increasing student engagement, motivation, teamwork, critical thinking, and problem solving (Darrah, Kuryla, & Bond, 2018; Goldman, Eguchi, & Sklar, 2004; Ansorge & Barker, 2007), all directly related to the application of CT constructs in a STEM domain.

2.3. Robotics in Mathematics

Barreto & Benitti (2012) noted that activities which integrate robotics into a math or science classroom should “*possess a high-level of structure that helps the robot to correctly guide the activities and the students through them,*” and that self-directed activities that “*promote personalized comprehension of STEM concepts through experimentation*” showed significant success - and added support for our approach in this domain as design space exploration activity. Our DSML has been highly scaffolded as a means of supporting these robotic integration requirements. In addition, the experimentation requirement is further supported through the display of curricular feedback following the execution of a robot sequence, to be described in Section 6.

Two recent studies were carried out by researchers from NYU that explored the use of a robotic agent to teach geometry to middle school students (Muldner, et. al., 2013; Giroto, et. al., 2016). Their environment consisted of a projector, a LEGO Mindstorms® robot, and two iPods for communication. These studies highlight the effectiveness of a tangible learning environment (TLE) in terms of delivering a much richer learning experience than traditional classroom methods. Moreover, TLEs have found considerable success in fostering creativity (Goldman, Eguchi, & Sklar, 2004), a benefit to our design space exploration approach, while also increasing motivation (Windham, 2007).

2.4. Domain Specific Modeling Language (DSML)

Van Deursen defines a domain specific language as “*a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain*” (2000). Typically, DSMLs are developed to facilitate the work of domain experts in application tasks. But they can also play an important role in helping learners focus on domain concepts when building models and solving problems in the domain. In our work, the DSML developed allows a student to define a set of instructions for a robot to solve middle school mathematics problems that are centered on concepts derived from coordinate geometry and solving path planning problems.

The benefit of developing a DSML is the affordability it creates in curricular implementation and expansion. Students can “*express and develop solutions ... at the level of abstraction of the target domain,*” “*build programs that are concise and self-documenting,*” and “*verify and validate models and results generated from the models*” (Hasan & Biswas, 2017). This provides a highly structured environment that enables the student to experiment with various solutions in a self-directed manner. This structure comes in part by how the model building environment is presented to the student (visualizer), how the model blocks themselves appear (decorator), and how the model is executed on the robot (communication plugin), to be detailed in Section 5.

Jackson and Sztipanovitz (2008) highlight three applications of DSML syntax: model transformations, correct-by-construction, and design space exploration. In the context of an educational setting, students engage with a robotics-based design studio to learn mathematics and CT concepts by performing tasks with their robots. The syntax our DSML most closely supports is the notion of design space exploration. This enhances “*the expressiveness of metamodeling constraints*” and “*the ability “to project behavioral properties on the syntactic level”*” (2008). Our robotics DSML supports model building and problem solving with robotics in a way that students can seamlessly learn domain and CT concepts and practices.

As it relates to our DSML development, we aimed to simplify the interactions between the robot and the students, so they may focus on learning the required mathematics and geometry concepts and applying them to planning and problem-solving tasks. An added goal is to provide for easy exploration within the domain, so that the open-ended nature of the learning is retained, and students can learn through the direct application of CT practices such as model construction and algorithm development.

Finally, as an educational product, it is imperative to understand the ramifications this implementation has on teacher curriculum development and productivity in the classroom. In Tennessee, the licensure and examination process does not require any assessment of computer science or CT knowledge (The Praxis Study Companion, 2017). As such, we assume limited CS experience of middle school mathematics teachers. To account for this, our DSML can be tailored at the classroom level to account for the capabilities of the teacher. This flexibility eases the transition from learning the system to learning the instructional material the system delivers.

3. CURRICULUM DEVELOPMENT

Understanding how students conceptualize, acquire, and retain geometric concepts must be understood in sufficient detail before designing a curriculum in conjunction with a TLE. Burger and Shaughnessy (1986) concluded that there are five major stages to student’s understanding of geometric concepts: visualization (pure visual reasoning), analysis (based on visualization), abstraction (understanding the properties), deduction (formal reasoning), and rigor (comparing different systems). Students are not typically

exposed to deduction or rigor until a high school geometry course.

We focus primarily on visualization, analysis, and abstraction by introducing a new concept with a description, graphic, and how this topic is relevant in a student's everyday life. Then we provide a set of problems in which the student must give the robot the correct information so it can achieve its goal. Geometric properties and definitions are introduced with their respective problems, and students are required to not only demonstrate mastery by generating the correct command sequences, but also with summative assessments at the end of each module. Below is a sample curriculum outline that is well suited for middle school geometry:

- 1) Coordinate Plane (Axis definitions, Points)
- 2) Lines (Properties, Line segments, Slope, Midpoints)
- 3) Shapes (Properties, Squares, Rectangles, Triangles)
- 4) Path Planning (Shortest path reasoning, Manhattan distances, Straight line distances)

As described in the introduction and requirements, our goal with the development of a robotics DSML was to provide the basis to enable an engaging, applicable curricular unit for a middle school mathematics classroom that connects the computational modeling task to modeling and problem solving in geometry. Our new learning environment promotes knowledge acquisition through a hands-on, visual-feedback approach that is consistent with the design of TLEs (Darrah, Kuryla, & Bond, 2018) and linked to the visualization, analysis, and abstraction stages of geometry understanding described by Burger and Shaughnessy. Our development of a model via WebGME (given the abstraction afforded in the DSML) with the added benefit of watching a real-life robot complete the programmed paths allows for easy applications of CT and geometry constructs and students will be more motivated by the experience.

As it pertains to CT learning gains, our curriculum is most applicable to the assessment of students' knowledge and abilities in implementing algorithms, understanding and addressing efficiency and performance constraints, and debugging. These practices, as defined by Grover and Pea (2013), are utilized in each curricular task designed to target the elements provided in the curriculum outline, above, as students are required to use our scaffolded DSMLs in a sequential order given physical and command constraints of the robot in order to complete each task. We surmise that the repetitive use of these practices to solve geometry problems will enhance students CT abilities for these practices.

4. ENVIRONMENT

With the establishment of our system requirements, the second step in our process was to design and develop our system environment. Our robot operates on a 7ft by 7ft platform that has been sectioned into a 10x10 grid. The robot is equipped with sensors that allow it to track its location on the grid. As such, if it is told to move forward by 3, the robot will travel forward until it has reached the third black line that is perpendicular to the direction the robot is moving. A video camera set-up is centered above the grid as shown in the figure. The video feed generated can be used by the

student or a teacher to track the robot as it moves along a path and verify the correctness of the path.

4.1. Robot

When activated, the robot starts a TCP server to communicate with the WebGME plugin and opens a serial port to communicate with the Arduino MCU. It manages these processes on separate threads. The main thread manages the various modes the user can utilize to control the robot, such as manual mode, sequence mode, or GME mode (the mode used in conjunction with this paper). The MCU runs one program that takes input from 3 IR tx/rx modules (line following sensor) and its output controls the motors. It communicates with the SBC as well to provide feedback for received commands and for mode switching. Figure 2 provides an overview of the modular system architecture.

The robot communicates with WebGME using the cross-platform *socketio* library. The plugin generates a JSON formatted string that is parsed within a minimal Flask web server running on the robot. Upon receipt, the Arduino MCU executes the command sequence and signals to the RCM when it is finished.

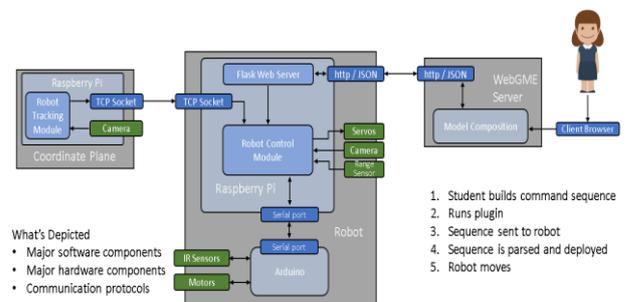


Figure 2. System Architecture

5. META-MODEL

As previously described, the utilization of a DSML provides curricular benefits in that it is constructed at a suitable level of abstraction to allow the learner to focus on what is important, and abstract away other CS details (e.g. syntax concerns). Through the analysis of geometry and CT requirements, our meta-model (Figure 3) was developed based on the implementation of four goals:

1. a scaffolded, curricular driven approach that focuses student actions on the concept(s) being addressed;
2. a simplified integration of robotics and mathematics that makes it easier for the teacher to follow the student work and assess it;
3. scalability in the classroom context; and
4. a systematic, stable connection between the robot environment and modeling environment that is easy to understand.

The students' problem-solving tasks (e.g., building shapes, following paths) are scaffolded, as exemplified through the four available commands. The reduced set of commands allows students to focus on the planning and computational components of their activities. In addition, the organization of the commands and sequences showcases the model's potential scalability and ease-of-use for the teacher.

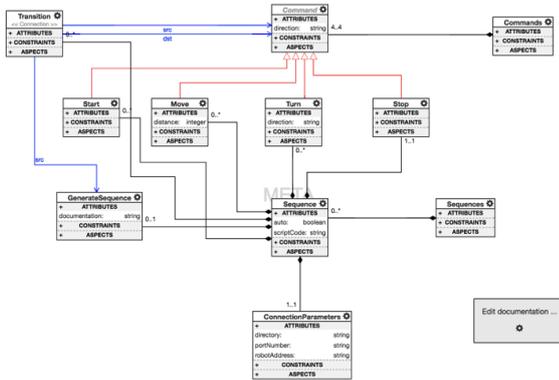


Figure 3. Robotics Meta-Model

5.1. Decorator

The target audience for this activity includes middle school students that may not have any programming experience. As such, the visual component of the environment may play a role in the motivation and buy-in of students, regardless of their capabilities, which is directly linked to positive learning outcomes. A Decorator is a component of the WebGME Design Studio that alters the way a node in the model looks in composition view (the student’s view). Figure 5 provides a zoomed-in image of relevant decorator components. Students can select the next command in their sequence via a drop-down menu located on the current node. When a command is selected, the transition between the two nodes is automatically created. In addition, each node contains the command name, attribute value, and an image – not only allowing for multimodal learning acquisition, but also easing the debugging process described in Section 2.2.



Figure 5. Model Decorator

5.2. Plugin

The final component needed to configure our WebGME design studio is the plugin that coordinates the compilation and delivery of the sequence of commands implemented by the student to be executed by the robot. In other words, the JavaScript plugin sends the visually represented sequence of commands to the robot in a machine-readable format. In the making of the plugin, we defined three requirements: Parsing the student defined command sequences into a standard structure, validating the sequence alongside reporting the errors, and finally, sending the commands to the robot.

Upon starting a session, the plugin connects the editor environment with the robot using the parameters defined in the “Connection Parameters” node. This is achieved through a one-to-one socket connection, which remains open until the user ends the session. To make sense of the visual chain of commands the plugin starts by querying the sequence to find the start node. It then records this block and its relevant attributes. Next, the outgoing connection is followed to similarly parse the next blocks until the stop command is reached. This information is then stored in JavaScript Object Notation (JSON) format and sent to the robot by emitting a submission event that the robot is listening for. The robot

then parses the sequence and executes the commands as detailed above.

6. Implementation

Following the development and design of the robotics studio and accompanying geometry curriculum, we had three middle school students complete the designed tasks as a means of testing the system and getting feedback on ease-of-use and system benefits or drawbacks. In this section, we present an application of our system in a classroom environment and demonstrate the use of the robotics design studio as a tool to complete a sample path planning module at the middle school level.

6.1. Sample Problem Set

A subset of the curriculum described in Section 3 includes three general problems:

- 1) Identifying the axes and positive or negative values
- 2) Plotting points given (x,y) and deriving (x,y) from a set of points
- 3) Path planning with multiple points, calculating the shortest Manhattan distance

Figure 6 illustrates the visual interface that provides the instructions for each task along with the overhead webcam feed in conjunction with the WebGME design studio. In this assignment, students are tasked with finding the most efficient path the robot can take ensuring stops at the police station, the fire station, and the courthouse prior to ending its trip at the post office. Typically, this type of assignment at the introductory level is distributed as on paper, limiting the multi-modal approach to learning that may benefit certain students.



Figure 6. Virtual Interface for Example Path Planning Problem

The direction the robot is facing, its current location, and number of spaces moved are displayed at the top of the information section which helps the student during the solution construction process. The problem is given below that, along with various hints that are given at predetermined times.

In the scenario shown in Figure 6, the student first identified the coordinates of all locations the robot must visit. When all points are correctly located, their coordinates are shown on the video feed. From the image provided, it can be seen that the student then completed a shortest path problem in which they generated the correct command sequence for the robot to visit all locations, starting at the Amazon warehouse $(2, -2)$. The automatic feedback response of “Nice Work!” is shown – demonstrating the successful completion of the task

9. REFERENCES

- Anderson, J.R., Boyl, C.F., Corbett, A.T., Lewis, M.W. (1990). Cognitive Modeling and Intelligent Tutoring. *Artificial Intelligence - Special issue*, 42-1.
- Ansonge, J., Barker, B. (2007). Robotics as a Means to Increase Achievement Scores in an Informal Learning Environment. *Journal of Research on Technology in Education*, 39-3.
- 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.
- Basu, S., Biswas, G., Kinnebrew, J. (2016). Using multiple representations to simultaneously learn computational thinking and middle school science. *Proceedings of the 30th AAAI Conference on Artificial Intelligence*.
- Benitti, F. & Barreto, V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3), 978-988.
- Burger, William F., Shaughnessy, J. Michael: Characterizing the van Hiele levels of development in geometry. *Journal for research in mathematics education*, p. 31-48. (1986)
- Darrah, T., Kuryla, E., & Bond, A. (2018). Improving STEM Education with an Open-Source Robotics Learning Environment. *Proceedings of the Hawaii International Conference on Education*.
- Giroto, V., Lozano, C., Muldner, K., Bursleson, W., Walker, E. (2016). Lessons Learned from In-School Use of rTag: A Robo-Tangible Learning Environment. *Proceedings of the ACM Conference on Human Factors in Computing Systems*.
- Goldman, R., Eguchi, A., Sklar, E. Using Educational Robotics to Engage Inner-City Students with Technology. (2004). *Proceedings of the 6th International Conference on Learning Sciences*, 214-221.
- Grover, S. & Pea, R. (2013). Computational Thinking in K-12: A Review of the State of the Field. *Educational Researcher*, 42(1), 38-43.
- Hasan, A. & Biswas, G. (2017). Domain Specific Modeling Language Design to support Synergistic Learning of STEM and Computational Thinking. In *Proceedings of the International Conference on Computational Thinking Education*.
- Hendricks, C., Alemdar, M., Olgetree, T. (2012). The Impact of Participation in Vex(R) Robotics Competition on Middle and High School Students' Interest in Pursuing STEM Studies and STEM-Related Careers. *American Society for Engineering Education*.
- Hutchins, N, Zhang, N, & Biswas, G (2017). The Role Gender Differences in Computational Thinking Confidence Levels Plays in STEM Applications. In *Proceedings of the International Conference on Computational Thinking Education*.
- Jackson, E. & Sztipanovits, J. (2008). Formalizing the Structural Semantics of Domain-Specific Modeling Languages. *Software & Systems Modeling*, 8(4), 451-478.
- Jensen, J. C., Chang, D. H. Lee, E. A. (2011). A Model-Based Design Methodology for Cyber-Physical Systems. *Proceedings of the IEEE Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*.
- Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *future directions in computer science education summit meeting*, Orlando, FL.
- Muldner, K., Lozano, C., Giroto, V., Bursleson, W., Walker, E. (2013). Designing a Tangible Learning Environment with a Teachable Agent. *Artificial Intelligence in Education*.
- National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press.
- Panadero, C., Villena-Roman, J., Delgado-Kloos, C. (2010). Impact of Learning Experiences Using LEGO Mindstorms(R) in Engineering Courses. *Proceedings of the IEEE Global Engineering Education Conference*.
- Papert, S. (1993). *Mindstorms: Children, computers, and powerful ideas* (2nd ed.). New York, NY: Basic Books.
- Royal Society. (2012). Shut down or restart: The way forward for computing in UK schools. Retrieved February 4, 2017, from <https://royalsociety.org/topics-policy/projects/computing-in-schools/report/>
- Sengupta, P., Kinnebrew, J.S., Basu, S., Biswas, G., & Clar, 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.
- The Praxis Study Companion - Mathematics: Content Knowledge. ETS, 2017.
- Van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: An annotated bibliography. *Sigplan Notices*, 35(6), 26-36.
- 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*, 1-21.
- Windham, C.: Why Today's Students Value Authentic Learning. (2007). *Educause Learning Initiative - Advancing Learning Through IT Innovation*.
- Wing, J. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-36.

Introducing Computational Thinking Across the Curriculum with Virtual Reality

Merijke COENRAAD*, David WEINTROP
University of Maryland
mcoenraa@umd.edu, weintrop@umd.edu

ABSTRACT

Computational thinking is increasingly important within modern society. It is essential that K-12 students are introduced to the powerful ideas of computational thinking and given opportunities to develop their computational thinking practices. Virtual reality (VR), a technological tool with increasing prevalence in society and schools, has the potential to widen computational thinking exposure for students not only in STEM environments, but also across the curriculum. Virtual reality is an engaging medium that has been shown to increase student learning. In this paper, we argue that virtual reality can serve as an effective means for helping students develop computational thinking practices related to systems thinking, data practices, and modeling. To do this, we present virtual reality-based lessons used in a classroom and show how they promote the development and use of computational thinking practices. These lessons are accompanied by findings reporting students' impression of virtual reality use within the classroom. The contribution of this work is in showing how virtual reality can serve as a possible means to integrate computational thinking within existing classrooms, thus giving students added exposure to these essential practices.

KEYWORDS

Computational Thinking, Virtual Reality, Computational Thinking Across the Curriculum

1. INTRODUCTION

The focus on computational thinking has expanded over the last decade. While the central ideas of computational thinking have been around for decades (Papert, 1980), Wing's (2006) call for the importance of computational thinking has renewed enthusiasm for the idea. In response to this call, educators at all levels have increased focus on computational thinking both within computer science courses and in other subject areas. With the acknowledged need for further computational thinking education and the expansion in the number of subjects that can incorporate computational thinking, new learning opportunities are being continuously developed. As the number of jobs requiring coding and technology is expected to increase over the next decade, a growing need to educate students in computational methods has emerged, especially within the Science, Technology, Engineering, and Mathematics (STEM) fields (Weintrop et al., 2016). At the same time that computational thinking education is increasing within schools, virtual reality equipment is becoming more affordable (Greenwald et al., 2017) and, therefore, more available to schools. Researchers have long found benefits to using VR and virtual environments including increased interest, motivation, and learning (Limniou et al., 2008). With its ability to "improve learners' ability of analyzing problems and exploring new concepts" (Pan et al., 2006,

p.20) and equipment being more available, VR could be a conduit for teaching computational thinking practices within a variety of subjects.

In this paper, we argue that VR can serve as one potential way to introduce foundational computational thinking ideas and practices to learners. We will discuss the benefits of incorporating VR into the classroom and explore ways that such inclusion could serve as an opportunity for the development of computational thinking. We will present VR as a mechanism for students to engage with computational thinking through examples of lessons that have been taught using VR and potential computational thinking practices that can be developed using similar lessons. Data from students reporting their impressions of VR will also be presented. The importance of computational thinking education is clear, our focus must now shift to methods through which computational thinking can be brought into the classroom.

2. BACKGROUND

2.1. Computational Thinking

Wing (2008) defines computational thinking as "taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing" (p. 3717). It focuses on how not only computers, but also humans can think and solve problems, specifically detailing the concepts that are used in problem solving and interactions, not the software or hardware that are developed (Wing, 2006). Computational thinking shifts the focus of computing from emphasizing computer programming skills to focusing on the principles of computing (Wing, 2008). This shift in emphasis does not seek to redefine the discipline of computing, but rather to help clarify the importance of computing and the knowledge one needs to effectively use it, drawing it out of a focus solely on computer programming and making connections between the existing principles of computing and themes that exist within current curricula (Henderson, Cortina, & Wing, 2007).

Computational thinking is ubiquitous in the modern era. According to Henderson et al. (2007), "computational reasoning is the core of all modern Science, Technology, Engineering and Mathematics (STEM) disciplines and is intrinsic to all other disciplines from A to Z" (p. 195). It has changed the way that work is completed, no matter the field (Barr, Harrison, & Conery, 2011) and these changes create a need to introduce computational thinking into classrooms, preparing students to be a part of the modern workforce (Weintrop et al., 2016). Although the number of undergraduates who are exposed to computational thinking has already increased, bringing computational thinking in the K-12 realm would have a greater impact on the number of students who are reached (Settle et al., 2012) and exposure to computational thinking early will help students

to have greater success when taking later computer science and computational thinking courses (Grover & Pea, 2013). Computational thinking activities allow “computational representations to make significant shifts in the way students learn, think, and practice science and mathematics” (Orton et al., 2016, p. 706), making them extremely important for K-12 students. The growing importance of computational thinking is signaled by its inclusion in the Next Generation Science Standards and points to the connection between mathematics, science, and computation (NGSS Lead States, 2013). Barr et al. (2011), argue that computational thinking is essential skills across K-12 curricula. They describe the importance of data collection, analysis, and representation within social studies and language arts for the analysis of historical events and linguistic patterns, algorithms and procedures for the writing of instructions and decomposition supporting the development of outlines, and simulations enabling reenactments for learning across the humanities. It might often be associated with STEM fields, but computational thinking can be applied to any content, as argued by Orton et al. (2016), by “having students employ these practices to various problems in diverse content areas, we can reinforce the broad applicability of these skills while both providing students concrete contexts to employ them” (p. 710).

Within this paper, we will use the computational thinking taxonomy developed by Weintrop et al. (2016). In this framework, computational thinking is broken into four separate, yet interconnected categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. The taxonomy acts as a guide for teachers as they incorporate computational thinking into classrooms, allowing for both a deepening of content understanding and an authentic environment in which to learn computational thinking practices (Weintrop et al., 2016). This paper focuses on three of the taxonomy’s categories: data practices, modeling and simulation practices, and systems thinking practices. Data practices include the collection, creation, manipulation, analysis, and visualization of data. Modeling and simulation practices consists of using models to understand concepts, find and test solutions, and assessing, designing, and constructing models. The category includes working with both models that others have generated and student created models. Systems thinking practices pertain to the investigation of a complex system as a whole, examining the relationships within a system, thinking in levels, communicating information about a system, and defining systems and managing complexity in order to examine individual parts of the system and how the system functions in its entirety (Weintrop et al., 2016). This taxonomy is useful for this work because the taxonomy’s goal is “not to radically change the existing practices of experienced teachers; instead...[it] serve[s] as a resource for augmenting existing pedagogy and curriculum with...computational thinking practices” (Weintrop et al., 2016, p. 129).

2.2. Virtual Reality

According to Huang, Rauch, and Liaw (2010), “virtual reality (VR) is understood as the use of 3D graphic systems in combination with various interface devices to provide the effect of immersion in an interactive visual environment” (p.

1172). There are many different types of VR: virtual environments include those on desktop computers controlled by mice and keyboards, projection based VR systems that project on an image at room scale, and head mounted visual displays (Greenwald et al., 2017; Limniou et al., 2008). Many have discussed the potential and success of VR due to its engaging nature and ability to transport students to locations where they cannot physically travel, whether due to physical, time, or money constraints, to rare experiences, or to gain access to experts (Greenwald et al., 2017). Especially since VR has had previous success in education and training environments, the increased availability of VR in the internet-age is only expected to bring it further success and new users and creators (Greenwald et al., 2017).

Virtual reality has the potential to enable learning experiences not possible with other, low-tech methods (Greenwald et al., 2017). This ability provides students with an immersive experience in an environment with which they can react, giving virtual environments the potential to lead students to knowledge construction (Winn, Windschitl, Fruland, & Lee, 2002). Multiple studies have shown that participating in VR activities increases student knowledge. For example, Limniou et al. (2008) demonstrated through chemistry and the observation of molecules that participating in 3D animation environments within a room-based VR projection led students to better comprehend molecular structure and changes based on chemical reactions as compared to students who used a desktop based 2D animation. The VR experience allowed students to develop a better sense of the volume of objects within a space. A second example can be seen with Merchant, Goetz, Cifuentes, Keeney-Kennicutt, and Davis (2014), who found that games, simulations, and virtual world were all successful in increasing learning outcomes. Even desktop virtual environments, although they are not fully immersive, enhanced learner engagement. Pan et al. (2006) describe the successful use of virtual reality in a number of different contexts including the use of synthetic characters to train students in group work, simulate peace keeping missions, and promote and enable storytelling. In all, participation in VR can lower the cognitive load that users are experiencing because the simulation is so real, enabling more learning to occur (Huang et al., 2010). Virtual environments can also support experimental and constructivist learning. Students are drawn to VR because it provides them with the opportunity to have first-person experiences. Students report feeling as though they are inside the phenomena being studied. This allows students to build their knowledge based on personal experiences (Limniou et al., 2008). Constructivist experiences occur by students situating themselves within a real situation and doing a realistic task, interacting with objects and events within virtual worlds, and using characters and avatars to learn through role playing (Huang et al., 2010).

For the purpose of this paper, we will be discussing the use of a stereoscopic, head mounted VR system. This means that users use VR goggles that block out the classroom environment and “provide to the eyes of the viewer two different images, representing two perspectives of the same object, with a minor deviation similar to the perspectives that both eyes naturally receive in binocular vision” (Limniou et

al., 2008, p. 585). The use of full immersion increases the benefits of VR by elevating interest and motivation while encouraging observing from various perspectives, active participation, and the asking of questions (Limniou et al., 2008). This leads “immersed students [to] learn more than non-immersed students” (Winn et al., 2002, p. 497). Immersed students also feel as though they are more “present” to the learning environment, leading them to take longer to complete the task and to say more as they are working. The immersion is especially beneficial when encountering concepts that are supported by the ability to look around and examine the surroundings (Winn et al., 2002).

3. METHODS

Alongside exploring potential instructional opportunities around bringing computational thinking into classrooms through VR, this paper presents data related to students’ experiences of using VR in their classrooms. The data was collected by a teacher in the middle school of a Pre-K – 8 religious school in the Northeastern United States. The mission of the school includes serving immigrant families resulting in a diverse student body with 70% of students receiving financial aid. After participating in classroom lessons using VR technology for a year, students were asked to complete questionnaires asking about their experiences with the technology. Altogether, 65 students participated in the study: 22 6th grade students, 29 7th grade students, and 14 8th grade students.

The lessons that students participated in were taught by multiple teachers across subjects including science, Spanish, social studies, and religion. Virtual reality was used within the existing curriculum to enhance understanding of topics already present and gave students proficiency both as participants in lessons and acting as the guides leading other students on VR trips. A subset of these lessons are presented in this work. For teacher-led activities, the lessons generally took place during a single 45-minute period while student-led activities were usually part of larger projects that allowed students more time to find their VR component and present it to the class. While computational thinking was not a focus of the instruction, in this paper we highlight potential synergies and design opportunities for this integration.

Students used handheld Mattel ViewMaster headsets with Asus ZenFone 2 devices (Figure 1). Most VR experiences were facilitated through the Google Expeditions App, but students also participated in a few activities facilitated by Google Street View and YouTube. After multiple exposures to the VR technology, students completed an end of the year survey detailing their impression of the technology and their learning from it. The survey that students completed was hosted online and students were asked to complete it as part of their end of the year activities. Students were aware that their teacher would see their responses and that aggregate data from the survey would be shared with the grant agency that funded the purchase of the VR equipment for the school.



Figure 1. Mattel ViewMaster headset with Asus ZenFone 2 on the Google Expeditions platform

4. INTEGRATING COMPUTATIONAL THINKING AND VIRTUAL REALITY

Within the classroom, VR has the potential to engage students in computational thinking. Due to the unique perspectives and interactions enabled by VR, students are able to view places and interact with objects not typically accessible to K-12 students, creating opportunities for developing important computational thinking practices. In this section, we detail potential ways to integrate VR and computational thinking across the curriculum. We conclude with a brief report of student perception on the use of VR in their classrooms. While VR technology was used with students throughout the school year and in a variety of subjects, here we present lessons from the science, Spanish, and social studies classrooms. These lessons are intended to demonstrate how students used VR equipment within the classroom and the computational thinking development that can occur through these lessons. This is not an exhaustive list of potential ways to integrate computational thinking via VR but serves as a demonstration of what it could look like to blend the two.

4.1. Computational Thinking in Science Class

VR was used to investigate both the cells and systems of the body during a 7th grade life science course. While studying the parts of a cell, the teacher guided students on a virtual tour of the cell through the *Into the Cell* Google Expedition (Figure 2). Each student received a VR device and was able to look on his/her own as the teacher led students through the series of computer generated images. Students were asked to identify various parts of the cell as the teacher pointed them out and the class together discussed cell functioning. Students were given time to use their devices to look around the image and work on their own to explore how the various parts of the cell come together and exist in relation to each other. In later classes, the teacher referred back to the VR experience, giving students the opportunity to recall their observations and apply them to their learning throughout the unit.



Figure 2. Into the Cell virtual expedition.

This lesson demonstrates a possible use of virtual reality to engage learners with modeling and simulation practices

including using computational models to understand a concept, using computation models to find and test solutions, and assessing computational models (Weintrop et al., 2016). Although students are not able to create models within the virtual environment, they are able to think critically about the bounds of model, assess what is included and excluded, discuss how the technology represented the phenomena, and answer questions through scientific inquiry gaining information from the model. Within this specific lesson, the use of models in VR allowed students to better visualize a cell while discussing the various parts of cells and their interactions. The ability to examine cell structure from all angles allowed students to instigate shape, proximity, and size in a way that is not possible through basic images. Further, students were better able to examine the relationship between parts of a cell. In this way, the VR context enabled new ways of learners engaging with the computational thinking practice of using computational models to understand a concept. As one student stated, “[virtual reality] helped me learn, because when we read straight from the textbook you can’t really visualize what you are reading. However now with the virtual reality you can.” Another student commented on how VR “helped me see the world in a different way...i never actually knew what was inside a cell, but [with virtual reality] i felt like i was living a part of that world.” These virtual, computational models were pedagogically useful in their ability to explain the relationship between parts of a cell and allowed students to use and interpret scientific models.

In a later unit, students used VR to study the systems of the body. Students were responsible for working in small groups to complete an in-depth study of one body system and present it to the class using VR to demonstrate their findings. To conduct their research, students used the VR devices and the Google Expeditions platform. The expeditions that students selected used computer graphics to demonstrate the various body systems from inside the body and used images to move through the system. Some of the expeditions that were selected also allowed students to demonstrate how the body system worked by demonstrating functionality based on their purpose, such as showing the spread of viruses. This activity provided opportunities for students to engage in systems thinking practices such as investigating a complex system as a whole, understanding the relationships within a system and thinking in levels (Weintrop et al., 2016).

Although the ability to develop systems thinking practices by studying the systems of the body and their interaction is possible through other methods, VR acts as an excellent conduit for this knowledge. Systems thinking practices include the ability to both view a system as a whole rather than simply as individual parts as well as to think in levels and move between different perspectives on the same system (Wilensky & Resnick, 1999). Virtual reality excels in supporting such practices as it allows for both an in-depth study of a system as a whole and studies of individual pieces and how those pieces interact. As with the use of VR in the study of cells, VR allows students to enter locations they would not be able to such as inside the lungs or the middle of the digestive system. With their 360 views of each of these parts, students are able to look at individual elements separately to investigate the behaviors they promote, two

important parts of systems thinking (Weintrop et al., 2016). Overall, systems thinking learning can be enhanced by studying those systems through VR and the unique views it enables.

4.2. Computational Thinking in Social Studies Class

While studying the American Civil War (1861-1865), students in the 8th grade used VR to visit Smithfield Plantation in Virginia. A plantation is a large farm or estate that grows a single crop. In the United States during this time, slaves were the primary form of labor on plantations. This virtual field trip allowed students to explore a plantation as part of their learning about life in the South and slavery. Prior to this field trip, students had spent time studying the development of slavery in the United States and life in the northern United States.

Using the Google Street View platform, students were given five minutes to “walk” around the plantation and make observations. After the time spent investigating the plantation individually, students shared their discoveries with the class and the entire class was given time to find the locations discovered by classmates. The following day, these observations were used as the class continued to talk about life on plantations and students were able to reference the physical landmarks of the plantation they saw as well as the differences between the plantation house and the cabins and quarters visible from the roads.

Since the taxonomy was created for use with mathematics and science courses, we diverge from it slightly while talking about social studies, but it still serves as a useful resource with regards to discussing computational thinking across subjects, especially with respect to the treatment and analysis of data. New technologies have enabled not only the collection of data to change, but also how those data are viewed and the connections that are made with them. Students need to learn to draw meaning from data rather than expecting the data to come with clearly visible patterns or conclusions (Lehrer, Giles, & Schauble, 2002). Visiting a plantation through immersive digital representations as seen in VR allows for an extension of concepts and data previously presented through less interactive forms like lectures and textbooks. This new context allows students to experience and engage with a new representation of data that they have seen previously. The context of VR can enable new ways to manipulate, analyze, and visualize data, all of which are valuable computational thinking practices. As a student noted, the use of VR demonstrated “how connected you can really be with the real world.” Virtual reality allows students to experience data in a completely different way and deepen their engagement with it. This experience develops computational thinking practices related to interacting with, communicating about, and drawing meaning from data, all mediated by a technological platform.

A second computational thinking in mathematics and science taxonomy category this lesson links to is systems thinking, although in this case, we are exploring a social system rather than a scientific one. Asking students to explore the plantation was part of a larger instructional goal of helping students understand Southern society during that period and the role of slavery in the culture. Through

immersive experiences such as these, students could investigate different dimensions of the culture and explore the relationship between slaves, slave-owners, industry, and the economic factors that contributed to the Civil War, viewing the various levels of the system and gathering information that can be used to discuss the relationships within the system and the system as a whole.

4.3. Computational Thinking in Spanish Class

In Spanish class, students used VR to visit Spanish speaking countries throughout the year. Classes visited Hispanic neighborhoods in the United States, Latin American countries, and Spain. While completing a Spanish culture unit that included study of the regions of Spain, 7th grade students had two opportunities to use VR. First, the teacher used VR in a series of stations that introduced some of the best-known landmarks of the country. The class concluded with a full class discussion of the similarities and differences between the sites themselves and between the sites and the United States. Later, students created a presentation about one region of the country and selected one major landmark that represented the region to share via VR. After working on the project for a week, students shared their presentations with the class in a gallery walk fashion, leaving their presentation and VR destination for other students to discover as they walked around the classroom.

The opportunities for computational thinking within this lesson are very similar to those experienced in the social studies lesson. Students are able to utilize data practices by making connections between what they had learned previously, viewing it represented in a new manner, and potentially treating the images that they are viewing as data themselves for data analysis. Additionally, students are experiencing a social and cultural system, employing the systems thinking practices to understand relationships and think in levels. The presentation of these computational thinking practices in Spanish class serves as yet another context for learners to develop these skills.

4.4. Student Impressions of Virtual Reality

Students recognized VR as a productive learning tool within their classrooms. Using a five-point Likert Scale, 94% of students agreed or strongly agreed that VR helped them learn, with a mean score of 4.33 out of 5 (*SD* .64). According to one student, “it made my understanding of the place better because we didn’t have to hear about it we could see the place ourselves.” Alongside this perceived learning utility, students’ reported increased feelings of engagement and connections with course material through the VR environment. Ninety-two percent of students agreed or strongly agreed that they felt more engaged in classes because of VR (Mean 4.48 out of 5, *SD* .64) and 97% agreed or strongly agreed that they made connections between what they were seeing the viewer and what they learned in class (Mean 4.50 out of 5, *SD* .56). When asked how VR helped them learn, students highlighted the engagement that they felt using the technology. Students stated that they were “really engaged because it was like we were really there and it was very interesting” and “[virtual reality] helped [me] become more focused and involved in a lesson.” Students also showed enjoyment from using VR calling the experiences “truly awesome” and “really enjoyable” while

requesting the use of VR more often and in more classes. Given this reaction to the use of VR in the classroom and the opportunities for it to serve as a context for the development and employment of computational thinking practices, the partnering of the two provides a productive means by which to introduce students to computational thinking across contexts.

5. DISCUSSION

Virtual reality has the potential to be a powerful tool for bringing computational thinking into the classroom, both within and beyond STEM subjects. The unique views that it allows, along with the increased engagement and learning that the environment brings, create a medium with great potential for computational thinking. The lessons in this article demonstrate a few of the ways in which VR can be used to situate computational thinking across the curriculum. This work aims to help start the discussion regarding VR as a conduit for computational thinking. By viewing social systems through a systems thinking lens and using three dimensional models as a context for learners to explore ideas and engage with data, students have the ability to develop computational thinking practices, no matter the subject they are studying.

Virtual reality demonstrates the breadth of computational thinking and shows how it can be experienced beyond computer science classrooms. Further, the flexibility of VR to fit across the curriculum provides a mechanism to show how computational thinking can serve as a set of cross-cutting practices without disciplinary constraints. Given the possibilities for incorporating computational thinking in the humanities, more work needs to be done on defining what computational thinking looks like in these contexts and how it differs from the presentation of computational thinking practices in STEM subjects.

Although VR is becoming increasingly accessible, there are still limitations to using it within the classroom. The cost of the equipment has decreased, but it remains out of reach for many classrooms. Additionally, the availability of VR programs limits classroom activities and most teachers do not have the skills to design their own VR programs. With the development of VR mainly driven by consumer electronics and technology companies, teachers need to be aware of the economic motivation of VR platforms and consumer opinions of such platforms. Teachers should also be aware of the effect that novelty can have on students and use of new technologies and ensure that the substance of a lesson is not missed due to being distracted by the novelty of a learning tool. Lastly, especially with head mounted devices, physical discomfort can be experienced by users. Some students require adjustments to use the technology comfortably.

6. CONCLUSION

Computational thinking is an important skill for all students to develop. With the ever-growing number of fields that rely on computation and an increasingly technical world, students must be prepared through diverse exposure to computational thinking tasks. Virtual reality offers one way to enable such exposure. Students are drawn to the technology and can benefit from the engagement, learning,

and connections that it offers. With VR and computational thinking working together, students will have the opportunity to experience computational thinking in not only STEM fields, but also in the humanities. There is a great need for computational thinking in modern society and VR is a tool that will help develop this essential mindset.

7. ACKNOWLEDGEMENTS

The authors would like to thank the McCarthy Dressman Education Foundation for the Academic Enrichment Grant which made these lessons possible and the students for inspiring us to develop new learning opportunities.

8. REFERNECES

- Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning and Leading with Technology*, 20–23.
- Greenwald, S. W., Kulik, A., Kunert, A., Beck, S., Fröhlich, B., Cobb, S., ... Maes, P. (2017). Technology and applications for collaborative learning in virtual reality. In *Making a Difference: Prioritizing Equity and Access in CSCL, 12th International Conference on Computer Supported Col- laborative Learning (CSCL)*, 719–726.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38–43.
- Henderson, P. B., Cortina, T. J., & Wing, J. M. (2007). Computational thinking. *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '07*, (February 2016), 195.
- Huang, H. M., Rauch, U., & Liaw, S. S. (2010). Investigating learners' attitudes toward virtual reality learning environments: Based on a constructivist approach. *Computers and Education*, 55(3), 1171–1182.
- Lehrer, R., Giles, N., & Schauble, L. (2002). Data Modeling. In R. Lehrer & L. Schauble (Eds.), *Investigating real data in the classroom: expanding children's understanding of mathematics and science* (pp. 1–26). New York: Teachers College Press.
- Limniou, M., Roberts, D., & Papadopoulos, N. (2008). Full immersive virtual environment CAVE in chemistry education. *Computers & Education*, 51(2), 584–593.
- Merchant, Z., Goetz, E. T., Cifuentes, L., Keeney-Kennicutt, W., & Davis, T. J. (2014). Effectiveness of virtual reality-based instruction on students' learning outcomes in K-12 and higher education: A meta-analysis. *Computers and Education*, 70, 29–40.
- NGSS Lead States. (2013). Next Generation Science Standards: For States, By States. Retrieved from <http://www.nextgenscience.org>
- Orton, K., Weintrop, D., Beheshti, E., Horn, M., Jona, K., & Wilensky, U. (2016). Bringing computational thinking into high school mathematics and science classrooms. In *Transforming Learning, Empowering Learners: The International Conference of the Learning Sciences (ICLS)* (pp. 705–712).
- Pan, Z., Cheok, A. D., Yang, H., Zhu, J., & Shi, J. (2006). Virtual reality and mixed reality for virtual learning environments. *Computers and Graphics (Pergamon)*, 30(1), 20–28.
- Papert, S. (1980). *Mindstorm: Children, Computers, and Powerful Ideas* (1st ed.). New York: Basic Books, Inc., Publishers. <https://doi.org/10.1017/CBO9781107415324.004>
- Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012). Infusing computational thinking into the middle- and high-school curriculum. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education - ITiCSE '12*.
- 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., & Resnick, M. (1999). Thinking in levels : A dynamic systems approach to making sense of the world. *Journal of Science Education and Technology*, 8(1), 3–19.
- 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*, 366, 3717–3725.
- Winn, W., Windschitl, M., Fruland, R., & Lee, Y. (2002). When does immersion in a virtual environment help students construct understanding. In *Proceedings of the International Conference of the Learning Sciences, ICLS* (pp. 497–503).

A Development of a SW-STEAM Education Program using the Flipped Learning

Hae-nam SONG*, Sun-gwan HAN

Department of STEAM Education, GyeongIn National University of Education, South Korea
goska9997@naver.com, han@gin.ac.kr

ABSTRACT

In this study, we developed SW STEAM education program that can be provided by using flipped learning. To do this, I applied flipped learning to a class of 4th grade elementary school in Gyeonggi Province, progress the online course site related to SW education at the same time. After that, we applied a program that combines the existing textbook theme with scratch. This will help students improve their Computational Thinking and motivation. In the future, it is expected that SW STEAM education will be activated in elementary education field by using flipped learning.

KEYWORDS

Computational Thinking, Flipped Learning, SW Education, STEAM, STEAM Education.

1. INTRODUCTION

In the 21st century society, more people need to be able to think convergently based on knowledge rather than those who memorize and understand simple knowledge. This change in social paradigm is too fast to catch up with the existing knowledge-based lecture education. Elvin Toffler diagnosed this as a "educational delay" and called for a change to a creative and STEAM education.

Flipped learning is emerging as a teaching method that can effectively cope with this educational crisis and raise talented people capable of fused thinking. Teachers can create and distribute video about the core contents of the curriculum to be delivered so that students can study at home. In the classroom, learning is organized by learner-centered activities. It is not a passive class that receives knowledge, but a class that worries for oneself to solve problems. In this process, students can naturally develop fusion thinking skills.

Although many educators agree on the effectiveness of flipped learning, it is difficult to apply it in schools. The reason why it is difficult to apply flipped learning to the school is 'difficulty in using the device'. Secondly, 'production burden of pre-learning video' may be the reason.

On the other hand, as the 4th industrial revolution is accelerating, the need for software education is increasing. The Ministry of Education and the Ministry of the Future revised the curriculum so that it emphasizes the software education. Now, study to software is essential. No one can deny that SW education is necessary for future social talent training.

As the need for STEAM education and SW education is increasing, practical learning programs are needed in elementary schools. Elementary SW STEAM education should enable the learner to be able to participate actively while maintaining interest and concentration. It also needs to be presented by the easy way to access, with topics related to curriculum.

Therefore, this study utilize MOOC - based flipped learning which uses pre - developed teaching - learning site lectures, learn the pre-production video by using the extra-curricular time. By using this method, we intend to conduct SW STEAM lesson connect with subjects(Korean, Mathematics, Science). During class, students can create a scratch project that fits their subject matter.

2. BACKGROUND

2.1. Concept of Flipped-learning

'Flipped' is a name given in the sense of reversing lectures and homework. In other words, it is a class that overturns the traditional way of teaching in terms of studying videos at home and conducting classroom activities based on what they have learned.

The concept of flipped learning can be defined as follows. Flipped learning is a learner-centered approach to self-study of core knowledge to learn in the home or school, making the network, communicating with friends, conducting project activities, group discussions, and quizzes.

2.2. Concept of SW Education

Software education is expanded from ICT education which teaches the functions of information devices such as word processor. Software education provided computer theory, and ability to think through procedural thinking and solve problems by using software. With the 4th Industrial Revolution, production methods in all sectors of the industry are changing, and software is at the center. The ability to deal with software, and the ability to solve various real-life problems using software is becoming important. Software education is rapidly spreading in educational fields around the world. Korea is also taking a step closer to change by strengthening SW education in the 2015 revision curriculum.

SW education aims to develop creative talents who have the ability to solve problems by collecting data and analyzing information on the basis of thinking rather than nurturing students as programmers who are simply coding. Therefore, it is not a one-time coding education, but real-life problems are solved through algorithms and programming in a practical subject.

2.3. Flipped-learning of the SW Education.

Students can learn basic functions necessary for SW education beforehand at home in the pre-learning stage through video. Beyond simple video viewing, teachers should provide opportunities for students to share their thoughts in a quiz or mind map.

In SW education, learning about EPL (Educational Programming Language) like scratch basic functions is essential, but there are limited class hours and difficulty for one teacher to proceed. Therefore, the difficulty of applying SW education can be solved through flipped learning.

3. RESEARCH METHOD

3.1. Research Procedure

In this study, we have found subjects and theme that can apply SW STEAM education. Among the 4th grade subjects, they were selected as 'talking' in Korean, 'polygon' in mathematics, and 'change of state of water' in science. After the selection of the topic it was subdivided and refined to implement the learning objectives to scratch without modifying the existing curriculum content. In order to increase the applicability in the field of elementary education in the future, we selected topics that can be connected with SW education among existing contents of textbook rather than modification of curriculum contents.

In addition, the selected online teaching and learning site allows students to study at home the function of the scratch program by flipped-learning. At the school, based on the functions learned at home, we conducted mind map and discussion learning

3.2 Application

A total of 19 students were selected, including 8 boys and 11 girls, in the 4th grade of K elementary school in Gyeonggi Province. There is one student with an intellectual disability, and that student is excluded from the application group because that student takes special classes in Korean language and mathematics. There are no students who have been exposed to scratches in advance, and there are no after school computer attendants, and there is no SW education experience.

4. DESIGN OF SW STEAM PROGRAM

4.1. Learning Model of Flipped Learning

The flipped Learning model to be applied in this study is based on the 'core activity process of flipped learning based instruction model', from the perspective of using scratch, a tool for SW education, was modified according to research characteristics, with reference to 'Development of flipped learning instruction model based on smart education'.

Table 1. Learning model of flipped Learning

Process	Core Activities	Rule
Before The class (outside of classroom)	<ul style="list-style-type: none"> ▪Prior Learning (watching the video) ▪Confirm contents of subject 	<ul style="list-style-type: none"> ▫Presenting pre-learning tasks by Mind-map, Scratch quiz, post-it quiz ▫Upload to classroom site after creating the reviewing project
In class	<ul style="list-style-type: none"> ▪Readiness check 	<ul style="list-style-type: none"> ▫Prior knowledge check ▫Identify the individual level with pre-learning assignments and reviewing project analysis

	<ul style="list-style-type: none"> ▪Objectives Recognition -Provide Motivational data -Curiosity inducing 	<ul style="list-style-type: none"> ▫Solvable Problems by Cooperation among students
	<ul style="list-style-type: none"> ▪Understanding the Knowledge And providing Feedback -Explore individual information activity -Individual Knowledge Organization Activities 	<ul style="list-style-type: none"> ▫ Confirm textbook And Formalization of knowledge. ▫ Individual structured data
	<ul style="list-style-type: none"> ▪ Seeking application examples for knowledge production and reconstruction -Team cooperative learning -Professor and peer evaluation 	<ul style="list-style-type: none"> ▫ Utilize project subject topics ▫ Making a plan specifically for what learners should do to solve problems ▫ Consider cognitive and social interaction
	<ul style="list-style-type: none"> ▪ Learning outcomes cleanup -Sharing and presenting 	<ul style="list-style-type: none"> ▫ Announced creative activity outcomes ▫ Teacher's facilitator activity
	<ul style="list-style-type: none"> ▪ Learning theorem and Reflection 	<ul style="list-style-type: none"> ▫Self-reflection with Writing reflective journals
After class (outside of classroom)	<ul style="list-style-type: none"> ▪ Providing the Deepening learning and Supplementary learning -Sharing activity, Interactive activity 	<ul style="list-style-type: none"> ▫Providing the Deepening activities based on students' creativity through experiential knowledge

4.2. Application of Flipped Learning

In the conventional flipped learning class, the teacher has to prepare and provides the class related video. However, in this study, the video is already produced and distributed online, so that the burden on the user can be reduced. In this study, Junior SW site (koreasw.org) was utilized. All of the lectures on this site were produced by teachers and agreed with the curriculum and were suitable for flipped learning of learners within 8 to 10 minutes. Based on the contents that students have heard from the online-learning site, teacher suggested prior learning so that students can share their thoughts with each other.

Table 2. Application plan for flipped Learning

Class time	Subject	Flipped Learning Activity
1	Introduction	◇ Introduction to Online Learning Site ◇ Join Scratch site
2	‘Let’s be the main character.’	◇ Watching a video (Pre-learning assignment) ◇ Create mind map
3	‘Let’s move the character.’	◇ Watching a video (Pre-learning assignment) ◇ Write new points on post-it
4	‘Let’s Decorate aquariums’	◇ Watching a video (Pre-learning assignment) ◇ Unravel the quiz
5	‘Dance Party’	◇ Watching a video (Pre-learning assignment) ◇ Summarize the contents of a lecture
6	‘Fireworks’	◇ Watching a video (Pre-learning assignment) ◇ Write Lecture Notes

In this study, it is aimed to reconstruct with SW STEAM Education using MOOC based flipped learning, by presenting real-life problems, pursuing connectivity with other subjects, stimulate students' interest and naturally develop communication skills and problem-solving skills.

4.3. SW STEAM Education Project Production

1) LANGUAGE ART

Table 3. SW KOREAN STEAM education Contents

Making polite conversations Collection	Class time 1	Presenting the situation	Create a polite conversation scratch that Anyone can easily see
	Class time 2~3	Creative Design	<ul style="list-style-type: none"> - Think about Situation of conversation with adults (A) - Think about the manners you need to talk to adults (A) - What blocks do you need to make the scratch that seems to be talk? (T) - Using scratch, Making polite conversation Collections (E)

Class time 4	Emotional experience	<ul style="list-style-type: none"> - Uploading the project to classroom site - Creating a ‘polite conversation Collections’ by collecting all of your projects (T)(E)(A) - writing a comment Watching each other's projects
	(Experience of success)	

2) MATHEMATICS

Table 4. SW Mathematics STEAM education Contents

Square maker	Class time 1	Presenting the situation	Create a tool that accurately draws a Square
	Class time 2~3	Creative Design	<ul style="list-style-type: none"> - Think of the square features (Rhombus, Parallelogram, Rectangle)(M) - What blocks do you need to draw a square accurately? (T)
	Class time 4	Emotional experience	<ul style="list-style-type: none"> - Draw a rectangle using scratch (M)(E) - Uploading the project to classroom site - writing a comment Watching each other's projects

3) SCIENCE

Table 5. SW Science STEAM education Contents

Creating a ‘Moon survival game’	Class time 1	Presenting the situation	Let's play with your friends by creating a game that you need to survive on the moon
	Class time 2~3	Creative Design	<ul style="list-style-type: none"> - Think about the difference between Moon and Earth (S) - Why can not a creature live on the moon? (S) - Designing the game situation(A)(E) - Exploring the block (T)
	Class time 4	Emotional experience	<ul style="list-style-type: none"> - Making game ‘Moon survival game’(S)(T)(E) - Game with friends

	of success)	- Share each other's rules and games
--	-------------	--------------------------------------

of the program on learning motivation by tests divided into pre and post motivation and analysis the satisfaction survey. It can be seen that SW STEAM class using flip learning gives a very high learning motivation than traditional lecture class.

4) Implementation of SW-STEAM Class
 Table 6. Picture of class

<p>SW KOREAN STEAM</p>	
<p>SW Mathematics STEAM</p>	
<p>SW Science STEAM</p>	

Table 7. Result of motivation test

Analysis	Correspondence	Average	SD	t	p
Attention	Pre	3.4722	.74206	-2.536	.021
	Post	3.9444	.78850		
Relevance	Pre	3.0889	.49573	-3.194	.005
	Post	3.5444	.52156		
Confidence	Pre	3.4889	.91065	-2.279	.036
	Post	3.9667	.81818		

Therefore, when SW STEAM class using flip learning is applied to students, it can positively affect students' motivation for learning. Especially, it showed improvement in attention, relevance, and confidence among the sub - areas of learning motivation.

6. REFERENCES

Alvin, T., & Toffler, H. (2006). Revolutionary wealth. *Alfred A Knopf*.

Bergmann, J. & Sams, A. (2012). Flipped Your Classroom: *Reach Every Student in Every Class Every Day*. International Society for Technology in Education.

Korea online SW education. (2016). Scratch Basic. Retrieved October 17, 2017, from <http://koreasw.org>.

Lee, M. K., Sung, M. K., Jung, J. Y., Kim, S. M., Kim, J. H., Ahn, H. H., Park, H. K., Patrick, T. T., Byeon, S. C., Bae, D. Y., Lee, K. H., Kim, S. C., Cha, J. H., Kim, E. J., Kim, K. Y., Lee, H. J., Kim, K. Y., & Kim, C. S. (2016). *Understanding and Practice of Flipped Learning*, city: Kyoyookbook.

Lim, J. H., & Kim, S. H., (2013). Effects of individual learning and collaborative learning on academic achievement, self-directed learning skills and social efficacy in smart learning, *Journal of Korean Association for Educational Information and Media*, 19(1), 1-24.

Missildine, K., Fountain, R., Summers, L., & Gosselin, K. (2013). *Flipping the classroom to improve student performance and satisfaction*. *Journal of Nursing Education*, 52(10), 597-599.

MOE. (2015). *Curriculum guideline for Practical Arts (Technology / Home Economics) / Information Science Curriculum*. Seoul: Ministry of Education.

Park, T. J., Cha, H. J., & Lee, G. Y. (2015). An Exploratory Study on Learning Analysis for Promoting Self - Regulated Learning in MOOCs Learning Environment. In *Proceedings of 2015 Conference on The Korean Society Educational Technology* (pp. 504-517).

5. DISCUSSION

The developed SW STEAM education applied to 4th grade students. The purpose of this study is to investigate the effect

Development of BIC-Science Module: An Interdisciplinary Approach of Computer Science and Primary Science Education

Tracy MENSAN, Kamisah OSMAN*
National University of Malaysia
tress1907@yahoo.com, kamisah@ukm.edu.my

ABSTRACT

Computational Thinking (CT) is being considered as a critical skill for students in the 21st century as it is increasingly valuable in education and workplace settings with the economy grows more dependent on digital literacy. Given the importance of CT, Malaysia has been integrating CT into Malaysian syllabus since January 2017. However, integration of CT into the Science curriculum is still a challenge. This study therefore aimed to develop an interdisciplinary module namely Brain-based learning, Inquiry-based approach and Computational thinking (BIC)-Science Module. In this paper, we first present the needs of the module to Malaysia's education and then presenting the approaches of BIC through the conceptual framework. We then propose activities that can jointly foster the development of computational thinking and elaborate on the instructional model to develop the module. Finally, we discuss the benefits of our module for future research.

KEYWORDS

Interdisciplinary, Brain-based learning, Inquiry, Computational thinking, Primary Science.

1. INTRODUCTION

In August 11th of 2016, the Prime Minister of Malaysia has announced that computational thinking and computer science will be added to the curriculum of primary and secondary schools in Malaysia (Abas 2016), which aimed to provide Malaysian students with the CT to be globally competitive. The Prime Minister highlighted every student from Primary One to Form Five should be taught CT and coding languages to give them a good foundation in preparing them for future digital economy jobs.

The implementation of CT has been rolled out as part of the new Standard Based Curriculum for Primary (KSSR) and Standard Based Curriculum for Secondary (KSSM) which has been started in January 2017 that will benefit up to 1.2 million students across 10,173 schools nationwide (Abas, 2016). The integration of CT, problem-solving and technology for the primary school curriculum will be across all of their subjects. Meanwhile, the integration of CT for secondary school curriculum is through their elective subject. These initiatives are spearheaded by Ministry of Education (MOE) and supported by Malaysia Digital Economy Corporation (MDEC) and aimed to participate 1.3 million students participating in co-curricular activities and digital production hubs with 260,000 students groomed for future digital economy jobs, e.g. data scientists and game developer (Ng, 2016).

A national study, S&T Human Capital: A Strategic Planning Towards 2020 in Academy of Sciences Malaysia (2015) confirmed that the country will need one million S&T workers by 2020, of which 500,000 will require at least a diploma or university degrees. At the same time, it is projected that a ratio of 70: 10,000 research personnel to workforce would be needed. Hence, the underlying statement indicates that Malaysia still does not have enough talent.

The implementation of the first National Science and Technology Enrolment Policy of 60:40 since 1970, which guaranteed that 60 percent of students would be enrolled in science with the remaining 40 percent in arts is still unachieved with the ration stood at 21:79 in 2015. Regarding the latest statistics on mean score in Program for International Student Assessment (PISA) and the Trends in International Mathematics and Science Study (TIMSS) 2012 which assess a variety of cognitive skills such as application and reasoning, Malaysia's science and mathematics achievement still ranked below the average mean score. Therefore, Malaysia education system aspires to be in the top third countries of international assessments such as TIMMS and PISA in 15 years (Ministry of Education, 2015).

In order to achieve the national goals, this paper proposed the interdisciplinary module that supports the development of students' scientific expertise for the design of coherent curriculum in which computational thinking are not taught as separate topic but are interwoven with learning in the science domains. Bringing computational tools and practices into science classrooms gives learners a more realistic view of what science fields are and better prepare students for STEM careers (Augustine, 2005; Osman, 2013). These practices are also central to the development of expertise in scientific and mathematical disciplines (Basu et al., 2012). In establishing this framework, we first propose the following three components:

- Relationship between BIC and Science Learning:* In section 2.1, we explicitly identify the synergies between BIC and science learning;
- Fostering CT with BIC-Science Module:* In section 2.2, we provide examples for the integration of CT in the selected topic that are amenable to our technology, but at the same time illustrate the generality of our approach;
- Instructional design of BIC-Science Module:* In section 2.3, we elaborate the Morison, Ross and Kemp (MRK) instructional model for developing the module.

2. DEVELOPMENT OF MODULE

2.1. Relationship between BIC and Science Learning

In order to comprehend the continuous development in the discipline of science, students should be aware of the basic science terms and they should gain the science skills throughout their schooling process (Fogarty, 2002) which can be achieved through interdisciplinary approach presented in this module. Interdisciplinary can be defined as a knowledge view and curriculum approach that consciously applies methodology and language from more than one discipline to examine a central theme, issue, problem, topic or experience (Jacobs, 1989). *Figure 1* shows the conceptual framework that shows the key concepts in developing the module.

Computer science element focused in this module is the use of computational thinking as the skills to solve problem systematically in the lesson. Meanwhile, the science learning will be focused on the curricular contexts in the topic of “Matter” which is difficult and important curricular topic at Year 5 level. Research reports some of the ideas students have about the particulate nature of matter as misconceptions, preconceptions, naive conceptions, or alternative conceptions (De Vos & Verdonk,1996).

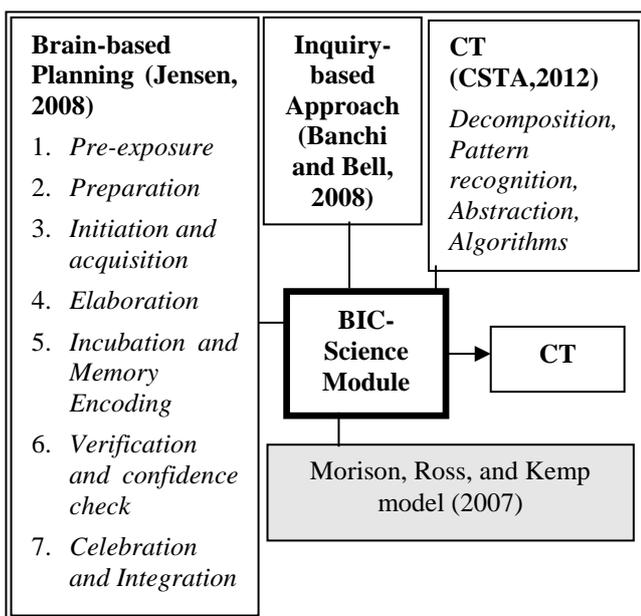


Figure 1. Conceptual framework.

BIC model is adapted from the model proposed by Cheah (2016) as an effective pedagogy that should consist of:

- a. structure: Brain-based learning;
- b. approach: Inquiry-based approach; and
- c. skill: Computational thinking.

The structure is the brain-based learning that recognized the need for constructing knowledge, prior conceptions into new knowledge through questioning and readjusting knowledge to fit with real-life experiences (Gardner,1991 in Mangan, 2007). This can be achieved through the Seven Stages of Brain-based Planning that can be applied in science classroom to “access the vast potential of the human brain and, in very real sense, improve education.” (Caine & Caine,

1991). In BIC-Science Module, every science lesson is structured into seven stages according to Jensen (2008) namely:

- i. *Pre-exposure* is the stage which provides the brain with an overview of the new learning before really digging in. Pre-exposure helps the brain develop better conceptual maps. Example: Students can use their prior knowledge about different types of materials around them to help them to understand the nature of different states of matter that can exist as solid, liquid and gas.
- ii. *Preparation* is the stage at which curiosity or excitement is created. It is similar to the ‘anticipatory set’ but goes further in preparing the students. Example: Students are instructed to put their hand into three closed black boxes which contain different types of matter separately. Each box may contain ice which represents solid, water which represents liquid and smoke which represents gases.
- iii. *Initiation and acquisition* is the stage which provides the immersion. Students are flooded with an initial virtual overload of ideas, details, complexity and meanings. The students are allowed to be temporarily overwhelmed. This will be followed by anticipation, curiosity and determination to discover meaning for oneself. It builds on what the learners already know and understand and helps them assimilate and integrate new information. Over time, the students are able to sort out the knowledge. Example: Students are allowed to do experiment to describe that water can change its state through several processes.
- iv. *Elaboration* is the stage for processing which requires genuine thinking on the part of the learners. This is the stage to make intellectual sense of the learning. Example: Students discussed openly the algorithm they experienced in changing the states of matter in water into solid or gas. Teachers and other students may ask questions to improve the algorithm.
- v. *Incubation and memory encoding* is the stage for the importance of downtime and review time is emphasized. Example: Students write the key points about the "changes in states of matter" in the form of thinking map in their journal.
- vi. *Verification and confidence check* is for the students to confirm their learning. Learning is best remembered when students possess a model or a metaphor regarding the new concepts or materials. Example: Students answered short quiz regarding the subtopic learned.
- vii. *Celebration and integration* is the stage which engage emotions. This stage instills the all-important love of learning. Example: Stickers are given to students who perform well and actively throughout the lesson. Top presentations are selected to be presented during Science Week.

While brain-based learning develops deep learning of science phenomenon as a process, inquiry-based approach offers the ability to do the scientific processes and the knowledge about the processes through student-centered exploration. Students are encouraged to raise questions and think critically throughout the exploration of lesson

activities which also will provide opportunity for students to learn by doing. The national performances in TIMMS and PISA proved that our students are still lacking in inquiry skills. Therefore, the design of activities in this module will be developed from the basic which is structured inquiry to guided inquiry or open inquiry (NRC,2000). Banchi and Bell (2008) differentiate the four levels of inquiry (confirmation inquiry, structured inquiry, guided inquiry and open inquiry) based on the amount of information and guidance the teacher provides the students. The information and guidance provided in the module will be minimized as the inquiry level shifts from structured inquiry to open inquiry.

CT will equip the module with relevant skills according to the science activity. The term “computational thinking” in education was first used in child education by Papert (1980) with reference to Logo, a computer language designed for children who believes that certain uses of very powerful computational technology and computational ideas can provide children with new possibilities for learning, thinking, and growing emotionally. According to Curzon *et al.* (2009), computational thinking is the 21st century skills. This is an idea explored by Jeannette Wing from Carnegie Mellon University:

Computational thinking is a way of solving problems, designing systems, and understanding human behavior that draws on concepts fundamental to computer science.
(Wing, 2006).

In this study, CT skills are needed to prepare a lesson for the learner in a systematic manner. Four concepts of the CT skills (CSTA, 2012) defined in *Table 1* will be utilized.

Table 1. Four concepts of CT

Concept	Definitions (Google, 2015)
Decomposition	Breaking down data, processes, or problems into smaller, manageable parts
Pattern recognition	Observing patterns, trends, and regularities in data
Abstraction	Identifying the general principles that generate these patterns
Algorithms	Developing the step-by-step instructions for solving problem

2.2. Fostering CT with BIC-Science Module

The long-term goal of this study is to support the development of CT throughout the Primary Science curriculum. BIC-Science module is designed to promote a specific set of CT skills for the topic. *Table 2* below shows examples that incorporates CT skills in the module.

Table 2. CT Concepts Explored with BIC-Science Module

Concept	Examples
Decomposition	Students decomposed the changes in states of matter which occur during the phenomena of rain.

Pattern recognition	Students classify the materials/objects in the classroom into solid, liquid and gas.
Abstraction	Students use abstraction to explain the changes in states of matter during the heating of naphthalene ball.
Algorithms	Students explore logical organization and sequencing when animate the movement of particles in solid, liquid and gas using visual programming application; Scratch.

2.3. Instructional design of BIC-Science Module

Morison, Ross and Kemp (MRK) model provide flexibility in manifesting the cyclical process of instructional design (Morrison *et al.*, 2007) in the development of this module. This circularity is achieved by viewing the nine core elements of the model as interdependent rather than singular and independent. This allows instructional designers a significant degree of flexibility because they are able to begin the design process with any of the nine components, rather than being constrained to work in a linear fashion (Akbulut, 2007). Every aspect of the module design and learning process is taken into consideration. This model focuses on these nine core elements which will be applied in this module:

- identifying instructional design problems and specifying relevant goals,
- examining learner characteristics,
- identifying subject content and analyzing task components that are related to instructional goals,
- stating instructional objectives for the learners,
- sequencing content within each unit to sustain logical learning,
- designing instructional strategies for each learner to master the objectives,
- planning instructional delivery,
- developing evaluation instruments, and
- selecting resources to support learning activities.

3. CONCLUSION

The development of BIC-Science module will be the foundation for a longer-term learning progression to integrate computational thinking into the science curriculum. The design of science lesson activities using brain-based learning, inquiry-based approach and computational thinking will be able to provide a student-centered, systematic and meaningful learning environment. With computational thinking’s growing importance in preparing relevant talent in digital age, this paper is a call to action for more research to integrate computational thinking in other disciplines and in the different level of education.

4. REFERENCES

- Abas, A. (2016, August 11). Computational thinking skills to be introduced in school curriculum next year. *New Straits Times*.
<https://www.nst.com.my/news/2016/08/164732/computational-thinking-skills-be-introduced-school-curriculum-next-year> [2 February 2018].

- Academy of Sciences Malaysia. (2015). *ASM Science Outlook*. Kuala Lumpur: Perpustakaan Negara Malaysia.
- Akbulut, Y. (2007). Implications of two well-known models for instructional designers in distance education: Dick-Carey versus Morrison-Ross-Kemp. *Turkish Online Journal of Distance Education*, 8(2).
- Augustine, N. R. (2005). *Rising above the gathering storm: Energizing and employing America for a brighter economic future*. Washington D.C.: National Academies Press.
- Banchi, H., & Bell, R. (2008). The many levels of inquiry. *Science and Children*. 46(2), 26-29.
- Basu, S., Kinnebrew, J., Dickes, A., Farris, A. V., Sengupta, P., Winger, J., & Biswas, G. (2012). *A Science Learning Environment using a Computational Thinking Approach*. Paper presented at the 20th International Conference on Computers in Education, Singapore.
- Caine, G., & Caine R. (1991). *Making Connections (Teaching and The Human Brain)*. USA: Banta Company.
- Caine, G., & Caine, R. (2006). *Making connections: Teaching & Human Brain (3rd ed.)*. Thousand Oaks, CA: Corwin Press.
- Cheah, H.M. (2016). Enhancing Creative Teaching using Computational Thinking. *International Conference on Teaching and Learning 2016*, hlm. 26–42. Faculty of Education, University of Malaya, Kuala Lumpur.
- Computer Science Teachers Association. (2012). Computational Thinking. <http://csta.acm.org/Curriculum/sub/CompThinking.html>.
- Curzon, P., Black, J., Meagher, L. R., & McOwan, P. (2009). cs4fn. org: Enthusing students about Computer Science, *Proceedings of Informatics Education Europe IV*, 73-80.
- De Vos, W., & Verdonk, A.H. (1996). The Particulate Nature of Matter in Science Education and in Science. *Journal of Research in Science Teaching*, 33(6), pp657-664.
- Google. (2015). Computational Thinking for Educators. Retrieved from <https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1>.
- Hodson, D. (1993). Re-thinking old ways: Towards a more critical approach to practical work in school science. *Studies in Science Education*, 22, 85–142.
- Jacobs, H. H. (1989). Design options for an integrated curriculum. *Interdisciplinary curriculum: Design and implementation (pp.13-15)*.
- Jensen, E. (2000). *Brain-based learning*. Thousand Oaks, CA: Corwin Press.
- Mangan, M. A. (2007). *Brain-compatible science (2nd ed.)*. Thousand Oaks, CA: Corwin Press.
- Ministry of Education. (2015). *Malaysia Education Blueprint 2015-2025 (Higher Education)*. Ministry of Education Malaysia.
- Morrison, G.R., Ross, S.M. & Kemp, J.E. (2007). *Designing effective instruction. 5th Edition*. New York: John Wiley & Sons.
- Ng, W.P. (2016). *Creating Connected, Empowered Communities Transforming The Digital Economy To Equip The Future Workforce With Vital Skillsets Dato' Ng Wan Peng Chief Operating Officer (COO) Malaysia Digital Economy Corporation (MDEC)*. Retrieved from https://asia.bettshow.com/sites/asia.bettshow.com/files/Plenary_0930-0945_Dato Ng Wan Peng.pdf
- OECD. (2012). *Program for International Student Assessment results (PISA) from PISA 2012: Country note- United States*. Retrieved from *PISA 2012 Results in Focus- What is 15 year-olds know and what they can do with what they know*, OECD, 2014, pp.1-44.
- Osman, K., Hiong, L. C. & Vebrianto, R. (2013). 21st Century Biology: An Interdisciplinary Approach of Biology, Technology, Engineering and Mathematics Education. *Procedia - Social and Behavioral Sciences* 102(Ifee 2012): 188–194.
- Palaniappan, A.K. (2009). Creative Teaching and Its assessment. 12th UNESCO-APIED International Conference. Bangkok, Thailand
- Papert, S. A. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*, Basic Books.
- Saleh, S. (2012). The effectiveness of Brain-Based Teaching Approach in dealing with the problems of students' conceptual understanding and learning motivation towards physics. *Educational Studies* 38(1): 19–29.
- Wing, J. M. (2006). Communications of the ACM. *Communications Of The ACM. March* 49(3). Retrieved from <https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf>

Thinking in Parts and Wholes: Part-Whole-Thinking as an Essential Computational Thinking Skill in Computer Science Education

Nils PANCRATZ*, Ira DIETHELM
Department of Computing Science
University of Oldenburg, Germany

nils.pancratz@uni-oldenburg.de, ira.diethelm@uni-oldenburg.de

ABSTRACT

Thinking in parts and wholes is a basic principle in Computer Science. Breaking down complex structures, objects, and systems into its componential parts and figuring out how they make the whole what it is, is an essential thinking skill that forms understandings on the functionalities on how these things work. But this skill, which is defined and presented as *Part-Whole-Thinking* in this paper, is also applicable to grasp non-physical ideas such as concepts, processes, and definitions. Either way, Part-Whole-Thinking is an often subconsciously happening cognitive process that forms knowledge representations. The contribution at hand aims at working out in which way Part-Whole-Thinking belongs and relates to *Computational Thinking*. By reviewing literature on suitable definitions of the involved terms it is shown that Part-Whole-Thinking plays a huge role in Computational Thinking processes. Afterwards, it is argued that a more vigorous inclusion of this essential thinking skill in *Computer Science Education* improves the overall understanding of Information Technology.

KEYWORDS

Part-Whole-Thinking, Computational Thinking, Cognitive Organization, Computer Science Education

1. INTRODUCTION

The term *Computational Thinking* (CT) is increasingly being used in discussions about *Life Long Learning* (LLL) recently. However, since Wing was the first to use the term in educational contexts in 2006, many authors defined this term differently in their work. We argue that CT is not only thinking *like* computers/computer scientists, IT-devices, etc.; more importantly it is a skill that enables thinking and reasoning *about* the way these devices work. Since it is a well-known fact, that breaking down problems into parts is a basic principle of Computer Science (CS), the core aspects of *Part-Whole-Thinking* (PWT) must be considered when discussing about pursued inclusions of CT skills in educational contexts. In this paper, the role of PWT in the context of CT is discussed and presented.

A literature review on suitable definitions for the terms CT and PWT is presented in the following Sec. 2. Afterwards, it is discussed how PWT; CT, and Computer Science Education (CSE) refer to each other in Sec. 3, before a summary is given and an overview on work to be done in the future is presented in Sec. 4. Considering these aspects, the contribution at hand aims at presenting the massive role that PWT plays in the context of CT.

2. THINKING ABOUT THINKING

Thinking is generally seen as a cognitive process that “allows humans to make sense of, interpret, represent or model the world they experience, and to make predictions about that world” (Kisak, 2015). This *mental act* leads to an acquisition of knowledge, a development of thoughts, and a formulation of reasons (Presseisen, 1991, p. 56). Besides, thinking generates “higher processes, like judging, problem solving, or conducting critical analyses” (Presseisen, 1991, p. 56). A huge emphasis in *thinking skills* is on *reasoning* as a major cognitive skill, “although *cognition* may account for several ways that something may come to be known – as in perception, reasoning, and intuition” (Presseisen, 1991, p. 56). One of the involved thinking skills getting more and more notice in discussions on possibilities to equip students with skills enabling *Life Long Learning* is *Computational Thinking* (CT). CT is defined in the following Sec. 2.1. Through cognitive processes like thinking, complex relationships, which “may be interconnected to an organized structure and may be expressed by the thinker in a variety of ways” (Presseisen, 1991, p. 56), are developed. Presseisen classifies the essential thinking skills involved in these cognitive processes and identifies the detection of Part-Whole-Relationships as one of them (Presseisen, 1991, p. 58). The involved ability to think in parts, wholes, and their relationships to each other is described as *Part-Whole-Thinking* (PWT) in the following Sec. 2.2.

2.1. Computational Thinking

Since Wing introduced the term “Computational Thinking” for the first time in 2006 (Wing, 2006), there has been a huge confusion about its exact definition (Selby, 2015, p. 81). Thus, it is no wonder that many different authors define this term differently in their publications. Many of these definitions “suggest that CT relates to coding or programming” (Shute, Sun, and Asbell-Clarke, 2017). By presenting three publications of Wing and two of authors that discuss her definition of CT it is shown that “considering CT as knowing how to program” (Shute, Sun, and Asbell-Clarke, 2017) definitely *is* too limiting. Instead of CT skills just being needed by programmers and software developers, all pupils should acquire CT skills in school to act responsibly in the *Digital Age* in both their future working and everyday lives.

2.1.1. Wing (2006, 2008, 2010)

When Wing was the first to coin the term “Computational Thinking” in her article of the same title in 2006, she originally presented her work with the subtitle “It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use”

(Wing, 2006, p. 33). According to her “the essence of Computational Thinking is *abstraction*” (Wing, 2008, p. 3717), which “focuses on modeling the workings of a complex problem/system” (Shute, Sun, and Asbell-Clarke, 2017, p. 4). It involves (Wing, 2008, as cited in Shute, Sun, and Asbell-Clarke, 2017, p. 3):

- (a) *abstraction in each layer*,
- (b) *abstraction as a whole*, and
- (c) *interconnection* among layers

The abstraction process is the “most important and high-level thought process in computational thinking” (Wing, 2010, p. 1) to her. As Wing describes, “abstraction gives us the power to scale and deal with complexity” (Wing, 2010, p. 1), while “it is defined as the ability to decide what details of a problem are important and what details can be ignored” (Wing, 2008, as cited in Selby, 2015, p. 81). Thereby the “layers of abstraction [...] reduce the level of complexity of a problem or a representation (Selby, 2015, p. 81).

Wings definition of *abstraction* in the context of CT is very close¹ to the one of (*problem*) *decomposition* (cf. Sec. 2.1.4.), which is another aspect being part of CT according to many authors as presented in the following Sec. 2.1.2. and 2.1.3.

2.1.2. Selby (2015)

The definition of CT Selby presents includes

- *decomposition*, which is “breaking down into smaller [...] parts” (Selby, 2015, p. 81),
- *abstraction*, which is “the ability to decide what details of a problem are important and what details can be ignored” (Wing, 2008, as cited in Selby, 2015, p. 81),
- *algorithm design*, which “is related to the idea of procedural thinking [...] [and defined] as a step-by-step set of instructions that can be carried out by a device” (National Research Council, 2010, p. 11, as cited in Selby, 2015, p. 81),
- *generalization*, which is a “powerful component of problem solving [...] [and] describes the ability to express a problem solution in generic terms” (Selby, 2015, p. 81), and
- *evaluation*, which is “the ability to evaluate processes, in terms of efficiency and resource utilisation, and the ability to recognise and evaluate outcomes” (L’Heureux, et al., 2012, as cited in Selby, 2015, p. 81).

According to her, these skills are “necessary for applying the tools of computer science to *understanding* the world around us” (Selby, 2015, p. 80).

2.1.3. Shute, Sun, and Asbell-Clarke (2017)

Shute, Sun, and Asbell-Clarke worked out five cognitive processes/components of CT that are engaged “with the goal of solving problems efficiently and creatively” (Shute, Sun, and Asbell-Clarke, 2017, p. 3) as stated by Wing (2006) for their part:

1. *problem reformulation*: “Reframe a problem into a solvable and familiar one” (Shute, Sun, and Asbell-Clarke, 2017, p. 3)
2. *recursion*: “Construct a system incrementally based on preceding information” (Shute, Sun, and Asbell-Clarke, 2017, p. 3)
3. *problem decomposition*: “Break the problem down into manageable units” (Shute, Sun, and Asbell-Clarke, 2017, p. 3)
4. *abstraction*: “Model the core aspects of complex problems or systems” (Shute, Sun, and Asbell-Clarke, 2017, p. 3)
5. *systematic testing*: “Take purposeful actions to derive solutions” (Shute, Sun, and Asbell-Clarke, 2017, p. 3)

2.1.4. Comparison and Summary of the Definitions of Computational Thinking

As this very brief literature review on profound definitions of CT already suggests, a huge part in CT skills is derived to the *decomposition* of whole systems into its componential parts. The ability to *decompose* is “required when dealing with large problems, complex systems, or complex tasks” (Selby, 2015, p. 81). Thereby “the divided parts are not random pieces, but functional elements that collectively comprise the whole system/problem” (Shute, Sun, and Asbell-Clarke, 2017, p. 12). The parallels to PWT are more than obvious at this point. But additionally, core aspects of PWT can be found in the understanding and definition of *abstraction* in the context of PWT, which the second aspect that each of the presented publications (cf. Sec 2.1) see as a part of CT. The ability to *abstract* includes the identification of “patterns/rules underlying the data/information structure” (Shute, Sun, and Asbell-Clarke, 2017, p. 12) amongst others. Again, this definition is very close to the understanding of PWT as defined in the following Sec. 2.2.

2.2. Part-Whole-Thinking

The almost endless variety of objects and living things in our world forces us as human beings, which are only equipped with limited cognitive resources, to map cognitive categories. The task of these “category systems is to provide maximum information with the least cognitive effort” (Rosch, 1978, p. 28). Since the objects in the world as we perceive it are in any ways structured by nature, “one decisive aspect of our thinking is the ability to detect similarities and differences between these various elements and then cognitively grouping them based on their differentiations and classifying them into categories”

¹ Especially the separation between abstraction in each layer, abstraction as a whole, and the interconnection among layers is very close to the basic idea of PWT (cf. Sec. 2.2).

(Tversky and Hemenway, 1984, as cited by Pancratz and Diethelm, 2018). These conceptual hierarchies are organized by subconsciously identifying, which parts the respective objects are made of (Tversky and Hemenway, 1984).

“The part-whole relation plays an important role [...] in knowledge processing, e.g. reasoning about objects” (Gerstl and Pribbenow, 1995, p. 865), and beyond: Generally, Part-Whole-Relations help “understanding objects, systems, processes, definitions[,] and concepts” (Pancratz and Diethelm, 2018) by “identifying the parts that constitute the whole, the function of each individual part and its contribution to the function of the whole” (Rao, 2005, p. 174). Views on the functionalities and principles of complex objects and systems are developed based on the knowledge about the single parts and their relationships to each other (Gerstl and Pribbenow, 1995, p. 867). In the context of our research we define this cognitive – and often subconsciously happening – process of partitioning as *Part-Whole-Thinking* (PWT). It is significant for many reasons: “knowing the parts of a whole, how the parts are determined, how they are related, and what they do is a crucial part of understanding the whole” (Tversky, Zacks, and Hard, 2008, p. 437 f.).

According to Tversky, Zacks, and Hard (2008) the following questions need to be considered when discussing and analyzing PWT processes:

- *Wholes*: How are wholes determined – that is, how are they distinguished from backgrounds?
- *Parts*: How are wholes partitioned into parts, and on the basis of what kind of information? Parts may be further partitioned into subparts; do the same bases for partition hold for the subparts?
- *Configuration*: How are the parts of the whole arranged?
- *Composition*: Each whole entity has a set of parts, which may be parts of other wholes as well. How does the entire set of parts get distributed to wholes?
- *Perception-to-function*: Are there relations between perception and appearance on the one hand and behavior and function on the other?” (Tversky, Zacks, and Hard, 2008, p. 437 f.)

3. HOW PART-WHOLE-THINKING, COMPUTATIONAL THINKING, AND COMPUTER SCIENCE (EDUCATION) REFER TO EACH OTHER

Breaking down problems into parts is a basic principle of CS. Typical examples are (Pancratz and Diethelm, 2018):

- the programming paradigm *Object Orientation*
- the algorithmic strategy *Divide and Conquer*
- the logical partitioning in software design called *Modularity*

Besides, many Information Technology (IT) devices, systems, and concepts make use of Part-Whole-Relationships (Pancratz and Diethelm, 2018):

- *The Internet* consists of many different servers, clients, and routers.
- *Computers* have processing units, graphic cards, motherboards, and storage units.
- *Algorithms* are composed of a finite number of well-defined steps.
- *Relational Databases* consist of various tables and relations.

These two lists can easily be stretched. Generally speaking, “part-whole relations often play a fundamental role in the modeling of information systems” (Guarino, Pribbenow, and Vieu, 1996, p. 257).

As depicted in Sec. 2.1, “Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science.” (Wing, 2006, p. 33). As the just given examples show, many CS concepts make use of PWT. Therefore, CSE could provide the perfect showcase to equip students with this essential thinking skill. Thus, it is quite criticizable that the focus of education in schools lies on conveying content (“what to think”) instead of teaching critical thinking skills (“how to think”) so far (Rao, 2005, p. 173), though thinking skills like CT enable us to acquire further knowledge on our own amongst other things (cf. Sec. 2.1). Rao for example noticed an improvement in learners’ cognitive learning processes when explicitly teaching them to use CT skills like PWT in class (Rao, 2005, p. 177).

PWT can especially be found in two of the core concepts of CT: While the definition of (*problem*) *decomposition* obviously fits very well to the core concepts of PWT, even the ways in which *abstraction* in the context of CT can be understood imply the close role that PWT plays in CT (cf. Sec. 2.1.1). In the end, CSE provides the perfect platform to include the fruitful skill of PWT.

4. SUMMARY AND FUTURE WORK

We are more and more surrounded by IT devices that rashly change and massively influence the Digital World we live in. Therefore, it is becoming progressively important to obtain further knowledge by oneself in order to succeed in one’s personal and working life. Discussions about this topic include the term *Life Long Learning* (LLL) recently. The possibilities of CT for LLL are obvious. In this paper, the massive role of PWT in the context of CT is presented.

To the authors of this paper, CT is not only thinking *like* a computer (scientist) to solve problems, but also to become acquainted with the *basic principles of CS* and IT devices and thereby grasp objects, systems, processes, definitions, and concepts of the most different disciplines (and not only CS). Since a massive amount of CS principles makes use of PWT aspects, we suggest to always have the underlying

Part-Whole-Relationships in mind when discussing, planning, and applying CT skills in educational contexts.

With this in mind, it is remarkable that according to Selby decomposition is the most difficult CT skill to master (Selby, 2015, p. 84). According to her, “teachers indicate that learners struggle with implementing the process of decomposition” (Selby, 2015, p. 85). The reasons for this fact “include a lack of experience, incomplete understanding of the problem to solve, and the order of teaching programming” (Selby, 2015, p. 85). Though students seem to understand the concept of breaking a problem down, they are “able to use the skill [...] more successfully in situations where they already know the solution or understand the problem very well” (Selby, 2015, p. 85). Selby points out that “understanding decomposition [...] is a prerequisite for abstraction, algorithm design, and evaluation” (Selby, 2015, p. 85). “As such, it must be mastered, to some extent, before the complexity of the following levels can be accessed” (Selby, 2015, p. 85).

The fact that decomposition is a prerequisite to the other aspects of CT already answers one of the challenges that Wing posed in 2008: “What would be an effective ordering of [CT] concepts in teaching children as their learning ability progresses over the years?” (Wing, 2008, p. 3721). The authors of this paper assume that an early on teaching of PWT has huge potential to improve the outcomes of education. Another challenge Wing describes is that “we do not want people to come away thinking they understand the concepts because they are adept at using [...] tool[s]” (Wing, 2008, p. 3721). She clarifies this challenge with the example of “using a calculator versus understanding arithmetic” (Wing, 2008, p. 3721). Again, this shows the importance of CSE in the *Digital Age*: The imagination of people being surrounded by technical artifacts they don’t understand simply is alarming. A proper knowledge in CS is becoming more and more important. With the paper at hand it is suggested that a more vigorous inclusion of PWT in CSE improves the overall understanding of our students. In order to achieve this, our future work lies on investigating PWT in CSE alongside the *Model of Educational Reconstruction* (Diethelm, Hubwieser, and Klaus, 2012).

5. REFERENCES

- Diethelm, I., Hubwieser, P., and Klaus, R. (2012). *Students, Teachers and Phenomena: Educational Reconstruction for Computer Science Education*. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. ACM. 164-173
- Gerstl, P. and Pribbenow, S. (1995). *Midwinters, end games, and body parts: a classification of part-whole-relations*. In *International Journal of Human-Computer-Studies*, 43(5), 865-889
- Guarino, N., Pribbenow, S., Vieu, L. (1996). *Modeling parts and wholes*. In *Data & Knowledge Engineering*, 20(3), 257-258
- Kisak, P. F. (2015). *Categories of The Thought Process*. North Charleston, South Carolina (USA): CreateSpace Independent Publishing Platform
- L’Heureux, J., Boisvert, D., Cohen, R., and Sanghera, K. (2012). *IT Problem Solving: An Implementation of Computational Thinking in Information Technology*. In *Proceedings of the 13th Annual Conference on Information Technology Education*. Calgary, Alberta, Canada: ACM, 183-188
- National Research Council (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. Washington D.C.: The National Academic Press
- Pancratz, N. and Diethelm, I. (2018). *Including Part-Whole-Thinking in a Girls’ Engineering Course through the Use of littleBits*. In *IEEE Global Engineering Education Conference (EDUCON)*.
- Presseisen, B. Z. (1991). *Thinking skills: Meanings and models revisited*. In Arthur L. Costa, editor, *Developing Minds, Volume 1, Alexandria, Virginia: Association for Supervision and Curriculum Development*, 56-62
- Rao, K. (2005). *Infusing Critical Thinking Skills into Content of AI Course*. SIGCSE Bull., 37(3), 173-177
- Rosch, E. (1978). *Principles of categorization*. In Rosch, Eleanor and Lloyd, Barbara B. (Eds.), *Cognition and Categorisation*, 27-48. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1978.
- Selby, C. C. (2015). *Relationships: computational thinking, pedagogy of programming, and Bloom’s taxonomy*. In *Proceedings of the Workshop in Primary and Secondary Computing Education (WiPSCE ’15)*. New York, NY: ACM, 80-87
- Shute, V. J., Sun, C., and Asbell-Clarke, J. (2017). *Demystifying computational thinking*. In *Educational Research Review*, 22, 142-158
- Tversky, B. and Hemenway, K. (1984). *Objects, parts, and categories*. In *Journal of Experimental Psychology: General*, 113, 169-193. American Psychological Association, Inc., Jun 1984.
- Tversky, B., Zacks, J. M., and Hard, B. M. (2008). *The structure of experience*. In T. F. Shipley and J. M. Zacks (Eds.), *Oxford series in visual cognition: Vol. 4. Understanding events: From perception to action*, 436-464
- Wing, J. M. (2006). *Computational Thinking*. In *Communications of the ACM*, March 2006, 49 (3), 33-35
- Wing, J. M. (2008). *Computational thinking and thinking about computing*. In *Philosophical Transactions of the Royal Society A*, 2008, 366, 3717-3725
- Wing, J. M. (2010). *Computational Thinking: What and Why?* Link Magazine, 2010

創客奇航-遊戲任務導向之運算思維活動設計初探

黃淑賢, 陳虹如, 葉芯妤, 蔡一帆, 施如齡*

國立臺南大學數位學習科技學系

shuhsienhuang@gmail.com, a91081469@gmail.com, quill4749@gmail.com, cristal8505054@gmail.com,

*juling@mail.nutn.edu.tw

摘要

對於社會快速成長的科技與經濟，STEM 教育的推動已成為全球教育的趨勢。本研究將 STEM 教育運用創客概念結合 Arduino 設計結合遊戲任務導向及運算思維的學習活動。學習活動性質包含運算思維的基礎程式學習、問題解決與競爭學習等面向。程式學習使用 mBlock 圖形化程式語言結合多變化性的 Arduino 進行軟硬體整合，應用在船體航行路線的設計。在學習活動中，期望本研究能提升學生對於程式的認知與興趣並學習運算思維的概念，引導學習者自發思考並將想法付諸實行。本研究實驗結果顯示，此活動有助於程式能力以及空間概念的提昇，能讓原本對於學習程式感到困難的人也提起對於撰寫程式的興趣。

關鍵字

STEM；Arduino 微控制器；運算思維；競爭學習；遊戲式學習

1. 緣由與目的

運算思維的能力可藉由 STEM 教育來實施 (林育慈、吳正己, 2016)。STEM 教育主要強調未來的學生應培養跨領域素養與解決問題的能力。現今許多學校將 STEM 教育融入課程與教育政策中來提升學生科技發展的競爭力，讓學生不僅具備知識，更具備解決問題的能力，懂得實證精神，以及能將各種資訊整合為可用資源的思維 (Huang, Tseng & Shih, 2017)。STEM 的重點概念為以「學生為本」的教學法，培養學生創造、協作和解決問題的能力、創新思維，建立學生的開拓與創新精神。STEM 具備教育改革的積極意義，力圖打破理科偏重課堂和傳統教育模式，釋放學生的自主學習精神。

為了增加學生的學習動機，本研究以遊戲任務的方式進行，藉由遊戲式學習影響學習者互動性引發內在學習動機，透過同儕之間的互動溝通以及操作經驗的回饋，進而增進其學習效果。然而，在真實的社會中，競爭是常見的社會現象；此外，在教學環境中，教師也經常使用競爭的心理來激發學生的學習成效與動機 (Lin, Huang, Shih, Covaci, & Ghinea, 2017)。競爭學習是指學習者在學習活動中和其他學習者互相比較、抗衡以打敗對手成就自己的成功來達成設定的目標。有鑑於此，本研究設計一套創客奇航-遊戲任務導向之運算思維活動，以培養學生的運算思維。藉由程式設計的概念，製作出實體船隻，使之在水中航行；將遊戲任務、STEM 教學及運算思維，融入於競速以及賽道的變化。透過遊戲競賽的方式，讓學習者使用圖形化程式語言 mBlock 連結 Arduino，以手機應用程式操控船隻的航行，學習程式語言的撰寫，培養方向和空間的概

念。希望運用遊戲任務及 STEM 教學，將運算思維融入教學活動中，讓學習者在教學活動中除了能習得課程知識外，也能學習運算思維的能力。

2. 文獻探討

2.1. 遊戲式學習 (Game-Based Learning)

遊戲式學習所建立的學習成效，主要是來自於學習者在遊戲中所得到的經驗以及立即的回饋，在遊戲中要引發的是競爭和合作的精神，且是好玩、可達成與富挑戰性 (Prensky, 2003)。

其中競爭學習為遊戲式學習裡常見的模式之一，是指學習者在學習過程中與他人做比較，互相抗衡，以別人的失敗造就自己的成功，以達到某一個目標 (黃政傑、林佩璇, 1996)。在這樣的情況下，在人與人之間存在著消極的互賴關係、視其他他人為競爭對手、使自己在競賽中獲得有利的位置或資源。但競爭學習也被證實會使學生在學習過程中產生焦慮、引發學生自私的心理、使學習低成就的學生感覺到低落、同儕間的相處處於敵對的狀態，進而使學生備感壓力。Johnson 和 Johnson (1991) 為了改善上述情況也提出一些建議：例如在競爭活動時，學習者常只為了贏過對手，卻忘了學習的樂趣。因此，在施行競爭活動時教師可搭配有趣的小遊戲降低競爭所帶來的焦慮感，為學習帶來歡樂的氣氛並告知學生學習樂趣比輸贏更為重要；或者，採用組間競爭，而不強調個體間競賽，在特定情況下，組間競爭可有不錯的學習成效。

除此之外，競爭的策略更被廣用於教學活動上，以激起學生的學習動機。Johnson 和 Johnson (1987) 發現，所有的學生會把 85% 以上的作業用競爭的方式完成。由此可知，在教育上，競爭可被用於提高學生的學習動機。

2.2. STEM 教育

STEM 教育集結了四個學科，分別是科學 (Science)、科技 (Technology)、工程 (Engineering) 及數學 (Mathematics)，鼓勵發展問題解決式、探索發現式學習的課程模式，這個模式要求學生積極參與，以尋求解問題的解決方法。STEM 教育乃強調綜合運用科學、工程、技術與數學等知能，解決日常生活中的問題，在學習上具有探究性與統整性，而正因為它涉及生活中現實的問題，所以也具有趣味性與挑戰性。

2.3. 運算思維 (Computer Thinking)

運算思維 (Computer Thinking) 是一種分析的思維，也是製定問題所涉及的思考過程，藉由數學思考 (Mathematical Thinking) 來解決問題的方式，運用科

學思維使得電腦或機器人能夠有效地執行 (Wing, 2014)。

運算思維是解決問題的方法，當面對複雜的問題，能夠理解問題本質、發展可能的解決辦法。在運算思維中，有四個基礎：分析，將複雜的問題拆解成容易理解與分類的部分；模式識別，找出問題之間的相似之處；抽象，將重要的部分列出，忽略不重要的部分；演算，為每個問題找尋解決的步驟。這四個方式能讓電腦和人明白與理解如何處理問題。而學習程式語言就是將這四種方式，有系統的學習與組合，並解決問題。

綜合上述之相關文獻數位遊戲式學習近年來在輔助教學上受到廣泛討論與應用，而培養邏輯思維能力也在台灣蔚為風潮。本研究運用運算思維融入 STEM 教育來設計教學活動，希望增加學習者問題解決與創新等能力之培養，並將競爭導入遊戲式學習活動中期望藉此提高學習者的學習成效及改善學習者的學習心態，並提升運算思維的能力。

3. 系統設計

本研究使用的是 Android 系統，Android 系統為開放性系統，軟體支援多樣化，適合運用在自行開發軟體之運用，設計一套可在水上航行的船，其系統架構如圖 1 所示。

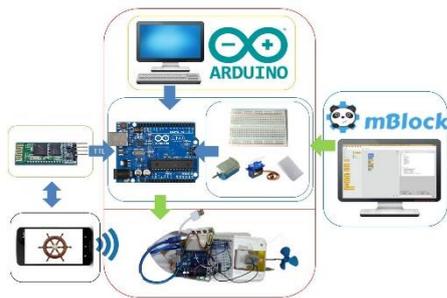


圖 1 為系統架構圖

搭配 MIT 所開發的 App Inventor 設計操控裝置之應用程式，其功能包含：手機應用程式與 Arduino 船體藍芽裝置配對、船體基本航行功能、船體零件改裝參數變化確認。使用者可在遊戲活動中，藉由應用程式的操作與其他學習者進行競速以及船隻航行的控制，藉此學習到零件配置對船體性能的影響。

4. 活動設計

其學習活動分成二個階段，第一階段為學習者遙控測試的參與 (Engage)，此階段為激發學習者的學習興趣，讓學習者熟悉船隻的基本操作，透過遊戲的方式提升學習者的參與程度，其過程中經由探索 (Explore) 與解釋 (Explain) 階段，使學習者理解課程的主題為程式撰寫船隻運行航道，並且由玩家所創造出的變化場域，練習程式撰寫。並思考先前操作過的經驗，不斷提升對於航行路線規劃與方向感。在此階段學習者透過策略性競爭，刺激學習者對於時間、空間與邏輯概念的整合。之後第二階段則進行實作 (Engineer) 與深化 (Enrich) 的運算思維階段，讓學習者實際根據指

定赛道編寫船隻航行的程式，藉由實作了解課程主題的核心，讓學習者有更深度的探究。並經由程式編寫後直接反饋在船體的運行，以加深對於程式學習內容的吸收。因此，學習者完成地形觀察後，根據自己的路線規劃編寫程式。因為是水上航行，所產生的變數較多，所以玩家必須根據船隻狀態已及船與水的互動狀態進行應變，以增加對於問題解決能力的訓練。如果失敗了則記取經驗對於程式進行改寫，直到到達終點為止。此階段學習者可以不斷改寫程式與下水測試，並思考不同路線的規劃，利用最短時間到達終點者為贏家。為了評估學習者在此活動的成效，最後將進行成效評估 (Evaluation)，經由填寫活動滿意度問券以瞭解學習者透過其活動的設計，達到的學習成效為何。

5. 實驗結果與分析

本研究實驗受測者為 20 位大學生，10 位為資訊相關科系的學生，而另 10 位為非資訊相關科系的學生。受測者在撰寫程式的過程中，必須依據地形規劃航行路線，並且根據上次的錯誤進行反覆的更正。因此學生在過程中需要預判船隻在空間中的位置，判斷航行方向輔助程式的撰寫，並且由錯誤中來回更正與學習，藉此提升學習者對於程式的熟悉度，下圖 2 為自動航行程式撰寫的實驗照。

從圖中可以發現，學習者在撰寫過程會模擬船隻航行的方向，而船舵轉彎的角度會影響空間概念，所以可以看到學習者會有肢體的動作。再者，對於空間的航行距離及轉彎角度概念較不佳的學習者，下水測試的次數就會變多。



圖 2 自動航行程式撰寫實驗照

從實驗的觀察中可以發現，資訊相關科系學生由於有程式撰寫的經驗，因此在撰寫控制船隻航行方向的程式時較容易上手，整個學習活動時間平均為 45 分鐘，在判斷船隻位置時也較快速，因此撰寫次數較少。而非資訊相關科系學生因對於程式撰寫較不熟悉，需要反覆操作才能完成學習活動，所以平均花費時間為 60 分鐘，但非資訊相關科系的學生在整個活動中參與非常踴躍，且願意多次嘗試；由此可見整體學習活動並沒有對非資訊背景的學生帶來太大的負擔。

另外，為了瞭解學習者對本研究活動設計的看法，使用問卷來調查，經由統計分析結果顯示，此問卷統計 Cronbach's $\alpha = .876$ ，表示此問卷有高度的可信度。其活動效益的部份結果如表 1。

表 1 活動效益描述性統計資料

題目	M	SD
1.此活動能使我對寫程式產生興趣	4.25	.716
2.此活動能幫助我了解程式的基本概念	4.40	.681
3.此活動能提升我的方位概念	4.30	.657
4.此活動能提升空間布局與設計	4.20	.696
5.此活動能增進我對於速度的測量與計算	4.15	.671
6.這類活動能提升我對物理概念的興趣	3.70	.865

其中大部分學生皆認為此活動有助於程式能力以及空間概念的提升，且認為此活動相當有趣，能讓原本對於學習程式感到困難的人也提起對於撰寫程式的興趣。此外，非資訊相關科系學生在活動流程中的學習力，與資訊相關科系學生是相近的，此活動對他們來說並沒有太大的負擔，且非資訊相關科系學生在活動中撰寫程式的速度以及嘗試的次數非常多，相當踴躍參與。在開放式問卷的回覆中，有幾位受測者提出關於船隻穩定性的建議，船在航行時，由於藍芽以及供電力的不穩定，導致遙控的靈敏度下降，因此本研究期望依照受測者所提出的建議對船的零件穩定性進行改善。

6. 結論

本研究以 Arduino 製作出一艘實體船搭配遊戲任務導向之運算思維的學習活動，學習者經由自動航行計分賽，讓學習者判斷船隻的空間、方位及運算思維，過程中經由圖形化程式編輯介面撰寫路徑程式，藉以增加學習者程式學習的興趣、增進學習者的運算思維，同時增加空間及方位的概念。

實驗活動中發現，非資訊相關科系學生較不具程式學習經驗者，對於本研究所設計之活動皆具有極高興趣，並踴躍參與且順利完成遊戲任務。這樣的結果也顯示，其實驗對象可以向下延伸至國中小，培養他們的方向、空間及運算思維的能力。

但在進行實驗的過程中，受測者大多有提出遙控船船體需要更加堅固、改善其穩定性與速度控制的意見，且活動場地的規畫需要能隔絕自然阻力，才不會因為開放空間若是風大會影響船隻的航行。

另外，透過 Arduino 的控制板結合 Maker 的概念，讓學習者能透過貿易遊戲取得實體零件，藉由手中現有的零件經由插件的方式改造船體，進而影響 Arduino 的馬達參數與旋轉幅度參數，使船體在速度與馬達轉幅等性能具有更多的靈活性。以此讓學習者實際測試各個功能與大小不同的零件對於水的阻力，方向的控制等影響，並且透過反覆操作經驗與創意運用使船達到最高效能，希望能從中培養學習者自造創新與解決問題的能力，並且透過結合數位遊戲進行學習活動，藉以增進學習成效。

7. 致謝

本研究承蒙科技部 106-2813-C-024-029-U 與 MOST104-2628-S024-002-MY4 專題研究計畫之經費補助，謹此感謝。

8. 參考文獻

- 林育慈、吳正己。(2016)。運算思維與中小學資訊科技課程。*教育脈動*，(6)，5-20。
- 林長信、許于仁、施如齡。(2012)。跨平台探索教育數位遊戲之合作學習課程設計。臺灣數位學習研討會 (TWELF 2012)，2012 年 10 月 26 日。臺南：成功大學。
- 黃政傑，林佩璇。(1996)。合作學習。台灣：五南圖書出版公司。
- 蕭嘉琳。(2016)。使用遊戲式實體互動介面提升幼兒運算思維能力。中原大學資訊管理研究所學位論文。
- Fletcher, S. (2011). *The impact of the 6E model in a third grade science classroom* (Doctoral dissertation, Bowling Green State University) .
- García-Peñalvo, F. J. (2016). What computational thinking is.
- Halverson, E. R., & Sheridan, K. (2014). The maker movement in education. *Harvard Educational Review*, 84 (4), 495-504.
- Huang, S. H., Tseng C. C., & Shih, J. L. (2017) The Design and Evaluation of a STEM Interdisciplinary Game-based Learning about the Great Voyage. The 25th International Conference on Computers in Education. New Zealand.
- Lin, C. H., Huang, S. H., Shih, J. L., Covaci, A., & Ghinea, G. (2017, July). Game-Based Learning Effectiveness and Motivation Study between Competitive and Cooperative Modes. In *Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on* (pp. 123-127). IEEE.
- Johnson, D. W., & Johnson, R. T. (1987). *Learning together and alone: Cooperative, competitive, and individualistic learning*: Prentice-Hall, Inc.
- Johnson, D. W., & Johnson, R. T. (1991). Cooperative learning and classroom and school climate. *Educational environments: Evaluation, antecedents and consequences*, 55-74.
- Prensky, M. (2003). Digital game-based learning. *Computers in Entertainment (CIE)*, 1(1), 21-21.
- Wing, J. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014.

Examining a Secondary School Computational Action Curriculum Using App

Inventor and the Internet of Things

Mike TISSENBAUM, Josh SHELDON, Hal ABELSON, Mark SHERMAN

Massachusetts Institute of Technology

miketissenbaum@gmail.com, jsheldon@mit.edu, hal@mit.edu, shermanm@mit.edu

ABSTRACT

This paper outlines a study in which we integrate computational action – a pedagogical shift in computing education towards educational designs that focus on students learning about, and creating with, computation in ways that connect to their lives and communities – into engineering and design classes at a large urban high school. This paper also outlines methodological approaches for understanding how a computational action curriculum can change students' perceptions of their computational identities and digital empowerment.

KEYWORDS

computational action, computational thinking, digital empowerment, computational identity, mobile computing

1. INTRODUCTION

Current approaches to computational thinking have largely followed Wing's (2006) model, which advocated for teaching computing with a focus on the "fundamentals" of programming, such as loops, variables, conditionals, data handling, and parallelism. However, subscribing to only this approach threatens to decontextualize computing education from the real-lives of learners, making them feel that it isn't something they need to learn, believing they won't need to use it in the future – a problem regularly faced by in math and physics (Williams et al., 2003; Flegg et al., 2012). In response, our work suggests an alternate framing of computing education that focuses on *computational action*. Computational action posits that young people should learn about, and create with, computing in ways that provide them the opportunity to have direct impact in their lives and their communities (Tissenbaum, Sheldon & Abelson, submitted).

Below we outline the theoretical foundations for computational action and outline the design of a high school curriculum that uses computational action to empower traditionally underrepresented students to use computing to have an impact in their communities.

2. COMPUTATIONAL ACTION

While approaches such as problem-based learning (Kay et al., 2000) have attempted to situate computing education in real-world contexts, they are often generic (e.g., designing supermarket checkout systems) and fail to connect to students' personal interests and needs.

While important for all students, the need to feel their work has the potential to have an impact in their lives and communities, is particularly critical for young women and groups traditionally underrepresented in computing and engineering (Pinkard et al., 2017). By refocusing computing

education into the real lives of learners we can help them feel empowered to use computing to effect change and to pursue career paths that employ computational problem solving.

We have termed this shift toward educational designs that focus on students learning about, and creating with, computation in ways that connect to their lives and communities *computational action*. To understand how to design and support learner engaging in computational action, we suggest it comprises of two key dimensions: computational identity and digital empowerment (Tissenbaum et al., 2017). Computational identity is a person's recognition that they can solve problems using computing and may have a place in the larger community of computational problem solvers. Digital empowerment is the belief that a person can put that identity into action in meaningful and impactful ways.

3. SUPPORTING COMPUTATIONAL ACTION WITH MIT APP INVENTOR

Many of the challenges faced when implementing a computational action curriculum can be attributed to where the learning takes place – traditional computer labs, which are far removed from their everyday lives. With the explosive growth of mobile and ubiquitous computing (e.g., the Internet of Things – IoT), students now have the opportunity to take what they build out into the world. This creates opportunities to contextualize what students can create, and perhaps more importantly, why they create it (Lee et al, 2016).

In addition to environments that allow development for mobile and ubiquitous devices, we also need environments that allow students to quickly build, test, and deploy their creations, and that provide powerful abstractions to harness today's incredible computing infrastructure with minimal previous experience. App Inventor is one such environment, a blocks-based programming language that allows learners to build fully functional mobile apps. App Inventor employs a drag-and-drop designer interface that allows users to layout the front-end (user facing) elements of their apps, abstracting away much of the complicated code usually required. App Inventor also allows users to harness a wide range of software and hardware logic, including creating and storing data locally or in the cloud, or accessing the phone's camera, GPS, or Bluetooth functions. Because it supports creation of mobile apps, can connect to IoT devices, and allows those new to programming to quickly access these and other powerful computational features, while not the only option, we believe App Inventor is particularly well-suited for supporting computational action-focused learning.

4. DESIGNING A COMPUTATIONAL ACTION CURRICULUM

To study how a computational action curriculum might support students as they begin to recognize their capabilities for making a real impacts in their lives using computation, we co-designed, with two teachers, (CITE) a 10-week curriculum for grade-10 students at a large urban American high school.

The students would come from two classrooms, taught by our two co-design teachers. The two classrooms were particularly interesting for a computational action approach. One class was an engineering design class, and the other a traditional computing class. In the computing class, the students would normally learn the basics of JavaScript, HTML, and a light introduction to Java. Additionally, the computing class had an extremely diverse population; nearly half the students were English language learners (ELL). The teachers recognized that these students traditionally felt outside of the computing culture (i.e. did not have strong computational identities) at the school. Thus, the teachers wanted to revamp the computing class to help these students develop their computational identities.

In discussions with the teachers, they identified an issue that was of interest to many of the students at the school: the local river was polluted and the students wanted to develop solutions to clean it up. The local river was ideal context for supporting students to engage in computational action and for them to engage in digital empowerment.

To situate students' projects in authentic contexts, the engineering design class developed IoT approaches for capturing and exploring river data. The engineering students then became the "clients" or partners of the computing class, presenting their designs and asking the computing students to develop apps that could work with and enhance their designs. To facilitate the design process, we adapted the Stanford D-School's design process. We also developed a set of design documents to help the students break down (decompose) their designs into more manageable sub-components. The paired groups met once a week in feedback sessions to coordinate and refine their designs. The curriculum will culminate with the students presenting their work at an annual work fair held at the school, which is attended by students, administrators and city officials.

In order to understand changes in students computational identities, digital empowerment, and computational problem solving skills over the course of the curriculum, we adapted several measures based on our own prior work, and other established identity measures. To understand changes in students computational identity and digital empowerment, we are using a combination of an adaptation Snow et al.'s (2017) validated multiple choice tool for measuring changes in students' CT perspectives, and open-ended reflective statements that previous research (Authors, submitted) has shown to reveal important changes in students perceptions of their ability to use computing to solve real world

problems. Using a combination of field notes, classroom observations, and regular individual interviews and focus groups throughout the intervention, we are developing rich case studies to reveal how students identities changed over time.

5. RESULTS AND DISCUSSION

As this work is currently underway, this poster will report on our early findings and will aim to engage visitors on critical discussions around the role of computational action as a new framing for computing education. We believe this work represents an important shift in what the goals of computing education can be and how we motivate students to be the empowered computational creators of the future.

6. REFERENCES

- Flegg, J., Mallet, D., & Lupton, M. (2012). Students' perceptions of the relevance of mathematics in engineering. *Intl. Journal of Mathematical Education in Science and Technology*, 43(6), 717-732.
- Lee, C. H., & Soep, E. (2016). None But Ourselves Can Free Our Minds: Critical Computational Literacy as a Pedagogy of Resistance. *Equity & Excellence in Education*, 49(4), 480-492.
- Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*, 10(2).
- Pinkard, N., Erete, S., Martin, C. K., & McKinney de Royston, M. (2017). Digital Youth Divas: Exploring Narrative-Driven Curriculum to Spark Middle School Girls' Interest in Computational Activities. *Journal of the Learning Sciences*, (just-accepted).
- Snow, E., Shear, L., Rutstein, D., Wang, H., Iwatani, E., Xu, Y., Basu, S., Tate, C. (September, 2017). *CoolThink@JC Evaluation: Baseline Report*. Menlo Park, CA: SRI International.
- Tissenbaum, M., Sheldon, J. & Abelson, H. (Submitted). From Computational Thinking to Computational Action. *Communications of the ACM*.
- Tissenbaum, M., Sheldon, J., Soep, L., Lee, C.H. Lao, N. (2017). Critical computational empowerment: Engaging youth as shapers of the digital future. *Proceedings fo the IEEE Global Engineering Education Conference*, Athens Greece, April, 1705-1708, Weintrop, D., & Wilensky, U. (2015). To block or not to block, that is the question. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*.
- Williams, C., Stanisstreet, M., Spall, K., Boyes, E., & Dickson, D. (2003). Why aren't secondary students interested in physics? *Physics Education*, 38(4), 324.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35

Computational Thinking and Special Education Needs

The Application of Minecraft in Education for Children with Autism in Special Schools

Wen-wen MU, Kuen-fung SIN*
The Education University of Hong Kong, Hong Kong
wenwenmu@s.eduhk.hk, kfsin@edu.hk

ABSTRACT

This paper aims at identifying the use, benefits and challenges of integrating Minecraft in teaching students with autism. Classroom observations, students-created manifests and interviews were conducted in two Chinese-speaking special schools in Hong Kong. It is concluded that Minecraft does have positive impact on how children with ASD learn. Students were more engaged in class, showed improved collaboration and communications skills, developed deeper relationship with their classmates and the teachers, and were more motivated to learn. Some potential challenges and concerns are discussed.

KEYWORDS

Minecraft, Autism, Special school

1. INTRODUCTION

Autism spectrum disorder (ASD) is defined by two core features that are restricted and repetitive behavior and interest; and impairment in social interaction. Both can negatively affect the academic performance, well-being and social engagement. Researchers have acknowledged that the simulation techniques used in computer/video games would provide significant results for motivation and comprehension, promote engagement and active learning for students including those with special learning needs. (Habgood, Ainsworth & Benford, 2005; Mohammadi & Fallah, 2007; Ke & Abras, 2013). Meanwhile, computer and playing video games are always the favorite learning activities for children with ASD (Eversole, 2016). To accommodate different learning styles and to maximize the learning effect for students with ASD, educators examine the appropriate teaching strategies and content delivery mechanisms that meet mostly the individual preferences of ASD.

Minecraft and its use in education

Minecraft is a "Three-dimensional Lego-like environment in which the user can build and interact with a virtual world" (Bos, Wilder, Cook & O'Donnell, 2014, p. 56). According to Zedda-Sampson (2013), about 40% of kids with ages 8 to 10 play Minecraft. The graphics of Minecraft are intentionally pixelated and blocky, which make them appealing to children, especially those with ASD (Kulman, 2015).

Minecraft has currently emerged as a tool that has clear educational values (Mark, 2015). Many educational activities based on Minecraft have been developed to teach students in subjects including History, Language, Arts, Science, Math, Engineering, Architecture, and Computer coding (Overby & Jones, 2015). Minecraft sparks children's

creativity and imagination, and enhances other important skills such as self-awareness, self-control, flexible thinking, and planning & organization (Kulman, 2015). Hollett and Ehret (2015) stressed that Minecraft helps children express and control their emotions, build strong social ties, enhance peer engagement and promote teamwork. Ringland (2016) stated that autistic population may possibly practice a wide variety of social skills in Minecraft. Furthermore, Minecraft may be considered as a kind of Computer Mediated Communication. For example, within Minecraft, users may communicate with each other by sending text through a chat window or talking with the help of modified accessories. The Minecraft space links tightly to other social platforms such as YouTube, discussion forums, and Wiki software (Pellicone & Ahn, 2014), that helps the social communication among individuals with ASD. They need not to face with the difficulties associated with face-to-face social interaction that requires nonverbal social cues such as eye contact, facial expression, and gestures (Mazurek, Engelhardt & Clark, 2015).

Efforts have been made to promote the use of Minecraft in schools in Hong Kong. In 2014, over 550 local school primary and secondary schools participated in a contest organized by Hong Kong Cyberport. The City University of Hong Kong completed a case-study to explore the teaching and learning of Chinese History in Minecraft in Hong Kong secondary schools (Zhu, 2017).

2. OBJECTIVES OF THE STUDY

In Hong Kong, there are 61 aided special schools with about 7,800 students with special educational needs. 41 of these schools are for students with intellectual disabilities classified into mild, mild to moderate, moderate and severe grades (Education Bureau, 2017). While Minecraft is popularly used in teaching and learning in mainstreaming schools, there is limited research on how Minecraft is used for students in special schools. It is worthwhile to study the use and effectiveness of Minecraft, particularly the strategies, benefits and challenges in teaching students with ASD in special schools.

3. METHOD

As a pilot study, two teachers, one principal and 15 students with ASD from two local special schools for students with mild intellectual disabilities in Hong Kong were invited in this study. All students were male, attending classes from grade three to grade seven.

A semi-structured interview was conducted for examining the use of Minecraft in classroom teaching. The guidelines were prepared with reference to the past work on exploring the use of Minecraft in education (Smeaton, 2012). It aimed

at examining how teachers used the game and incorporating it into their existing teaching practices. Furthermore, more data was collected from the classroom observations, weekly diary for after-class Minecraft interest club and students' "digital footprints". The student-created work in Minecraft and student-managed Minecraft servers were tracked by using the screen captures and recorded videos. The data source from the interview data, observation notes, student-created Minecraft works helped the thematic analysis.

4. FINDINGS

4.1. The use of Minecraft in special schools

Minecraft were used in teaching different subjects such as Visual Arts, Computer, Language, Mathematics and Social Study. Teachers reported topics with architectural and storytelling elements were particularly suitable for using Minecraft. Topics with animals, space and history were also reported.

The schools supported the use of Minecraft by setting up a private Minecraft server in school with restricted access. Only students who have been given the permission can log in the server to play. Since no one else can access the server, students will feel free and safe to socialize and work with each other. Two servers respectively for the new users and experienced players were set up. Students who were new to Minecraft used the server for beginners to play and socialized with their fellow classmates.

In addition to using Minecraft in classroom teaching, teachers also organized the after-school interest club and workshops. A teacher organized a Minecraft workshop with the theme "Smart Home" in the summer vacation. A group of about 6 students with ASD worked together to design and build a smart home for the elderly inside Minecraft.

The Principal attempted to explore the effectiveness of using Minecraft in his school and highly encouraged his teachers to use the tool in the classroom. He started an after-school interest group that met on every Friday. Students worked together to learn Chinese, Mathematics and Social Subjects through Minecraft under the teacher guidance.

Minecraft provided an interesting way for students to learn the 3-dimensional modelling. With some software tools (e.g. 'Mineways', a free and open-source program for exporting Minecraft models for 3D printing), students were able to export what they had built in Minecraft for 3D printing.

It was observed that autistic children often had difficulty in expressing their thoughts in words. Minecraft became a language for them to communicate with others. When the students were building in Minecraft, they were acting out a story in their own mind. And they might tell that story by using screenshots of different Minecraft scenes.

4.2. Benefits

Enhancing collaboration and teamwork

Working with other people is probably one of the most challenging aspects of school life for students with ASD. Effective teamwork requires the students to learn skills such as negotiating, active listening, following directions and accepting criticism. Playing in Minecraft offers a lot of opportunities to develop these skills. Large-scale creation in

Minecraft can seldom be built by a single student. It requires a team of at least 4-6 students, working seamlessly together to complete.

"Students choose their own role based on their own expertise and interest. For example, some students are good at building railways, some are good at building Redstone devices and some are good at crafting building".

When working together in Minecraft, students have many opportunities to discuss with members of their own team or other teams. Building is a truly collaborative effort.

"...during construction, when one student found that he did not have enough space, he would proactively propose to another student and ask for more space"

Even when they are not working together in the same project, the students are still playing in the same virtual environment, trying to ignore distractions and avoid conflicts from the outside world. In school, teachers can teach the students how to work together effectively by planning, building, and presenting a Minecraft project together as a group.

Improving social interaction and developing relationship

Lack of social communication and interaction is a core deficit of students with ASD, and as a result, most of them have difficulties in developing and maintaining relationship with other people. When working together on a Minecraft project, the students must learn to express their needs and opinions, make suggestion, ask for help and negotiate with others.

"A parent shared with me that his child never called his classmates at home. But now when he faced with a problem in completing a task in Minecraft, he would take the initiative to call his classmates for help."

Students are willing to talk and share their interest in Minecraft with peers and teachers. A common interest helps develop new friendship and deepen the relationship among teachers and students.

"When they see their classmates building something interesting, they will go over and ask them how they did it. There is a strong motivation to interact with each other."

"There are WhatsApp groups between me and the students, as well as among the students themselves. And they regularly exchange information about Minecraft. Many of them would report on their tasks and share their creations in Minecraft with me. I am getting closer to my students."

Minecraft is a social game among all the players. The desire of completing and sharing their work in Minecraft encourages students with ASD to practice communication and social skills. The active social interaction and develop deeper relationship with their classmates.

Becoming active learners

Learning through Minecraft encourages the students to be active learners and to take full responsibility of their own learning. They have the freedom to choose what to learn and how they are going to learn it.

"They will go online (for example, YouTube) to find solutions. Even if the video is not in Chinese, they will find a

way to understand the materials. They are learning how to learn independently which is an important 21st century skill.”

“E-learning is not just a one-way instruction from the teacher. Instead, students find the answers to their questions by interacting with others, and develop the spirit of inquiry along the way.”

When creating stories in Minecraft, students must find their own contents that made up the story. In the process of creation, the listening, speaking, writing and logical thinking skills of the students are greatly enhanced.

4.3. Issues and challenges

Online addiction and safety

The online addiction and safety are the concerns. It was noted that teachers restricted the playing time of Minecraft to prevent addiction. Teacher A set the school Minecraft servers to be available from 7 am to 11 pm. Teacher B and the Principal only allowed their students to use Minecraft in school under teacher supervision.

Teachers needed to prevent cyber bullying before it happened. Teacher A decided to set up her own private server to protect the students from potential harassment by strangers. Teachers and students jointly setup playing rules, such forbidding the use of “TNT”, killing of animals, or bullying each other, etc.

Detailed instructional design

It is not an easy task integrate Minecraft in classroom learning. Teachers reported to spend a lot of time on instructional design and material preparation. This was especially true in the beginning when the teachers did not have a lot of experience in using Minecraft.

“I once conducted a project of building a “smart school” in Minecraft. Firstly, I need to guide the students to discuss what should be built and where, what information they need to find out, before they can actually build them.”

In projects that require cooperation among the students, the teacher had to help the discussion, instead of leaving the students on their own. Some teachers also used thinking tools such as mind map to help students discuss the project approach and work allocation. Teachers also needed to have good time management skills and kept reminding the students of the time management.

“They need to think about who the protagonist is, what time the story happens, etc. I need to give them enough instructions or they will get stuck in some parts of the story and neglect the rest.”

When recreating a story in Minecraft based on the story "pig nose elephant", the students had to fully understand the story and then answer some important questions beforehand.

“When a chicken suddenly appeared in the Minecraft virtual world, the students all got excited and joked to burn the chicken. I immediately explained why we should not do that.”

Sometimes the students' reaction was observed to be fierce and brutal. Educators must seize the opportunity to tell the students how they should properly behave. These are all very

challenging tasks that require a lot of experience and wisdom from the teachers.

Home-school cooperation

Parents are the important stakeholders in learning and understanding Minecraft with their children. Many parents worried about their children getting addicted to Minecraft. But some parents were willing to explore how to play the game with their children. It is very important to get the understanding and support from the parents.

“Whether you let them play or not, they will play. You don't know what they're doing if you don't get actively involved. Parents will see that the child is not just playing game, he/she is doing homework assigned by the teachers. Showing the products made by the students to their parents helps.”

As teachers came to understand the benefits of using Minecraft for learning, they began to share this information with parents. Teachers and parents worked together to determine the proper use of Minecraft. Eventually, parents understood that game-based learning under proper guidance really helped their children learn.

5. DISCUSSION AND CONCLUSION

Using Minecraft as an alternative educational tool

Minecraft has been used as the main, optional or supplementary educational tool in many mainstream schools (Petrov, 2014). One major difference in the approach taken by the special schools is the extent to which Minecraft is being used. In these schools, Minecraft is more likely to be used as an alternative teaching tool for students with special needs to express their understanding because these students vary a lot in both their capability and their interests. This is consistent with the philosophy that special education should respect individual differences and emphasize individualized learning.

In the case study, the use of Minecraft is not mandated to the whole class. Both paper-pencil worksheets and other digital tools are also available to students. Students with ASD, however, prefer to use Minecraft over other means. But even for those who have chosen to use Minecraft, they are using it in many ways. Students with lower communication skills may choose to use just screen captures and voice recording to present their work.

Student learning and teacher competency

The two school cases started their Minecraft journey very differently but they achieved the positive outcome in supporting the learning of students with ASD. Teachers reported that they recognized how their students reacted to the use of Minecraft and the impact that Minecraft had on behavior, motivation and learning. They used Minecraft for the benefit of the students.

While playing Minecraft, the students are often the experts. Learning with and from students allows the students to be the center of learning. Previous research and experience using Game Based Learning have shown how useful a Game Based Learning approach can be in creating student-centered learning environment (Motschnig-Pitrik & Holzinger, 2002). Teachers who use Minecraft in their schools must maintain a student-guided mentality for the

best outcome (Petrov, 2014). In this research, the teachers may not be the experts in using the Minecraft but they allow students the full autonomy in managing the school Minecraft server and structuring their learning experience. With this approach, students develop self-learning skills, take more responsibility for their own learning and have more freedom to choose what they want to take.

Even though students can learn by themselves, teachers play an important role in facilitating and supporting the learning of the students. Technology provides many learning opportunities that are both engaging and motivating to the students. However, it will work effectively if teachers integrate it appropriately in the course design.

Some stated that teachers must be familiar with the contents of the video games so that they can use them to support teaching (Barbour, Evans & Toker, 2009). On the other hand, Smeaton (2014) argued that instruction experience is an even more important factor because experienced teachers would be able to deliver knowledge more effectively. Students with ASD demonstrated great motivation when using Minecraft to learn, but they also required strict behavior management from the teacher. One of the reasons why computer games such as Minecraft fails as a teaching tool could be due to the lack of preparation and understanding by the teacher. The experience of the teacher is a crucial factor. According to the research findings, the familiarity with Minecraft is not a decisive factor. The caring of the students and the design of the learning activities are much more important than the teacher's personal interests and skills in the game.

Conclusion

It was concluded that the use of Minecraft does help the learning of students with ASD. The result associated with this practice was positive. Students were more engaged in class, showed improved collaboration and communications skills, developed deeper relationship with their classmates and the teachers, and were more motivated to learn. Despite the benefits of using Minecraft, some major challenges and issues were also identified. The cases presented in this study suggest that Minecraft can be a valuable educational tool in special school and inspire more evidence-based practice and further research.

6. REFERENCES

- Abrams, S. S. (2017). Emotionally crafted experiences: Layering literacies in Minecraft. *Reading Teacher*, 70(4), 501-506.
- Anderson, C. (2017). Minecraft in the classroom. *The Medium (Online)*, 0_1-5.
- Barbour, M., Evans, M. & Toker, S. (2009). Making sense of video games: Pre-service teachers struggle with this new medium. In I. Gibson, R. Weber, K. McFerrin, R. Carlsen & D. Willis (Eds.), *Proceedings of SITE 2009--Society for Information Technology & Teacher Education International Conference* (pp. 1367-1372). Charleston, SC, USA.
- Callaghan, N. (2016). Investigating the role of Minecraft in educational learning environments. *Educational Media International*, 53(4), 244-260.
- Connolly, T. M., Stansfield, M., & Hailey, T. (2011). An alternate reality game for language learning: ARGuing for multilingual motivation. *Computers & Education*, 57(1), 1389-1415.
- Cosh, J. (2015). Minecraft's massive landscape for learning. *Primary Teacher Update*, 2015(43), 20-22.
- Dodgson, D. (2017). Digging deeper: Learning and re-learning with student and teacher Minecraft communities. *Tesl-Ej*, 20(4), EJ, 2017, Vol.20(4).
- Ellison, T. L., & Evans, J. N. (2016). "Minecraft," teachers, parents, and learning: What they need to know and understand. *School Community Journal*, 26(2), 25-43.
- Eversole, M., Collins, D. M., Karmarkar, A., Colton, L., Quinn, J. P., & Karsbaek, R., et al. (2016). Leisure activity enjoyment of children with autism spectrum disorders. *Journal of Autism and Developmental Disorders*, 46(1), 10-20.
- Gallagher, C., Asselstine, S., & Bloom, D. (2015). *Minecraft in the classroom: Ideas, inspiration, and student projects for teachers* Berkeley, CA : Peachpit Press.
- Habgood, M. P. J., Ainsworth, S. E., & Benford, S. (2005). Endogenous fantasy and learning in digital games. *Simulation & Gaming*, 36(4), 483-498.
- Hollett, T., & Ehret, C. (2015). "Bean's world": (mine) crafting affective atmospheres of gameplay, learning, and care in a children's hospital. *New Media & Society*, 17(11), 1849-1866.
- Holzinger, A., & Renate Motschnig-Pitrik. (2002). Student-centered teaching meets new media: Concept and case study. *Educational Technology & Society*, 5(4), 160-172.
- Ke, F., & Abras, T. (2013). Games for engaged learning of middle school children with special learning needs. *British Journal of Educational Technology*, 44(2), 225-242.
- Kuhn, J., & Stevens, V. (2017). Participatory culture as professional development: Preparing teachers to use Minecraft in the classroom. *TESOL Journal*, 8(4), 753-767.
- Overby, A., & Jones, B. L. (2015). Virtual LEGOs: Incorporating Minecraft into the Art education curriculum. *Art Education*, 68(1), 21-27.
- Pellicone, A., & Ahn, J. (2014). Construction and community: Investigating interaction in a Minecraft affinity space. In *Proceedings of the Tenth Conference for Games + Learning + Society - GLS 2014 Madison*, WI: ETC Press.
- Perez, S. (2016). *Microsoft to launch "Minecraft education edition" for classrooms this summer, following acquisition of Learning Game*. New York:
- Preston, S. D. (2008). Putting the subjective back into intersubjective: The importance of person-specific, distributed, neural representations in perception-action mechanisms. *Behavioral and Brain Sciences*, 31(1), 36-37.
- Ringland, K.E., Wolf, C.T. & Hayes, G.R. "Making 'Safe': Community-centered practices in a virtual world

- dedicated to children with Autism". *Proceedings of the 2015 ACM International Conference on Computer Supported Collaborative Work*, ACM (2015).
- Ringland, K.E., Wolf, C.T., Faucett, H., Dombrowskiand, L. & Hayes, G.R. (2016). "Will I always be not social?": Re-conceptualizing sociality in the context of a Minecraft community for Autism. *2016 CHI Conference on Human Factors in Computing Systems: 1256-1269*. ACM New York, NY, USA.
- Ruotsalainen, H. (. (2016). *Designing educational game experiences for k12 students in context of informal minecraft club* University of Oulu.
- Ružic-Baf, M., Strnak, H., & Debeljuh, A. (2016). Online video games and young people. *International Journal of Research in Education and Science*, 2(1), 94-103.
- Steinbeiss, G. (2017). *Minecraft as a learning and teaching tool: Designing integrated game experiences for formal and informal learning activities* University of Oulu.
- Ting, Y. (2015). Tapping into students' digital literacy and designing negotiated learning to promote learner autonomy. *The Internet and Higher Education*, 26, 25-32.
- Tromba, P. (2013). Build engagement and knowledge one block at a time with Minecraft. *Learning & Leading with Technology*, 40(8), 20-23.
- Wu, H. (2016). Video game prosumers: Case study of a Minecraft affinity space. *Visual Arts Research*, 42(1), 22-37.
- Wu, M. (2015). In Dickson P., Lin C., Mishra P. and Ratan R. (Eds.), *Teachers' experience, attitudes, self-efficacy and perceived barriers to the use of digital game-based learning: A survey study through the lens of a typology of educational digital games* ProQuest Dissertations Publishing.
- Zedda-Sampson, L. (2013). Is U a word or do you spell it with a Z? English spelling in Australian schools - are we getting it write? *Literacy Learning: The Middle Years*, 21(2), 4

結合運算思維在國小特殊教育需求的數學教學活動之發展

廖晨惠¹，郭伯臣²，白鎧誌^{2*}，鄔珮甄²

¹ 國立臺中教育大學特殊教育學系

² 國立臺中教育大學教育資訊與測驗統計研究所

chenhueiliao@gmail.com, kbc@mail.ntcu.edu.tw, minbai0926@gmail.com, seashellpeach@gmail.com

摘要

本研究旨在結合運算思維概念幫助國小學習障礙學生進行數學學習活動，本研究依據明確教學原則(Explicit Instruction)的教學模式與考慮學習障礙類別學生的特殊需求(例如：書寫困難、文字理解困難等)，使用數學概念中的空間推理概念設計不插電的校園地圖 PAPAGO 的運算思維教學活動，教導學生如何運用運算思維概念辨認地圖上方向、路徑規劃等數學相關概念。

關鍵字

運算思維；學習障礙；空間推理。

1. 前言

運算思維近年來已受到各國教育上的重視，過去研究提出運算思維不僅侷限於資訊科學領域，例如在閱讀、寫作和算術皆需要用到運算思維(Wing, 2006)，在數學教育領域方面，國際研究也開始關注於如何運用運算思維來學習數學與增進數學能力，並探究如何將運算思維應用於學校的一些數學主題以改善或豐富傳統教學(Sysło & Kwiatkowska, 2016)，顯示運用運算思維於數學教育已備受重視。但運算思維的教學活動在特殊教育需求的學生仍較少研究，相對應的教材仍相對缺乏，因此本研究在於幫助特殊需求的學習障礙學生設計一套結合運算思維教學的數學學習活動，以幫助學生運用運算思維概念進行數學解題。

2. 運算思維在特殊教育領域之探究

近幾年有學者提出在特殊教育領域對於運算思維的教學方式是一項重要且具挑戰性的研究，尤其如何在特殊教育領域導入運算思維的教學是值得探究之研究議題(Snodgrass, Israel, & Reese, 2016)。Barefoot(2016)提出教導特殊需求學生學習運算思維有幾項優點：(1).運算思維是運算課程中重要的核心概念，培養學生問題解決的能力並可應用於課程中，例如分解與調整問題的能力可以應用於數學或是運算問題中。(2).運算思維是一種具有創造性、可行的方法來增進課程中的學習。(3).科技可以幫助有特殊教育需求或是障礙的學生通過學習、資訊與休閒的教學活動。

3. 不插電之運算思維教學活動

過去已有研究針對運算思維教學設計不插電的運算思維教學活動，Bell、Witten與Fellows(2010)設計不插電的活動來教學生資訊科學的概念，例如透過海戰棋讓學生能夠了解運算思維中的演算法，且學生須對於數字大小的排序了解與幾何概念中的圖形探索與空間座標。目前也有幾個網站，例如Code.org、Barefoot、ICT

in Practice 等網站都有設計不插電的運算思維教學活動，顯示針對不插電的運算思維教學活動設計成為近年來的教育研究趨勢之一，但針對特殊教育需求學生的相關教學活動仍非常缺乏。

4. 校園地圖 PAPAGO 教學活動之發展

本研究主要是結合運算思維發展國小特殊教育需求的數學教學活動，其研究對象以學習障礙學生為主，而學習障礙學生又分為數學障礙、閱讀障礙與書寫障礙，因此在設計教學活動時須考慮到這三組障礙類別的特殊需求。

過去研究提到數學障礙的學生多半在訊息處理過程中對於涉及方位或方向的數學解題或數線問題有學習上的困難(邱上真, 2001)，因此本研究設計地圖相關的不插電遊戲進行教學活動，並運用運算思維的概念教導學生。而為了能夠結合學生實際生活中會遭遇之情境，本研究設計了校園地圖 PAPAGO 教學活動，透過巧拼模擬校園地圖，帶學生透過地圖的教學活動來學習運算思維的概念，也希望能夠幫助學生增進其中的數學概念。本研究設計的教學活動所對應之運算思維概念與數學概念如表 1。

本研究參考 Israel 等學者(2015)對於運算思維在特殊教育需求中提出的明確的教學原則(Explicit Instruction)來設計運算思維教學活動，教學活動的設計也需考慮學生學習較弱的部分，例如書寫障礙的學生受限其書寫能力，本研究改用紙牌讓學生進行紙牌排列，此外紙牌上會以圖形化呈現紙牌代表的方向、前進的步數，減輕閱讀障礙的學生對於文字閱讀的困難等，本研究設計的教學課程範例如表 3，依照任務設計的方式，讓學生透過不同任務學習運算思維概念。學生須利用提供的方向與前進步數的紙牌，排列出從起到如何到達指定位置。

5. 未來研究工作

本研究目前已設計一套國小特殊需求學生的運算思維進行數學教學課程，後續將先進行資料蒐集，進行思維教學活動的數學課程教學，從教學活動中以「錄影」或是「觀察紀錄」的方式了解學生對於教學活動與練習單元的反應與學習狀況，並於活動結束後分析學生在任務單元的作答狀況，並對照運算思維概念與行為及數學概念的觀點，了解學生的學習程度，並詢問教學教師之建議，修改所設計的教學課程。

6. 參考文獻

邱上真 (2001)。跨領域、多層次的數學學障研究：從學習障礙的官方定義談起。2001 數學學習障礙研討會手冊，9-41，台北：台灣師範大學。

Barefoot (2016). Activities for pupils with special educational needs. Retrieved from <http://barefootcas.org.uk/activities/sen/>

Bell, T., Arpaci-Dusseau, A., Witten, I. and Fellows, M. (2010). *Computer Science Unplugged: Understanding Computing Through Games and Puzzles*. Hubei: Huazhong University of Science and Technology Press. (Authored Books).

Israel, M., Pearson, J., Tapia, T., Wherfel, Q., & Reese, G. (2015). Supporting all learners in school-wide

computational thinking: A cross case analysis. *Computers & Education*, 82, 263-279.

Sysło, M. M., & Kwiatkowska, A. B. (2014). Learning Mathematics supported by computational thinking, In: Futschek, G., Kynigos, C. (eds.) *Constructionism and Creativity*, pp. 258–268. Österreichische Computer Gesellschaft, Vienna.

Snodgrass, M. R., Israel, M. & Reese, G. (2016). Instructional supports for students with disabilities in K-5 computing: Findings from a cross-case analysis. *Computers & Education*. 100, 1-17.

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

表 1 校園地圖 PAPAGO 教學概念對應

活動目標	1. 讓學生能夠讀懂地圖 2. 讓學生能具備基本空間概念，包括方位、距離
運算思維概念	物件、序列、條件、模式辨認
數學概念	1. 能辨識日常經驗「向右轉」視為順時針轉 90 度，「向左轉」視為逆時針 90 度。 2. 能辨識日常經驗「向後轉」視為轉了 180 度的平角。 3. 能從平面圖形的放大或縮小，分辨任兩點之間的長度距離也以相同的比例放大或縮小。 4. 能理解用不同個別單位測量同一長度時，其數值不同，並能說明原因

表 2 運算思維教學原則範例(Israel et al., 2015)

明確教學法的教學元素	教學說明
Focus instruction on critical content	決定要教導運算思維的哪一個能力（例如：演算法、模式辨識）。
Sequence skills logically	針對問題的解題歷程，逐一運用運算思維概念與行為模式進行教學。
Provide step-by-step demonstrations	根據本研究開發的不插電運算思維教學活動提供每一步驟的教學演示或是範例，例如教導學生如何找出遊戲的規律。
Provide immediate affirmative and corrective feedback	當學生實際進行數學解題結果時結果並不如預期或是錯誤時，老師可以藉由引導式提問教導學生正確的解題方式。

表 3 校園地圖 PAPAGO 教學活動範例

活動流程	教材或教具	時間	運算思維概念
<p>*辨認東西南北、方位、地圖</p> <p>帶領小朋友辨認方位，並教導學生辨認地圖上的建築物</p> <p>*實際演練</p> <p>老師：等一下我們要來玩一個遊戲，你要按照老師給你的指令來進行動作喔，請用紙牌排列出你要走的路徑。</p> <p>任務 1. 校門口在地圖上的右下角，現在你面對北方，請直走 3 步，再往左 2 步，請問你會到達哪裡？</p> <p>任務 2. 那現在要請你動動腦，設計一條和剛剛不同的路線回來起點</p>	<p>地圖(以巧拼製作，不同顏色代表不同建築物，白色代表道路)</p> <p>學習單</p>	30 分鐘	<p>物件</p> <p>序列</p> <p>條件</p>

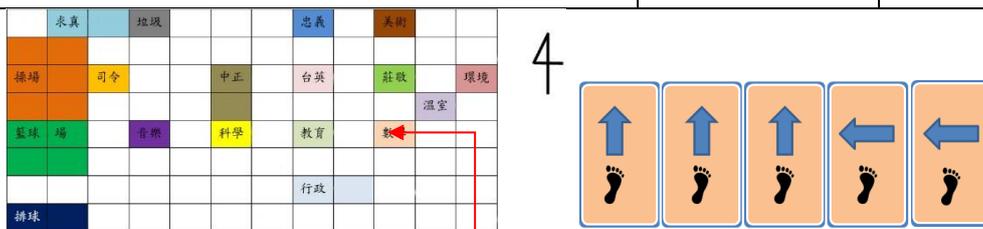


圖 1、地圖範例與排列路徑之紙牌卡

Computational Thinking and Evaluation

Evaluating Computational Thinking in Jupyter Notebook Data Science Projects

Clara SORENSEN, Eni MUSTAFARAJ*
Wellesley College, The United States
csorensen@wellesley.edu, emustafaraj@wellesley.edu

ABSTRACT

The interdisciplinary field of data science requires a strong foundation in computational thinking (CT) concepts and practices. In this paper, we describe the use of qualitative and quantitative methods to study data science projects completed by undergraduate students who are learning data science but have already learned computer programming. The projects are stored as Jupyter notebooks: documents that store code, as well as its output from execution, formatted text for self-explanations, and graphics. Our analysis of the notebooks discovers two kinds of student attitudes: explorers, who work iteratively, and goal accomplishes, who work incrementally. Despite varying attitudes, we find that students often fluctuate between the two learner types depending on their computational goals for a given notebook. Moreover, when students practice the explorer approach, they often engage more actively with many CT skills such as pattern generalization and communication of results. Finally, we propose ways to utilize these findings to encourage CT practices in future data science curricula.

KEYWORDS

computational thinking, data science education, Jupyter notebooks, learner types

1. INTRODUCTION

As we are still debating the theoretical and operational definition of computational thinking (CT), it is worthwhile to engage in the question: how can we observe, describe, and quantify its expression in learners, if at all possible? Such an exercise has the potential to shed light into the kind of thought processes in which learners already engage or are trying to master as they go about solving problems through computation. Given that learners of different ages and at different stages in their learning will display different levels of understanding, we need to study a variety of situations and groups of learners engaged in computational thinking to arrive at a more complete picture.

Our focus in this paper is undergraduate students who have already completed at least two courses in computer science: an introductory course in Python to learn computational concepts such as sequencing, loops, and conditionals, and a second course about data structures in Java, which provides opportunities to engage in CT practices such as program design, testing and debugging, and accessing and writing documentation. After completing these two courses, the students enrolled in an introductory data science course taught in Python that utilized Jupyter notebooks¹ as its environment for learning and practicing data science.

Data science, with its current focus on large amounts of automatically captured data, provides a rich context for observing CT in practice because it offers a wide range of

problems that are new and challenging, but also meaningful to explore—something that motivates learners. Concretely, in the examples analyzed for this paper, the students were able to work with a variety of real-world datasets ranging from their personal email inbox to the web server entry logs for the courses offered in our computer science department. Finding answers to questions about these datasets was not trivial. To be more efficient, students had to learn new data structures and new operators. There was no established algorithm for coming to a solution, thus, they needed to be creative, work incrementally, and iterate often, all practices inherent to computational thinking.

In this paper, we begin by providing background on Jupyter notebooks and some previous methods for assessing computational thinking that were helpful in framing our work. We then continue to discuss the data and methodology for our notebook analyses and the variables that we created from the raw notebooks. We discuss our findings from the data analysis including our uncovered learner types (explorer and goal accomplisher) and conclude with a discussion of how we propose to integrate computational thinking practices in data science education in the future.

2. BACKGROUND

We shaped our work with both the nature of the Jupyter notebooks and previous methods to assess computational thinking in mind. In this section, we provide some historical background and previous research on both these topics.

2.1. Jupyter notebooks

The choice of a programming language and its integrated development environment (IDE) impacts what can be taught and how it can be taught (Pears et al., 2007), especially when teaching novice learners. The rapid adoption in recent years of block-based programming environments such as Scratch and App Inventor for teaching programming is due in part to their ability to allow learners to focus on what is important—the computational concepts—while avoiding the struggle with the syntactical details of the underlying languages (Bau et al., 2017). Similarly, teaching data science benefits from environments that allow for frequent data exploration, incremental problem solving, and easy access to previous results of analysis. For these reasons, the Jupyter notebook is a strong candidate for teaching data science at any level of the curriculum. Furthermore, Jupyter notebooks have become the environment of choice for many computational scientists (Kluyver et al., 2016) because they encourage reproducibility in science, a practice that is important to foster in students early in their data science endeavors.

The Jupyter notebook traces its roots to the IPython (interactive Python) extension for the Python programming language (Pérez and Granger, 2007). From its inception,

¹ <http://jupyter.org>

IPython was designed to augment the Python interactive shell with features that go beyond the usual read-evaluate-print loop (REPL), common in most interpreted languages. The evolution from shell interaction to the notebook (as a single document that captures all aspects of a programming session) was inspired by the existence of notebooks for teaching Mathematics in proprietary software such as Mathematica and Maple. By making the Jupyter notebook open-source, web-based, and language-agnostic (i.e., it can be used with many different programming languages in the back-end), its community of developers has created a platform with broad appeal for educators and practitioners alike. The fact that it is also used by practitioners makes it appealing to undergraduate students who prefer real-world development environments (where they learn by doing) to pedagogical ones (Oblinger, 2004).

2.2. Assessment of CT

For our purposes, we adopt the terminology presented in Brennan and Resnick (2012), who define computational thinking as a composition of computational concepts, practices and perspectives. More specifically, computational concepts refer to the concepts students engage with when they program (e.g. iteration and parallelism). Computational practices refer to the various practices students develop as they engage with the concepts (e.g. being incremental and iterative in design). And, lastly, computational perspectives refer to shifts in perspectives about the world around the student (e.g. by expressing and connecting their work).

Past research on both measuring and assessing these computational thinking skills has focused primarily on developing assessment material for pre-college programs. Brennan and Resnick's work specifically, which concentrated on young students working with Scratch, described a variety of different assessment approaches including project content analysis and artifact-based interviews. Boechler et al. (2012) took a slightly different approach in that they calculated a variety of metrics as evidence of CT skill development in Scratch applications. Specifically, they calculated the number of scripts, number of blocks, number of variables, number of child scripts, and the nesting complexity of student Scratch projects. More recently, Moreno-León et al. (2017) obtained quantitative measurements of seven different CT dimensions in Scratch projects using a static code analyzer, Dr. Scratch, in order to cluster projects based on CT complexity. In similar block-based programming environments, students' CT skills have been assessed by way of analyzing student programming actions in their log data (Grover et al., 2017). In this instance, researchers designed specific programming tasks to draw out CT skills to make for easier evaluation. Likewise, Bienkowski et al. (2015) created design patterns for major CT practices as a way to assess how learners may be applying such skills as they develop a deeper computational understanding.

In light of the specific assessment of CT, many computational thinking researchers have explicitly emphasized the importance of data and information as a core CT practice. Barr and Stephenson (2011) include data collection, data analysis, and data representation as three of their nine core concepts and capabilities of CT. Further,

communication in the sense of explaining computational results is one of the six practices found to complement the content knowledge of computer scientists by The College Board (2014). Despite the undisputed importance of these data science elements of CT, we find that their evaluation has been explored to a lesser extent than that of other CT practices (abstraction, design complexity, etc.) in the assessments described above.

3. DATA & METHODOLOGY

For this study, we focus on assessing and evaluating computational thinking in the context of data science learners. We analyze student work in the form of Jupyter notebooks from students who took an introductory data science course at Wellesley College, a female population, in either Spring 2016 or Fall 2017. In total, we analyzed 132 notebooks created by 37 students from the two times the course was offered. The notebooks ranged in focus, utilized different (but often similar) data sets, and worked through the data science process (Figure 1) in one way or another to solve a problem. Since problem solving through computation is becoming a popularized manner of completing a data science workflow, each student notebook emphasized various CT skills while working to answer a question.

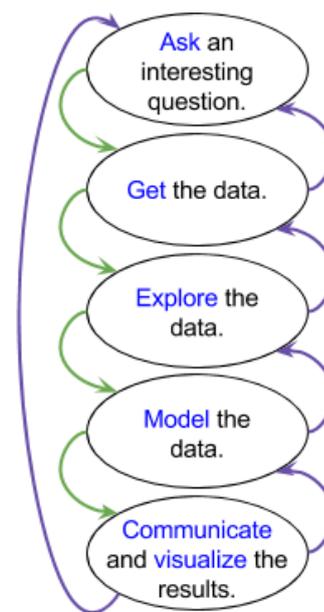


Figure 1. The data science process by Pfister and Blitzstein (2013).

Jupyter notebooks are automatically stored as JSON (JavaScript Object Notation) files, a format common on the Web. This allows for an easy analysis of the notebooks, especially to extract the input cells that contain the code entered by the students, the output cells that contain the result of the code execution, the Markdown cells (special text cells that can contain formatting such as headings, lists, emphasized text, formulas, etc.) that contain self-explanations or other useful comments. We don't manipulate the Jupyter notebooks to collect data beyond what the notebook itself stores. Moreover, we wrote a script to extract 15 different metrics that encompass computational thinking skills and computational complexity in one way or

another from each notebook in JSON format (Table 1). These metrics aim to quantify CT skills such as problem breakdown, pattern recognition, and communication. We then calculate descriptive statistics for each of these metrics in order to better understand how they vary in practice.

Table 1. Calculated metrics for a given Jupyter notebook.

Variable	Definition
numCodeCells	total number of code cells
numMkdnCells	total number of Markdown cells
numLinesOfCode	total number of lines of code
meanLinesCodeCell	mean number of lines of code in a code cell
numMkdnWords	total number of Markdown words
meanWordsMkdnCell	mean number of words in a Markdown cell
numHeaders	total number of Markdown headers
numImages	total number of images (plots, diagrams, etc.)
maxExecution	highest execution value
executionRatio	ratio of the maximum execution value to the total number of code cells
executionRange	range of the lowest execution value to highest execution value
meanExecutionPerCell	mean number of executions of a code cell
numFunctions	total number of functions created
numFunctionCalls	total number of calls to functions
meanFunctionUse	mean number of calls to a function

We further discover that from the information in the notebooks it is possible to identify two distinct types of behavior in these problem solving scenarios that we qualitatively label as "goal accomplisher" and "explorer". In brief, a goal accomplisher is a student that works incrementally toward the desired outcome, while an explorer engages in multiple iterations and sometimes off-track activities. Of all the Jupyter notebooks collected, we labeled 71 notebooks (44 "goal accomplisher" and 27 "explorer") from 19 students in the second course offering based on manual review of each notebook. Because nature of the assigned projects differed between the two course offerings, we chose to only label notebooks in the second course offering to reduce a potential course-based dependency in our metrics when comparing the learner types. We went on to plot the trajectories of each student's notebook executions in an effort to visually depict the learner type of a given notebook. We also used the labeled notebooks to determine which of our extracted CT metrics are critical in differentiating the learner types. With this, we also looked to see if certain students are prone to practicing one of these learner approaches more than the other.

4. RESULTS

4.1. Features to quantify CT behavior in notebooks

We defined 15 different variables (Table 1) in an attempt to properly assess student computational thinking ability with our JSON scraping script. After calculating these metrics for all 132 student Jupyter notebooks, we found that the distribution of values for all the features varied greatly across the notebooks (Figure 2). Based on this, we observed how some students were stronger in particular CT skills than others.

More specifically for example, the maximum number of self-declared functions in a notebook (a metric relating to both pattern recognition ability and knowledge of existing software tools) was 27 as compared to the minimum of 0 self-declared functions. Because these students are taught to utilize existing Python packages to manipulate data, we found that computationally stronger students were

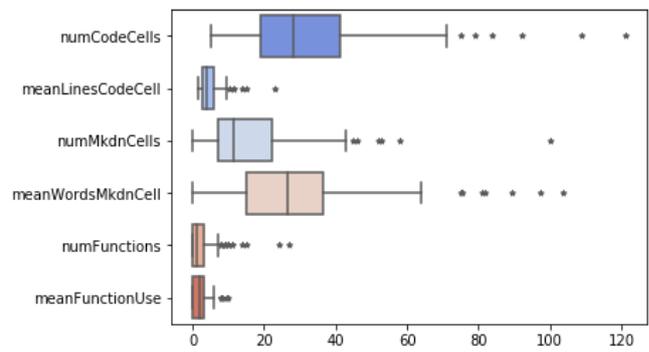


Figure 2. Boxplot depicting the distribution of selected features calculated based on all student Jupyter notebooks.

somewhere in between these two extremes. We observed that students who declare more functions often are over-declaring in the sense they aren't actively utilizing functions from other packages and they're often repetitively writing similar functions. On the other side of the spectrum, students who don't declare any functions tend to manually manipulate their data with code that is copy-pasted from their earlier code—suggesting a potential weakness in both their pattern generalization abilities and knowledge of existing software tools.

Another notable feature with a great range in values was the mean number of words in a markdown cell, a variable that links directly to the communication abilities of these data science learners. Since students were encouraged to utilize Markdown to communicate, evaluate, and explain results from their code, we found that this feature directly correlated with a student's ability to communicate their understanding with others. Specifically, students stronger on the communication front had more Markdown in their Jupyter notebooks.

4.2. Explorers vs. Goal Accomplishers

In addition to utilizing our JSON scraping script with the student Jupyter notebooks, we classified the notebooks into two groups based on behavioral trends in a student's approach to the data science cycle, trends that visually stood out in the trajectory of their notebooks.

Our "explorer" archetype consisted of students who behaved more iteratively in their approach to a given data science task. These students would often find something interesting in their analysis then go beyond—building on their conclusions by modifying their initial analysis and taking it numerous steps further. Additionally, these students were effective in their use of text explanations throughout their notebooks in order to explain and discuss both their thought processes and their computational approach to the analysis. They also provided ideas as to how they could extend their analysis and conclusions even further.

On the other hand, our "goal accomplisher" archetype featured students whose notebooks focused on answering a specific scientific question with the intention to reach a conclusion to that question and thus end their analysis. Once identifying and planning out their approach, these students would spend most time cleaning the data before going on to use this cleaner data set to answer their initial research

question. Though these students generally had a shorter data exploration period within a given notebook, they were particularly strong at identifying a major takeaway or trend from their analyses.

In our “explorer” example (Figure 3), the student tried to analyze her email behavior by creating new questions to answer throughout the course of her notebook. She started by importing the data and organizing it into a DataFrame² with some slight data cleaning and exploring. However, early on she reimported the data, presumably because she wanted to use the original data for deeper analysis (A). A bit later in the notebook she defines a function to label the day of the week an email was received to explore daily email variation (B). We see that this function is executed much earlier than surrounding cells—this is because she went back to rerun old cells but never needed to update the function itself. Then, after some initial analyzing and visualizing her email behavior on a daily basis, she went on to see trend variation on a monthly timescale and between her various social groups (C). Generally, here the student went back and forth between cells when she decided to modify her analysis as she developed new interest in the various contexts (temporal and social) of the data.

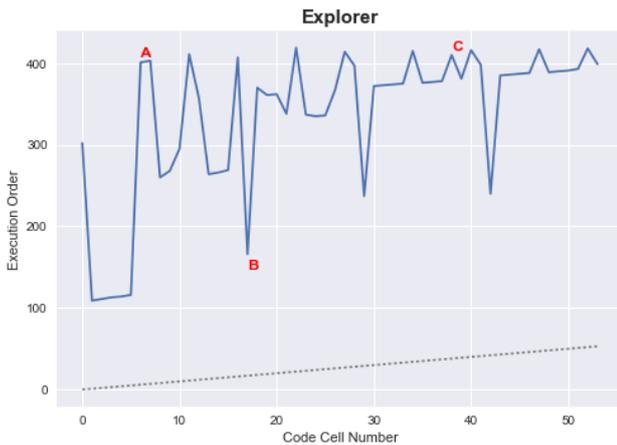


Figure 3. Example notebook of an explorer.

Our “goal accomplisher” (Figure 4) began her notebook by defining functions specific to formatting her data in a way appropriate to answer her scientific question: “who are my emails from and how does this change over time?” (A). Once she had successfully validated the cleanliness of her data, having categorized individual emails using her initially defined function, she immediately went on to analyze the categorized emails as a function of time and plotted the result (B). Continuing on, she further subsetted her categorized emails with modified functions and replotted the result once again—determining that most of her emails were “Wellesley Emails” with peaks occurring during specific points of the semester (C). In particular, throughout this notebook she worked on the same question with impeccable focus and continued to smoothly progress until she successfully found her answer.



Figure 4. Example notebook of a goal accomplisher.

4.3. Differentiating learner types

We wanted to determine whether our learner types were differentiable with our variables extracted using our JSON scraping script. After labeling 71 of the notebooks as “explorer” or “goal accomplisher” style, we ran two-sample *t*-tests to compare all our metrics between the two groups (Table 2). We found a significant difference ($\alpha = 0.05$) in the number of code cells, the number of Markdown cells, the number of lines of code, the number of Markdown words, the number of images, the maximum execution, the execution range, the mean execution per cell, the number of function calls, and the mean function use between an explorer notebook and a goal accomplisher notebook. For all these variables, explorers had a significantly higher average value than goal accomplishers.

Table 2. Results of two-sample *t*-tests comparing CT metrics between explorer and goal accomplisher notebooks.

	Explorer		Goal Accomplisher		Results	
	Mean	Std.Dev.	Mean	Std.Dev.	t-stat	p-value
numCodeCells*	49.2	26.1	24.7	15.9	4.92	<0.01
numMkdnCells*	23.1	19.1	13.7	12.6	2.49	0.015
numLines-OfCode*	245.4	151.3	132.0	118.9	3.51	<0.01
meanLines-CodeCell	5.2	2.5	5.5	3.8	-0.31	0.760
numMkdn-Words*	627.3	550.1	250.8	191.5	4.16	<0.01
meanWords-MkdnCell	29.1	19.8	21.9	15.7	1.69	0.096
numHeaders	13.8	16.2	9.2	11.5	1.40	0.166
numImages*	6.6	6.1	1.7	4.0	4.16	<0.01
maxExecution*	260.9	268.1	102.0	114.8	3.46	<0.01
executionRatio	5.3	3.7	4.3	4.3	0.94	0.351
executionRange*	244.9	252.2	72.6	80.9	4.21	<0.01
meanExecution-PerCell*	174.0	231.2	62.3	97.6	2.83	<0.01
numFunctions	4.1	5.6	2.0	4.0	1.85	0.069
numFunction-Calls*	15.9	22.8	5.0	8.9	2.86	<0.01
meanFunction-Use*	2.7	2.0	1.3	1.7	3.05	<0.01

Additionally, we wanted to evaluate these behavioral patterns across students and see if students tend to favor one learner type over the other in their notebooks. We found that most students had notebooks in both styles (Figure 5). This suggests that students work differently depending on the

² a two-dimensional labeled data structure that organizes a dataset into columns of potentially different (data) types

purpose of a given notebook. Furthermore, with this it appears that students are flexible in many of the CT practices that we quantify with extracted features from the notebooks. In particular, explorer notebooks typically featured a higher number of images (or data visualizations), a metric that directly exhibits a student’s ability and effort to visualize their data and communicate information to non-experts. This metric, however, varied greatly across notebooks for an individual student, often depending on the learner style of a particular notebook (Figure 5). This suggests that most students have already developed many of these CT skills but that they selectively apply them, naturally, in situations where they are more useful.

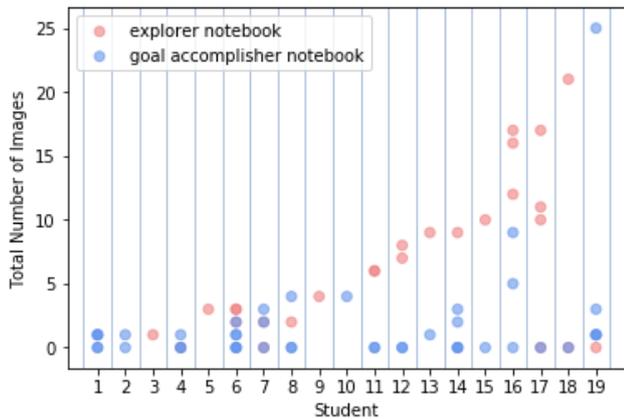


Figure 5. Number of images in a Jupyter notebook based on student and learner type.

5. DISCUSSION

In this paper, we presented our findings from both a quantitative and qualitative evaluation of student Jupyter notebook projects for an undergraduate data science course in the context of computational thinking. We developed a script to scrape the JSON-formatted notebooks for various metrics that relate to the computational ability and efforts of a student. We further found that we could classify student behavior into two groups based on behavioral trends in their notebooks, a classification that could be visually depicted by the trajectory of a student’s notebook execution. Furthermore, utilizing our extracted metrics we found that students practicing the “explorer” approach in their notebook often engaged in greater CT habits than those practicing the “goal accomplisher” approach. Seeing as explorers are more iterative by our definition and that iteration is a CT skill on its own, it appears as though many CT practices correlate and perhaps promote one another.

Though we found that the “explorer” notebooks were typically more iterative in their data science process and more thorough in their utilization of CT skills, we believe that both learner types are important to data science workflows. Since it was apparent that very few students practiced only one of the two approaches in their notebooks (68.4% of students had at least one notebook of each learner type), it makes sense to consider the learner types as flexible measures that depend on the desired goal of a project notebook. A “goal accomplisher” is not inferior to an “explorer” but rather a “goal accomplisher” at the time may be seeking something specific to glean in their notebook as

opposed to undergoing a full project that seeks to deeply uncover something new. As this is the case, however, we encourage data science instructors to promote the “explorer” approach if they’re concurrently attempting to stimulate the development and practice of CT.

It is also possible that the “explorer” approach is less natural to a young computer scientist than the “goal accomplisher” approach. When exploring a student needs to be flexible as compared to when they have a goal in mind and they generally already know how to accomplish it. This idea is similar to the “expertise effect” seen in chess: an expert chess player sees the field in terms of patterns whereas a novice player sees it as a list of the positions of all the pieces. The expertise here comes with familiarity of the data and knowledge of the tools available to work with it efficiently and effectively. Since computer science students are often used to completing assignments with concrete instructions and purpose, working in the goal accomplisher manner may be more intuitive for new data science students. With the greater discomfort that may come with the explorer approach, students may be naturally practicing already developed CT skills as they iteratively work through a problem.

Notebook behavior also likely depends on both the individual and the type of task. Though most individuals (68.4%) exhibited both learner type behaviors in their notebooks, 31.6% of students only featured one learner type in their notebooks (15.8% of students were explorer exclusive and 15.8% of students were goal accomplisher exclusive). Students whose notebooks focused more heavily on data cleaning tended to favor the goal accomplisher style. In contrast, student notebooks that were more focused on understanding the data in a variety of ways, often including some sort of modeling aspect, favored the explorer style. We noticed this trend in learner type based on notebook focus particularly in the email inbox analysis project in which student analysis was less structured and more up for student interpretation than some of the other projects.

Additionally, it is important to note that a binary classification for a given notebook may not always be appropriate. For our purposes, we felt that the notebooks we labeled fit well into one of the notebook styles based on a manual review of the student’s approach. However, based on the statistics of the measured CT metrics for each learner type, we know that there is a great range in CT expression within both styles so a binary classification may not be applicable in all cases. We believe it is important to further explore the idea that there may exist a spectrum between the two learner types observed here.

Even more, since data science requires individuals to solve complex problems with computation it also requires continuous learning. Here, we looked at the work of students new to data science but not new to computer science—they all had previously taken at least two other courses. Based on our feature extraction, it appears that some students are more flexible and willing than others to adopt new tools and practices that are taught (e.g. consider the function declaration and use metrics). This student resistance to upgrade their skills and learn new tools (when they believe they already have the tools to solve a problem) we feel can

hinder CT and instructors need to be conscious of this problem. With this, we think our metrics should be utilized in a cautionary manner for instructors to use to evaluate student flexibility and effort in learning new material.

6. CONCLUSION

Learning data science requires students to utilize a variety of computational thinking concepts and practices. Here, we developed an approach to convert Jupyter notebooks into a series of metrics that might be associated with certain CT skills. These metrics include, but aren't limited to, the number of functions created—a feature that may depict signs of pattern generalization—and the mean number of lines of code—a feature that may correlate to algorithmic efficiency. However, more research is necessary to connect all our metrics with concrete CT skills and practices. Further, we find that when these metrics are compared across various students, it becomes easier to assess how students are performing in relation to one another and perhaps helps to identify weaknesses in certain CT areas of individual students.

One advantage of our findings is a “minimum effort approach” that can be used by instructors without the need for a sophisticated research infrastructure. Jupyter has an online version, JupyterHub, which makes it easy for students to upload their work online. Over time, this should make it easy for an instructor to observe student skills by utilizing our learning analytics. However, to be successful in practice our approach relies on students following instructions about storing everything in their Jupyter notebook by considering it exactly as a personal notebook where they record everything that happens during their learning and not as a polished, final product to submit for a grade. Additionally, we will continue to develop these metrics and ideally go on to produce a type of teacher dashboard in which a teacher can see data about the progress of their students. We also aim to provide feedback and potential recommendations for what CT skills may need to be worked on from this tool.

7. REFERENCES

- 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.
- Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: blocks and beyond. *Communications of the ACM*, 60(6), 72-80.
- Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). Assessment design patterns for computational thinking practices in secondary computer science: A First Look. *SRI International* 2015.
- Boechler, P., Artym, C., Dejong, E., Carbonaro, M., & Stroulia, E. (2014, July). Computational Thinking, Code Complexity, and Prior Experience in a Videogame-Building Assignment. In *Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on* (pp. 396-398). IEEE.
- Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).
- Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A Framework for Using Hypothesis-Driven Approaches to Support Data-Driven Learning Analytics in Measuring Computational Thinking in Block-Based Programming Environments. *ACM Transactions on Computing Education (TOCE)*, 17(3), 14.
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., ... & Ivanov, P. (2016, May). Jupyter Notebooks—a publishing format for reproducible computational workflows. In *ELPUB* (pp. 87-90).
- Moreno-León, J., Robles, G., & Román-González, M. (2017). Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing*.
- Oblinger, D. (2004). The next generation of educational engagement. *Journal of interactive media in education*, 2004(1).
- Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., ... & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, 39(4), 204-223.
- Pérez, F., & Granger, B. E. (2007). IPython: a system for interactive scientific computing. *Computing in Science & Engineering*, 9(3).
- Pfister, H., & Blitzstein, J. (2013). Course: CS109 Data Science.
- The College Board. (2014, June). AP Computer Science Principles Draft Curriculum Framework. Retrieved from <http://media.collegeboard.com/digitalServices/pdf/ap/comp-sci-principles-draft-cf-final.pdf>

Assessment of Computational Thinking

Nikolina BUBICA¹, Ivica BOLJAT^{2*}

¹Mokosica Middle School Dubrovnik, Croatia

²Faculty of Natural Science Split, Croatia
nikolina.bubica@skole.hr, boljat@pmfst.hr

ABSTRACT

With the new CS Curriculum in the Republic of Croatia, Computational thinking (CT) has finally been introduced in the educational process. In addition to the benefits that CT concepts bring to CS education, the question of evaluating CT and programming learning outcomes is also opening. The purpose of this paper is to present a model of evaluation of CT concepts based on the learning outcomes of the Croatian CS Curriculum using the Evidence-center design approach. The model is independent of the programming tool or environment and is intended for use with students who are CS novices.

KEYWORDS

Computational thinking, evaluation, programming novices, evidence-center design.

1. INTRODUCTION

New trends in technology development have a great impact on our daily lives. Technology enters the fabric of our lives regardless of the occupation area, but also regardless of the age of the user. We hear more demands for changes in K12 education. Also, regardless of the type of technology students use and the occupations they are being educated for, they are increasingly expected to possess some generic competencies such as ability to solve problems in everyday life, disaggregate complex problems to simpler ones, generalize solutions, etc. The fundamental question today is how to respond to such challenges. Leaders of CS education increasingly emphasize the need to modify existing CS curricula and to include the development of these competencies. Jeannette Wing (Wing J. M., 2006) points out that besides the standard types of literacy, such as mathematical, engineering and reading literacy, students are expected to have the ability to solve problems. She defines CT as "...the process of formulating problems and their solutions, but in ways that solutions are presented in a form that enables them to perform effectively with some information processing agent " (Wing J. M., 2010).

2. COMPUTATIONAL THINKING

There is still a lot of confusion over the very definition of the concept of computational thinking, and many surrounding questions and challenges need to be addressed. It is considered to be the universal competence of every child that would, together with analytical skills, be the foundation for each child's school learning (Wing J. M., 2006). Denning (Denning P. J., 2009) discusses whether CT belongs exclusively to the field of CS. Guzdial (Guzdial, 2008) describes CT like a 21st century literacy that is necessary to a whole series of faculties. It is often discussed how CT differs from algorithmic thinking, and Denning adds that "... CT means interpreting the problem as an information

process for which we are then trying to find an algorithmic solution" (Denning P. , 2010). To create an operational definition of CT, the ISTE and CSTA organizations analyzed feedback from about 700 surveyed teachers, scientists and CS researchers. The result was formulated in the operational definition of CT for K12 education as a problem-solving process which includes formulating problems, logically organizing, analyzing and representing data with abstractions, automating solutions through algorithmic thinking and generalizing the problem-solving process (ISTE & CSTA, 2011). When talking about teaching and learning CT, perhaps the most interesting is the role of programming. How much programming, if any, is needed to adopt CT? There is no unique answer, but practice points to different levels of programming involvement. Some define CT as a fundamental ubiquitous problem-solving tool and suggest several activities and projects which address CT (Astrachan, Hambruch, Peckham, & Settle, 2009). Other approaches suggest various ways of incorporating programming into teaching and learning of CT, from those in which programming is the fundamental CT skill to those that integrate CT through various general education courses.

3. CT IN THE CROATIAN PROPOSAL OF CS CURRICULUM

In May 2016, Croatian Ministry of Education published Proposal of CS Curriculum for K12 education. The proposal was a promising hope for CS teachers since most of them were restrained by the old and outdated curriculums. Moreover, CS curriculum proposal finally accepted CT to be a significant part of the CS education in general. Croatian curriculum subject domains are e-Society, Digital literacy and Communication, Information and Digital technology and CT and Programming (Brođanac, et al., 2016). The role of CT and programming domain in CS Curriculum aims to make students to be involved in logical thinking, modeling, abstracting and problem-solving because solid ICT education, based on CT and creativity, should enable understanding and alteration of the world around us (Brođanac, et al., 2016). CT learning outcomes are created from the beginning of primary education starting with elementary pupils, age 6-7, through middle school pupils, age 11-14, and finally high school students, age 15-18 (Brođanac, et al., 2016)

4. HOW TO ASSESS CT?

Everyone agrees that learning programming is hard, but it seems that evaluating new knowledge through evaluating new definitions and programming commands is far simpler than evaluating the way students apply computing and programming language to solve problems and to design different computer work. To assess CT, it is necessary to find evidence of a deeper understanding of the problem

solved by the student or to find evidence of understanding how the student created his coded solution. Since CT concepts include, for example, abstraction (ISTE & CSTA, 2011), it means that we must find ways to evaluate how student applied abstraction in his solution while trying to solve a problem. As there is very little agreement about the CT definition, it is even less known about the tools for assessing such thinking. However, there are approaches for evaluating the development of CT that are currently in use or are still in development. They could serve as a solid foundation for developing a general approach for evaluating CT. Brennan and Resnick propose a valuation method that includes project portfolio analysis, document-based interviews, and development of design scenarios (Brennan & Resnick, 2012). Such approach estimates the fluidity of computer-based practice of testing and debugging, experimenting and repetition, abstraction and modulation, and reusing and remixing/scaling. Expertise is assessed through three levels: low, medium and high. The evaluation approach of student's documentation consists of building creative projects from students but also of creating visible traces of their work on the project. Such traces could be achieved in the form of paper or digital diaries. Also, it could be achieved by using Scratch's commentary capabilities for explaining some project's features and screen views that will graphically present the project, its intent or the main advantages and disadvantages. Still, there is not enough research data to validate this approach. Dorling and Walker specifically study the practice of teaching CT in the classroom environment and propose a framework for evaluating the Computing Progression Pathway that recognizes the major areas of CS and offers specific levels of adoption (Dorling, 2014). Within the PACT project (Principled Assessment of CT) general CS practice is represented through some design patterns which emphasize application and reviewing of design skill while solving the computational problem rather than evaluating the knowledge of the concepts necessary to apply such skills (Bienkowski, Snow, Rutstein, & Grover, 2015). This approach is based on Evidence-centered design (ECD) (Hendrickson, Ewing, Kaliski, & Huff, April, 2013) for creating a structured description of the domain evidence argument and highlights knowledge and skills complexity or other features or behaviors that should be valued. The ECD approach is usually represented through five layers: domain analysis, domain modeling, conceptual evaluation framework, evaluation application and delivery. SRI Education group, within the PACT project, proposed application modes for every layer to create the practice of CT assessment. Also, it is possible to find several published computer-based or paper-pencil tests that differ in context, intended for the age of those who are important in testing and reevaluating (Werner, Denner, & Campe, 2012). This paper offers a framework for assessing CT demonstrated on Croatian Learning Outcomes of CT and Programming Domain based on ECD and PACT evaluation proposal.

5. PROPOSAL OF CT ASSESSMENT

Despite the advantages of introducing CT into the new curriculum, we can't ignore possible difficulties and new problems that arise from this new approach to teaching CS. Evaluation of CT becomes a new challenge in the present

CS educational work and requires a more serious approach than finding individual solutions by teachers' practitioners. One proposal of CT evaluation will be presented in the next paragraphs. It uses ECD as an orientation towards multiple activities necessary to create useful documentation like domain analysis, domain modeling, construction of framework and assessment implementation and delivery (Mislevy & Harertel, 2006).

5.1. Domain analysis

Appropriate pedagogical practice, emphasizing the constructivist approach to learning and putting students at the heart of the learning process, should develop the competencies like independence, self-confidence, responsibility, and entrepreneurship. CS curriculum created according to the learning outcomes instead to the prescribed content, enables the realization of learning and teaching directed at each student level and the development of his or her potential. It provides flexibility and gives freedom to the teachers in designing the learning and teaching process. The basic goal of the domain analysis layer is to find and explore all relevant materials concerning the target learning outcomes. In this article, we will use the sixth-grade CT learning outcomes, student age 11-12 (<http://bit.ly/2018cte>, Table 1). These learning outcomes stem from several documents but mostly Croatian National Educational Standards, CS Teacher Standards and Proposal of Croatian CS Curriculum. Croatian National Educational Standards defines the way in which CS is involved in Croatian primary, secondary and higher education. Croatian CS Curriculum and CS Teacher Standards defines CS learning outcomes at each educational level with its adoption level specification. Every learning outcome is expressed in detail within Bloom taxonomy, through different adoption levels: satisfactory, good, very good and exceptional level (<http://bit.ly/2018cte>, Table 2). These learning outcomes are a basis for our assessment process. In following sections, we will try to identify more design patterns that will help us create appropriate evaluation.

5.2. Domain modeling

Domain modeling has the task to identify elements for describing the domain we want to evaluate. According to ECD approach, Domain modeling is organized into five categories: fundamental and additional knowledge, skills and features, possible working products, variable feature and possible observations (Bienkowski, Snow, Rutstein, & Grover, 2015). An example of domain modeling for CT sixth-grade learning outcomes can be found on author's personal page (<http://bit.ly/2018cte>, Table 3).

5.3. Assessment framework

CT evaluation is highly dependent on the context within which the evaluation is performed. Is it necessary to conduct CT assessment using some programming tool or environment? The question of the connection between CT and programming must be defined regarding the context of the applied evaluation. There are different approaches to incorporating programming into the process of teaching and thus the process of CT assessment. We differentiate them according to the role of programming and CT in the course curriculum (Astrachan, Hambruch, Peckham, & Settle, 2009). In this paper, assessment of CT is achieved through

the approach that is not dependent on the programming tool or environment. This approach could serve for evaluation of adopted learning outcomes in real classroom situations at some stage of education. Precisely, the independence of the programming tool or environment enables wider application of the evaluation tool and highlights the concepts of evaluation rather than the syntax of a programming tool or environment possibilities. For the same reason, such a tool could be used with students that have no programming background. According to ECD (Hendrickson, Ewing, Kaliski, & Huff, April, 2013), evaluation framework aims to assist assessment designers while they validate their task model. Every assessment designer should validate his work with questions regarding construct relevance, specificity, and scalability and questions related to item statistics and item complexity. This evaluation framework should provide information about evidence, students model and task model, observable characteristics, measurement models and test specifications. For testing this model of evaluation, a similar measuring instrument adapted to Python programming language was conducted during 2016/2017 school year. Evaluation instrument was applied after 12 weeks (6th grade) or 14 weeks (7th grade) of learning and teaching process on a sample of 15 students of 6th grade (8 female) or 10 students of 7th grade (3 female). The positive and promising results of probe evaluation encouraged the creation of this evaluation model, independent of the programming tool and the programming environment.

Model of students

Given that the evaluation is intended for use in middle and secondary schools in the Republic of Croatia where there is a big diversity in applied programming tools and languages, an evaluation that is not dependent on the programming tool could be widely applicable. Programming tool or environment independence emphasizes on CT concepts rather than the ability to work with specific tool or environment. Also, if it is crucial for the actual CS curriculum to use certain programming tool or an environment, these tasks could be easily customized and constructed in it.

Model of tasks

Evaluation tasks are created for students with little or no programming knowledge. Each represents one puzzle used to help the main character in solving problems. Puzzles are supposed to assess one or more CT concepts. CT concepts, concealed in puzzles, have been selected and aligned with the expected learning outcomes (Brođanac, et al., 2016) and detailed domain analysis (<http://bit.ly/2018cte>, Fig. 2). Assessment tool should be implemented in the form of online knowledge test consisted of 10 questions. The types of questions that will appear in the evaluation tool are: multiple choice questions (mostly used for identification of some fundamental misconceptions or unsustainable mental model), short answer questions; essay questions (used for student's authentic algorithmic solutions). Feedback for multiple choice questions should be defined automatically, while short answer questions and essay questions should be manually evaluated by the researcher or teacher.

Model of evidence

Design and application of high-quality assessment are very demanding and also time-consuming. According to ECD approach (Hendrickson, Ewing, Kaliski, & Huff, April, 2013) our assumptions and hypothesis represent evidence about the way student's abilities are represented in his work. Such evidence should reveal student's adoption of learning outcomes. Each algorithm solution is always difficult to evaluate automatically. Evidence analysis helps us in creation of evidence model for similar tasks. While analyzing possible student's answers, it is crucial to know which computational concepts are evaluated with the default task (<http://bit.ly/2018cte>, Table 4). Evidence of student work varies from the situation where the student doesn't even try to do anything, further through several partial solutions and finally to a fully correct solution (<http://bit.ly/2018cte>, Table 5).



Task 8. Dangerous frogs appeared on different places in the labyrinth. Frog wants to stop Maja on her way to the yellow flower. So, in order to help Maja we will allow her to jump over the frog whenever she encounter one while going up. We apply the new rule: if Maja encounters the frog on her way up, she may jump over it by doing two steps at once. Write your own commands in the form of new Action `go_up_jumpover_frog` for Maja moving up and jumping over the frog.

```

Action Labyrinth:
set Step_number to nul
while not (flower_up OR flower_right) repeat:
  walk

Action walk:
while NOT obstacle_up:
  go_up_jumpover_frog ↑
  increase Step_number by 1
if NOT obstacle_right:
  go 1 step right →
  Increase Step_number by 1
else |
  if NOT obstacle_left:
    go 1 step left ←
    Increase Step_number by 1

Action go_up_jumpover_frog:
?????

```

Figure 1: Example of essay task question

Model of measurement

To complete domain analysis and modeling, it is necessary to define the model of measurement. For the task example in Fig. 1, the possible evidence is presented according to its complexity. If the student does not offer any response or his answer has no links to the task itself, such answer should be rewarded with zero points. With each of the following evidence, it has been recognized a higher level of adoption from the previous one (<http://bit.ly/2018cte>, Table 5).

5.4. Assessment implementation/delivery

The realization of the test assessment, adopted for Python programming tool and conducted during the 2016/2017 school year, was performed as online assessment within Loomen Learning Management System (LMS). The assessment consisted of eight tasks (one pairing task, four multiple choice tasks, three essay tasks) and was conducted during a 45-minute regular school class. The students showed great satisfaction by conducting online assessment instead of standard paper-pen assessment even though it was their first real encounter with such form of evaluation. The assessment task discrimination analysis showed that as many as six tasks proved to be excellent (task discrimination

index > 0.35) while two of them were discarded from further modeling due to the negative index of discrimination. As for the task difficulty, two of them have proved to be extremely simple (0.93), but they have already been dismissed from the further modeling because of their extremely low discrimination. Two tasks had recommended difficulty (0.5-0.6) and four task acceptable difficulty index (0.3-0.7). Further application of the assessment tool will be used to test the validity and reliability of the measuring instrument and will help in creating this new CT assessment model. CT assessment model proposed in this paper will also be organized in the form of online testing within the Loomen LMS. Students' access to Loomen LMS will be enabled through their unique user data, provided to every middle and secondary student in the Republic of Croatia. In that way, the authenticity of the research participants' data will be preserved. In the phase of pilot research, it is expected the involvement of 50-60 students with the purpose of testing the clarity of task texts and detecting potential ambiguities or some other problems. Also, several CS teachers will be invited to evaluate the assessment tool as valuable practitioners with attention on measurement model. After defining the final version, the assessment tool will be applied to as many 11-12 year old students as possible who are just encountering fundamental concepts of CS. In addition to the evaluation tool, the students will be previously asked to fill out a questionnaire that aims to collect some personal information interesting to the research like general data such as gender, general academic achievement or some data related to programming knowledge. Also, evaluation tool will be applied even with some number of high school students. The results of the research should reveal the power of the tool itself, but also could explore if there is a difference in the results among participants who have some programming experience from those who have none, and further investigate whether there are differences in gender related to results and so on.

6. CONCLUSION

Many teachers are increasingly emphasizing the need for a stronger involvement of the CT concepts in CS courses, but it is also noticed within some other sciences such as biology, physics, mathematics, chemistry (Interdisciplinary Computational Thinking, 2017). The purpose of this paper was to present one approach to the assessing CT adapted to the actual classroom situation. The proposed assessment tool was developed knowing that there are several programming tools and environments used in CS education in the Republic of Croatia, but also accepting the fact that CS is an elective subject in elementary/middle schools where programming is only minor part of the subject curriculum. The new CS curriculum proposal introduces the concept of computational thinking, and thus opens the question of evaluating its concepts. The proposed evaluation model is based on defined learning outcomes from the CT and programming domain of the new CS curriculum proposal and offers the possibility of assessing CT concepts independently of applied programming tools and

environments in the teaching process. Also, it could serve as the basis for making similar assessment tools. The real power of the tool, its validity, and reliability, but also its weaknesses will be able to reveal through its application, which is our next step.

7. REFERENCE

- Astrachan, O., Hambrusch, S., Peckham, J., & Settle, A. (2009). The present and the Future of Computational Thinking. *ACM 978-1-60558-183-5/09/03*, pp. 549-550. Chattanooga, Tennessee, USA.
- Bienkowski, M., Snow, E., Rutstein, D., & Grover, S. (2015). *Assessment Design Patterns for Computational Thinking Practices in Secondary Computer Science: a first look*. Menlo Park, CA: SRI Education.
- Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*.
- Brođanac, P., Bubica, N., Kralj, L., Markučić, Z., Mirković, M., Rubić, M., & Sudarević, D. (2016, February). *Computer Science National Curriculum - proposal*. Retrieved from kurikulum.hr: <http://www.kurikulum.hr/wp-content/uploads/2016/03/Informatika.pdf>
- Denning, P. (2010). The Great Principles of Computing. *American Scientist*, vol. 98, str. 369-372.
- Denning, P. J. (2009). The Profession of IT Beyond Computational Thinking. *Communications of the ACM*, vol. 52, no. 6, pp. 28-30.
- Dorling, M. &. (2014). *Computing Progression Pathways*. Retrieved from <http://community.computingatschool.org.uk/resources/1692>
- Guzdial, M. (2008, August). Paving the way for the Computational Thinking. *Communications of the ACM*, vol. 51, no. 8, pp. 25-27.
- Hendrickson, A., Ewing, M., Kaliski, P., & Huff, K. (April, 2013). Evidence - Centered Design: Recommendations for Implementation and Practice. *Journal of Applied Testing Technology, JATT, volume 14, Association of Test Publishers*.
- Interdisciplinary Computational Thinking*. (2017, July 11). Retrieved from Teaching London Computing: A RESOURCE HUB from CAS LONDON: <https://teachinglondoncomputing.org/interdisciplinary-computational-thinking/>
- ISTE, & CSTA. (2011). *CSTeachers*. Retrieved from Computational Thinking resources: <https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CompThinkingFlyer.pdf>
- Mislevy, R. J., & Hartertel, G. (2006). Implications for evidence-centered design for educational assessment. *Educational Measurement: Issues and Practice*, 25, 6-20.
- Werner, L., Denner, J., & Campe, S. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle School. *SIGCSE'12*. Raleigh, North Carolina, USA: Copyright 2012 ACM.
- Wing, J. M. (2006, March). Computational thinking. *Communication of the ACM*, . 49 vol., no.3, pp. 33-35.
- Wing, J. M. (2010, November 17). Retrieved from www.cs.cmu.edu: <https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>

Cross Comparison of Multiple Computational Thinking Activities:

a Grey-based approach

Meng-leong HOW, Chee-kit LOOI*

National Institute of Education, Nanyang Technological University, Singapore
mengleong.how@nie.edu.sg, cheekit.looi@nie.edu.sg

ABSTRACT

The current paper proposes a grey-based approach to conduct cross comparison analysis of multiple Computational Thinking (CT) activities, which could be used to better inform decision makers and policy makers in education about the exact CT activities which they might like to consider selecting for the learners; regardless of whether these CT activities are screen-based block-based programming (such as Blockly or Scratch), or text-based programming (such as using the Python, or Java, or C# programming language, et cetera), or unplugged CT activities, or physical computing activities (such as programmable robots, or circuit boards with microcontrollers such as Arduino and the BBC Microbit). Further, this grey-based cross comparison approach can be used regardless of the rubric or test being used to assess each individual CT activity (for example, CT-Profile, PECT, PACT, Dr Scratch, CTt psychometric test, ACTMA, CT-Stem, or Bloom's Taxonomy, or SOLO Taxonomy). Potentially, this grey-based approach of cross comparing multiple CT activities could be useful for anyone who is interested in pulling together all of the analyses for different CT activities into one coherent meta-analysis of multiple CT activities.

KEYWORDS

Computational Thinking, evaluation, multiple comparisons, Grey-based approach, assessment

1. INTRODUCTION

1.1 Computational Thinking

Griffin (2016) points out that it is important for novice programmers to develop a mental model of a notional machine (du Boulay, O'Shea, & Monk, 1981), which is a rudimentary model that describes the instructions of a computer program. Strong interest in how the novice programmer could develop this mental model have more precisely elucidated this mental model of a notional machine into what is now known as Computational Thinking (CT) (Wing, 2008). The constituents of this mental model of CT include decomposition, algorithmic thinking, abstraction of data, abstract of functionality, evaluation, and generalization. Indeed, CT is indispensable to problem-solving in the real world, and is considered to be essential in education (Wing, 2008). According to Gouws et al. (2013), decomposition refers to the process of breaking down a problem into multiple steps in order to solve it. Algorithmic thinking refers to the repetitive execution of patterns of instructions, which might involve loops for iteration or recursion. Abstraction of data and functionality refers to the notion of representations in data storage and the

manipulation of those data in functions. Generalization refers to the ability to create adaptable solutions that are reusable for a wider range of problems. Evaluation is the ability to select the best solution for a given problem, as well as to identify and correct errors.

The following is an overview of various CT assessments that are useful for assessing the suitability of individual CT activities for learners, prior to doing a cross comparison of multiple CT activities using the proposed grey-based approach in the current paper.

1.2 CT Assessments of screen-based CT activities

Screen-based CT activities involve block-based programming using drag-and-drop graphical elements. Examples of block-based programming include Scratch, Alice, and AgentSheets. A seminal assessment framework for block-based programming is the Systems of Assessments for Deeper Learning of Computational Thinking for K-12 by Grover (2015).

1.3 CT Assessments of Unplugged CT activities

Unplugged CT activities teach computing concepts without screen-based devices. They include those offered by CS Unplugged (Bell, Alexander, Freeman, & Grimley, 2009), Code.org, and CAS London. Assessments for unplugged CT activities have been propounded by Rodriguez (2015) and also by Takaoka, Fukushima, Hirose, and Hasegawa (2014).

1.4 CT Assessments of Physical Computing activities

Examples of physical computing in education include Arduino, Raspberry Pi, and the BBC Microbit. Assessments for computational thinking in physical computing-based activities include (ACTMA) Assessing Computational Thinking in Maker Activities, and the CT-Stem taxonomy (Weintrop et al., 2014).

1.5 Research Problem

Although there is myriad of CT assessments, almost nothing exists in the extant literature which looks at systematically performing comparisons in a transparent way across multiple CT activities, which could be used to inform educators and policy makers about the developmental level of CT skills involved in each activity, thus enabling them to select those activities that might best fit the learners' CT skills development needs.

In assessments, there are usually four types of measurement scales – nominal, ordinal, interval and ratio (Anderson, 1961). A nominal scale assigns numbers that can be utilized to categorize items. For example, a CT activity might be assessed according to whether it is a screen-based activity, or an unplugged activity, or a physical computing activity. It

does not compare whether one category is superior to another, and vice-versa.

An ordinal scale uses variables of increasing or decreasing values to provide meaningful information for comparing categories of items. For example, a CT activity might be assessed according to whether it is low-level, medium-level, or high-level in terms of difficulty.

An interval scale provides precise information on the rank order of the item being measured, with equidistant “spacing”, however the interval scale does not have an absolute zero point. For example, a CT activity might be rated on its age-appropriateness by assessors, which normally does not include the point of birth (age at absolute zero number of years).

The ratio scale provides the most amount of information; not only is it equidistant, it also has an absolute zero point. Examples that utilize the ratio scale might include the length of time that a CT activity takes, or the amount of money that a CT activity costs.

Hence, it can be challenging to compare multiple CT activities. “Poor information” or “incompleteness of information” is likely due to a lack of consensus when comparing multiple CT activities, each of which might have utilized a different measurement scale, or even multiple measurement scales. Incompleteness in information is the fundamental meaning of being “grey” (Deng, 1989), which is also what makes comparison of multiple CT activities challenging. Therefore, we proffer that a grey-based approach is particularly suitable for comparing multiple CT activities.

In the present paper, we propose a grey-based approach of cross comparing multiple CT activities. The rest of the paper is organized as follows: in Section 2, Grey Theory (Deng, 1989) will be briefly discussed with a more specific focus on grey-based (MADM) Multiple Attribute Decision Making (Li, Yamaguchi, & Nagai, 2007), which forms the foundation upon which this proposed method of a grey-based approach to conduct cross comparison analysis of multiple CT activities is built on. In Section 3, a worked example will be used to apply the proposed grey-based cross comparison approach to a set of hypothetical data from six CT assessments. Finally, the implications for education of this proposed cross comparison of multiple CT activities will be discussed.

2. GREY-BASED APPROACH

Following Liu and Lin (2010, p. 15), we use the conceptual notion of “black” to represent completely unknown information, “white” to represent completely known information, and “grey” to represent partially known and partially unknown information. A grey number is defined as a number with uncertain information and is denoted as $\otimes G$ (Deng, 1989; Liu & Lin, 2010; Liu, Yang, & Forrest, 2016). A grey-based approach of performing cross comparison of multiple CT activities is proposed in this paper, because it excels in comparing multiple entities in situations where there might be a diversity of characteristics in the various entities, uncertainty, scarcity of quantitative data, or incomplete information; situations which educators or

decision makers might find themselves in when evaluating different CT activities offered by different people for their learners.

Using a grey-based approach, ratings of CT attributes described by qualitative linguistic variables from different CT Assessments can also be expressed in grey numbers (see Table 1), after consensus has been reached by the decision makers. To illustrate the point that the grey intervals agreed upon by the decision makers do not even have to be strictly equidistant, Advanced (A) has a slightly wider grey interval compared to the rest of the developmental levels of CT skills in this suggested example of a grey interval table. This proposed grey-based approach is not a rigid framework. It is intended to be flexibly adapted by the CT evaluators.

To ensure fairness in the assessments, each of the decision makers would be independently assessing the CT activities “blind”; unaware of what ratings the other assessors might give. There would be no need to address how agreement or disagreement between the assessors was handled in the procedure. Hence, interrater reliability calculations between the assessors would be unnecessary.

Table 1: Scale of CT skills attribute ratings using intervals of grey number $\otimes G$

Scale (level of CT skill)	Intervals of grey number $\otimes G$
Very Rudimentary (VR)	[0, 1]
Rudimentary (R)	[1, 3]
Rudimentary-Intermediate (RI)	[3, 4]
Intermediate (I)	[4, 5]
Intermediate-Advanced (IA)	[5, 6]
Advanced (A)	[6, 9]
Very Advanced (VA)	[9, 10]

3. APPLICATION AND ANALYSIS

A grey-based approach for the comparison of multiple CT activities, which could include but are not limited to activities that are screen-based, unplugged or physical computing, is proposed as follows: in this worked example (see Table 2), let us suppose that there are six CT activities S_i ($i = 1, 2, \dots, 6$) selected for comparison against five CT skills attributes Q_j ($j = 1, 2, \dots, 5$). The CT skills attribute Q_1 represents Abstraction, Q_2 represents Algorithmic Thinking, Q_3 represents Decomposition, Q_4 represents Generalization, and Q_5 represents Evaluation respectively.

Table 2: Attribute rating values for Computational Thinking Activities

Q _i CT Skill	S _i CT Activity	D ₁ Decision Maker 1	D ₂ Decision Maker 2	D ₃ Decision Maker 3	D ₄ Decision Maker 4	⊗G _{ij} Grey interval
Q ₁ Abstraction	CT Activity 1	A	IA	A	A	[5.75, 8.25]
	CT Activity 2	IA	A	A	IA	[5.00, 6.50]
	CT Activity 3	I	I	IA	A	[4.75, 6.25]
	CT Activity 4	I	IA	IA	I	[4.50, 5.50]
	CT Activity 5	IA	I	I	IA	[4.50, 5.50]
	CT Activity 6	A	IA	IA	IA	[5.25, 6.75]
Q ₂ Algorithmic Thinking	CT Activity 1	A	A	IA	IA	[5.50, 7.50]
	CT Activity 2	A	IA	IA	A	[5.50, 7.50]
	CT Activity 3	I	I	R	I	[3.25, 4.50]
	CT Activity 4	R	RI	RI	R	[2.00, 3.50]
	CT Activity 5	RI	RI	R	RI	[2.50, 3.75]
	CT Activity 6	RI	R	R	RI	[2.00, 3.50]
Q ₃ Decomposition	CT Activity 1	A	IA	IA	A	[5.75, 8.25]
	CT Activity 2	IA	A	A	A	[5.75, 8.25]
	CT Activity 3	A	A	I	IA	[5.25, 7.25]
	CT Activity 4	A	IA	IA	A	[5.50, 7.50]
	CT Activity 5	IA	I	I	IA	[4.50, 5.50]
	CT Activity 6	I	I	IA	I	[4.25, 5.25]
Q ₄ Generalisation	CT Activity 1	I	A	A	A	[5.50, 8.00]
	CT Activity 2	A	A	I	IA	[5.75, 8.25]
	CT Activity 3	VA	VA	A	A	[7.50, 9.50]
	CT Activity 4	A	IA	A	A	[5.75, 8.25]
	CT Activity 5	IA	IA	A	IA	[5.25, 6.75]
	CT Activity 6	A	VA	VA	A	[7.50, 9.50]
Q ₅ Evaluation	CT Activity 1	A	A	IA	A	[5.75, 8.25]
	CT Activity 2	IA	A	A	IA	[5.00, 6.50]
	CT Activity 3	A	IA	I	I	[4.75, 6.25]
	CT Activity 4	I	IA	IA	I	[4.50, 5.50]
	CT Activity 5	IA	I	I	IA	[4.50, 5.50]
	CT Activity 6	IA	IA	IA	A	[5.25, 6.75]

A committee of four CT activities assessors, who can also be referred to as Decision Makers D1, D2, D3 and D4 has been formed to express their preferences of CT activities for the learners. Examples of CT assessors or decision makers could include, for example, teachers, heads of departments, school principals, or researchers.

Step 1

The equation for calculating the average of the lower and upper bounds of the grey intervals respectively is:

$$\otimes G_{ij} = \frac{1}{K} [\otimes G_{ij}^1 + \otimes G_{ij}^2 + \dots + \otimes G_{ij}^K] \quad (1)$$

in which $\otimes G_{ij}^K$ is the average value of the attribute ratings for each CT Activity, where $i = 1, 2, \dots, m; j = 1, 2, \dots, n$

Step 2

Normalize the grey decision matrix (see Table 3). The normalization method is utilized to preserve the property that the ranges of the normalized grey number belong to, that is [0, 1].

Table 3: Grey normalized attributes for CT Activities

S _i (CT Activity)	Q ₁ Abstraction	Q ₂ Algorithmic Thinking	Q ₃ Decomposition	Q ₄ Generalization	Q ₅ Evaluation
CT Activity 1	[0.697, 1.000]	[0.733, 1.000]	[0.667, 0.909]	[0.656, 0.955]	[0.697, 1.000]
CT Activity 2	[0.606, 0.788]	[0.733, 1.000]	[0.697, 1.000]	[0.724, 1.000]	[0.606, 0.788]
CT Activity 3	[0.576, 0.758]	[0.433, 0.600]	[0.636, 0.879]	[0.553, 0.700]	[0.576, 0.758]
CT Activity 4	[0.545, 0.667]	[0.267, 0.467]	[0.667, 0.909]	[0.636, 0.913]	[0.545, 0.667]
CT Activity 5	[0.545, 0.667]	[0.333, 0.500]	[0.545, 0.545]	[0.778, 1.000]	[0.545, 0.667]
CT Activity 6	[0.636, 0.818]	[0.267, 0.467]	[0.515, 0.636]	[0.553, 0.700]	[0.636, 0.818]

Each normalized grey interval is expressed as

$$\otimes G_{ij}^* = \left[\frac{\underline{G}_{ij}}{G_j^{\max}}, \frac{\overline{G}_{ij}}{G_j^{\max}} \right] \quad (2)$$

Step 3

As a suggestion, perhaps we could consider taking the more “conservative” lower value from each grey interval that corresponds to each CT skill (see Table 4).

Table 4: values of the lower bound in grey intervals

CT Activity	Abstraction	Algorithmic	Decomposition	Generalization	Evaluation
CT Activity 1	0.697	0.733	0.667	0.656	0.697
CT Activity 2	0.606	0.733	0.697	0.724	0.606
CT Activity 3	0.576	0.433	0.636	0.553	0.576
CT Activity 4	0.545	0.267	0.667	0.636	0.545
CT Activity 5	0.545	0.333	0.545	0.778	0.545
CT Activity 6	0.636	0.267	0.515	0.553	0.636

Step 4

Comparison of the six CT activities that are being considered for their suitability to the learners’ CT developmental needs can be accomplished using, for example, a bar chart (see Figure 1) or a box and whiskers chart (see Figure 2).

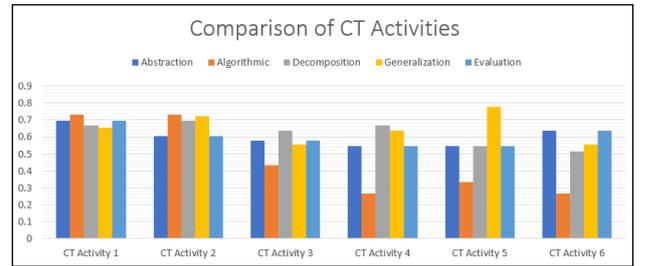


Figure 1: Bar chart comparing multiple CT activities

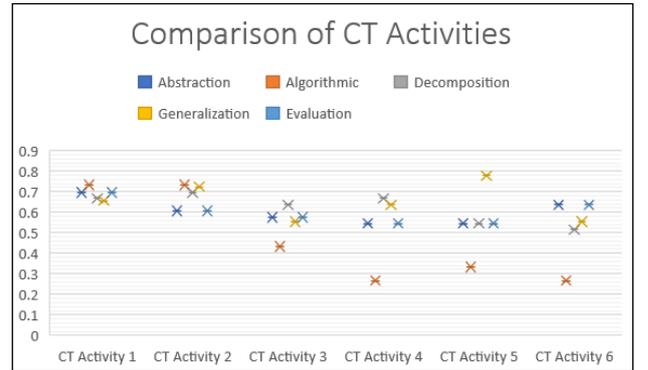


Figure 2: Box and whisker chart comparing multiple CT activities

4. IMPLICATION FOR CT EDUCATION

Researchers (such as Bers, Flannery, Kazakoff, & Sullivan, 2014; Grover, 2013; Portelance & Bers, 2015) concur that CT developmental activities ought to be age- and grade-appropriate. Instead of relying on one decision maker’s “gut feel” or the words of the marketing manager of a third-party CT activity training provider to gauge whether some CT activities would be suitable for the educational institute’s learners, a grey-based approach has been proposed in the current paper for the cross comparison of multiple CT activities. Different combinations of CT skills development offered by each CT activity could be used to inform the decision makers in educational institutions about the suitability of each of the CT activities for their learners. For example, CT Activity 6 (see Figure 1) might involve a lower developmental level of Algorithmic Thinking; however, this type of CT Activity might be more age- or grade-

appropriate for beginner learners of CT. Conversely, CT Activities 1 and 2 might involve higher developmental levels of CT skills, which suggests that they could be more suitable for learners who need to be engaged with something more challenging. Further, in situations where multiple third-party training providers approach educational institutions to offer their CT training services, this proposed approach could be used by the stakeholders (for example, Ministry of Education, principals, vice-principals, heads of departments, and teachers) of the educational institutions to document the cross comparison process of multiple CT activities offered by these third-party vendors, thus contributing to increased transparency in the educational institutions' corporate governance.

5. CONCLUSION

The focus of the paper is on the lack of tools that show what CT skills are addressed and to what extent across various CT activities, especially when there is no consensus on what the CT skill of Algorithmic thinking means, for instance. The tool is useful when dealing with such ambiguity by averaging the inputs of multiple evaluators. Until now, although there are many frameworks for assessing individual CT activities (as mentioned earlier in Section 1), there is no approach in the extant literature for performing the cross comparison of multiple CT activities, which could be used to transparently document the selection criteria by multiple decision makers. These decision makers could include teachers, heads of departments, school principals, researchers, or the Ministry of Education. The transparency of this grey-based approach could potentially contribute to the democratization of the selection process of CT activities, as the input of each decision maker is taken into serious consideration. A worked example of cross comparison between multiple CT activities using hypothetical data has been used to illustrate a proposed grey-based approach. This proposed grey-based approach is a reasonably easy to understand, easy to calculate, and easily implementable CT evaluation tool, which we hope would be considered by decision makers in educational institutions for performing cross comparison of multiple CT activities when they need to evaluate them to find out if they are at the appropriate developmental levels for their learners.

6. REFERENCES

- Anderson, N. (1961). Scales and Statistics. *Psychological Bulletin*, 58(4), 305–316. <https://doi.org/10.1037/h0042576>
- Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer Science Unplugged: School Students Doing Real Computing Without Computers. *Journal of Applied Computing and Information Technology*, 13(1), 20–29.
- Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, 72, 145–157. <https://doi.org/10.1016/j.compedu.2013.10.020>
- Deng, J. (1989). Introduction to Grey System Theory. *The Journal of Grey System*, 1, 1–24.
- du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14, 237–249.
- Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE '13*, 10. <https://doi.org/10.1145/2462476.2466518>
- Griffin, J. M. (2016). Learning by Taking Apart: Deconstructing Code by Reading, Tracing, and Debugging. *Proceedings of the 17th Annual Conference on Information Technology Education (SIGITE '16)*, 148–153. <https://doi.org/10.1145/2978192.2978231>
- Grover, S. (2013). Using a Discourse-Intensive Pedagogy and Android 's App Middle School Students. *Sigsce*, 723–728. <https://doi.org/10.1145/2445196.2445404>
- Grover, S. (2015). "Systems of Assessments" for Deeper Learning of Computational Thinking in K-12. *Annual Meeting of the American Educational Research Association*, (650).
- Li, G., Yamaguchi, D., & Nagai, M. (2007). A grey-based decision-making approach to the supplier selection problem, 46, 573–581. <https://doi.org/10.1016/j.mcm.2006.11.021>
- Liu, S., & Lin, Y. (2010). *Grey Systems: Theory and Applications*. Berlin: Springer-Verlag.
- Liu, S., Yang, Y., & Forrest, J. (2016). *Grey Data Analysis*. Singapore: Springer-Verlag.
- Portelance, D. J., & Bers, M. U. (2015). Code and tell. *Proceedings of the 14th International Conference on Interaction Design and Children - IDC '15*, 271–274. <https://doi.org/10.1145/2771839.2771894>
- Rodriguez, B. R. (2015). *Assessing Computational Thinking in Computer Science Unplugged Activities*. Colorado School of Mines. <https://doi.org/10.1017/CBO9781107415324.004>
- Takaoka, E., Fukushima, Y., Hirose, K., & Hasegawa, T. (2014). Learning Based On Computer Science Unplugged in Computer Science Education : Design , Development , and Assessment, 8(7), 3–7.
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2014). Defining Computational Thinking for Science, Technology, Engineering, and Math. In *American Educational Research Association Annual Meeting*. Philadelphia, Pennsylvania.
- Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London: Mathematical, Physical and Engineering Sciences*, (July), 3717–3725. <https://doi.org/10.1109/IPDPS.2008.4536091>

On Tools that Support the Development of Computational Thinking Skills:

Some Thoughts and Future Vision

Gregorio ROBLES^{1*}, Jean Carlo Rossa HAUCK², Jesús MORENO-LEÓN³, Marcos ROMÁN-GONZÁLEZ⁴,
Roberto NOMBELA⁵, Christiane Gresse von Wangenheim⁶

^{1,3,5} KGB-L3, Universidad Rey Juan Carlos, Madrid, Spain

^{2,6} Department of Informatics and Statistics, Federal University of Santa Catarina, Florianópolis, Brasil

³ INTEF, Madrid, Spain

⁴ Facultad de Educación, Universidad Nacional de Educación a Distancia, Madrid, Spain

grex@gsysc.urjc.es, jean.hauck@ufsc.br, j.moreno@gmail.com, mroman@edu.uned.es,

r.nombelaa@alumnos.urjc.es, gresse@gmail.com

ABSTRACT

Development of Computational Thinking (CT) is an area of many initiatives in the last years, due to the importance of having CT skills. There are many environments that allow learners to develop such skills, for instance Scratch and MIT App Inventor, in a visual and intuitive way. As in professional software development, assisting tools that help and guide learners are starting to appear. In this paper, we discuss the current status of these tools, based on an analysis of what state-of-the-art CT assessment tools, such as Dr. Scratch for Scratch and CodeMaster for App Inventor, offer. We report their limitations and envision and discuss future enhancements.

KEYWORDS

computational thinking, tools, assessment, Scratch, App Inventor

1. INTRODUCTION

The inclusion of computer programming and computational thinking (CT) skills in the school curriculum is one of the main trends in the educational landscape worldwide. This movement has motivated a deep interest among scholars and research institutions, who are analyzing and comparing the approaches and plans of the different initiatives. The reviews on the state of CT in education that have been performed coincide in three main, fundamental aspects that require urgent attention from academia: assessment of CT skills, transference of CT skills and factors affecting CT skills. The topic of this paper is related to assessment of CT skills, although its reach is beyond that specific topic.

There are many initiatives fostering the development of CT skills (Lye & Koh, 2014), such as tools where learners can acquire programming skills by means of using visual programming languages. Some of the most commonly used tools to support CT learning are Scratch (<https://scratch.mit.edu/>), MIT App Inventor (<http://appinventor.mit.edu>), Code.org (<https://code.org/>), Snap! (<https://snap.berkeley.edu/>), among others.

In the teaching of CT in schools, practical activities are typically carried out where learners develop programs using these tools. The resulting projects need to be evaluated in relation to the extent to which they reached the pedagogical goals and also in relation to other aspects, such as: fundamentals of algorithms, use of variables, flow control, modularization of complex tasks, etc. (CSTA, 2017). Most

of these aspects can be evaluated in an automated way, through analysis of the source code developed by the learners (Moreno-León et al., 2015), thus supporting the educator in the assessment and grading of learner's work.

Currently, there are some tools that perform the assessment of CT aspects through the static analysis of projects developed by learners, such as: Dr. Scratch (<http://www.drscratch.org/>), CodeMaster (<http://apps.computacaonaescola.ufsc.br:8080/>), Quizly (<http://appinventor.cs.trincoll.edu/csp/quizly/>), and Ninja Code Village (<http://ik1-325-22639.vs.sakura.ne.jp/ncv4s/>), among others.

This type of assessment presents some limitations, first of all because the tools generally work on the source code, typically only after the learner has finished his/her work. This focus on the source code also limits the assessment, not covering essential CT practices like creativity and collaboration, and sometimes does not provide valuable support for the learner.

The goals of this paper are following: (1) to review the current state of computational thinking assistance tools, and (2) to propose future enhancements for them.

The paper is structured as follows: In the next section we will introduce the state of the art in assessment of CT skills, and focus on two CT assessment tools (Dr. Scratch and Code Master). Section 3 reports the limitations and deficiencies that the aforementioned tools present, while Section 4 offers some enhancements to address those limitations. Conclusions are drawn in Section 5.

2. ASSESSMENT OF CT

Assessment of CT skills is a topic that has gained attention of the research community in recent years. Besides, Dr. Scratch (see Section 2.1) and CodeMaster (see Section 2.2), many other research efforts have been devoted to it, such as Quizly (Maiorana et al., 2015), Fairy Assessment (Werner et al., 2012) and REACT (Koh et al., 2014).

2.1. Dr. Scratch

Dr. Scratch (Moreno-León, Robles & Román-González, 2015) is a free/libre/open source tool that analyzes Scratch projects to assess their level of development of CT skills by inspecting their source code. Dr. Scratch (<http://www.drscratch.org/>) is inspired by Scrape (Wolz, Hallberg & Taylor, 2011) and is based on Hairball, a static

code analyzer for Scratch projects that detects potential issues in the code (Boe et al., 2013).

The CT assessment of Dr. Scratch is based on the degree of development of seven dimensions of the CT competence: abstraction and problem decomposition, logical thinking, synchronization, parallelism, algorithmic notions of control flow, user interactivity and data representation. Each dimension is assigned a score, resulting in an aggregated total mastery score. With this information Dr. Scratch generates a feedback report that include ideas and proposals to enhance the CT score by encourage learners to try new blocks and structures.

Different actions have been performed to validate Dr. Scratch from distinct points of view, showing that the tool is useful for learners and proving its ecological validity (Moreno-León et al., 2015), and comparing Dr. Scratch results to other measurements, such as educator grades of Scratch projects or software engineering complexity metrics, showing convergent validity (Moreno-León, Robles, & Román-González, 2016a; Moreno-León et al., 2017; Román-González et al., 2017).

Finally, since Scratch creations are categorized under different types of projects, such as games, stories or music creations, among others, the results of the analysis of 250 projects of 5 different types show that this topology is replicated when projects are analyzed with Dr. Scratch, thus proving its discriminant validity (Moreno-León, Robles & Román-González, 2018).

2.2. Code Master

CodeMaster is a free web-based tool (<http://apps.computacaonaescola.ufsc.br:8080>) developed to facilitate the assessment and grade of App Inventor and Snap! projects, in a problem-based context, focusing on learning computational thinking in K-12 education. CodeMaster can be used by learners to evaluate their own projects obtaining direct feedback and also by educators to assess and grade all class projects at once, in a comprehensive assessment.

CodeMaster measures the complexity of the App Inventor and Snap! learners' projects using an extended rubric based on the CT framework by Brennan & Resnick (2012), Dr. Scratch and the Mobile CT rubric (Sherman & Martin, 2015). CodeMaster, thus, evaluates several dimensions of CT, such as abstraction, synchronization, parallelism, flow control, user interactivity and data representation. Assessment results are presented to the learner in a visually appealing and stimulating way, represented by a character who has a varied color badge depending on the score reached in the code assessment.

The tool has been tested and applied in real environments and has been observed as a useful, functional, performance-efficient tool to support the assessment of App Inventor and Snap! projects.

3. CURRENT LIMITATIONS

In their current form, the main beneficiaries of CT assessment tools are not learners, but educators. This is because the tools offer an evaluation that is based on the final product, emphasizing the abilities that learners have. If the

tools would address more the learning process, they should emphasize feedback on bad practices and on how the learner can learn more (Robles et al., 2017).

The exclusive focus on source code analysis tends to facilitate the assessment of CT aspects that can be evaluated by automation. However, this focus limits in several ways a more comprehensive assessment of the CT development. It is very difficult, if not impossible, to evaluate creativity or collaboration, for example, only by the static analysis of a learners' piece of source-code.

Despite automated assessment allows educators to devote time to pedagogical issues that require more educator-learner interaction, which has proven to be very positive (Ala-Mutka, 2005), offering an important support to the educator, it may not be directly contributing to the learning process itself.

Automated CT assessment tools typically do not provide a personalized learning experience, tracking the entire learning process, but only evaluating the outcomes at the end of the development process. So, the opportunity to support the learner throughout the learning process and to suggest systematic ways for the development of learner's skills is been lost.

In summary, even if not comprehensive, educators are the main beneficiaries of current CT assessment tools. Their evaluation can be supported and enhanced with these tools; so, even if some aspects such as user interface quality and creativity may not be considered by the tools, the information they offer and the amount of time saved is of high value for educators.

4. ENHANCEMENTS

In this section, we propose a set of enhancements that could be implemented in CT assistance tools.

4.1. Tools More Learner Driven

Tools should focus more on the learner and on the learning process. This means that the major point of interest should not be on the blocks that are used, but on the identification (and explanation of) bad smells (i.e., bad programming practices), dead code (i.e., parts of the program that are never reached), among others (Robles et al., 2017). The rationale for this is that learners are familiar with their own code and, if done properly, will understand the problems of their current solution.

4.2. Assess UI of Projects

Despite its importance, the quality of User Interfaces (UI) has been, in general, ignored during CT learning. Some tools only count the interface components and if some type of arrangement is used. Although some artistic aspects of UIs are difficult to assess, other dimensions of the UI quality, however, can be objectively evaluated, using well-known good practices as a basis. This type of evaluation, if automated, can help learners to improve the quality of their developed UIs.

4.3. Personalize (and follow) the Assessment Process

Tracking the development of CT learners' skills becomes important in order to customize his learning experience. To make this possible, automated assessment and learning

support tools need to be able to identify the learner through the creation of individual accounts.

In addition, individual identification enables educators to follow the development of each learner abilities in the various aspects of the CT, allowing to identify if the learner's progress is adequate and to personalize the tasks and exercises, among others.

4.4. Educator Dashboard

Every modern learning management system includes educator dashboard and learning analytics tracking systems, in order to assess and intervene in real-time and in a personalized way (Kalelioğlu, 2015).

Similar functionality should be included in the assistance tools to help educators have a comprehensive view of their learners, and to follow their learning process. The educator dashboard should be designed in such a way that it highlights the most relevant information, i.e., that information that is easily to obtain in an automated way (i.e., a learner lagging behind or abandoning), but that requires human intervention to solve.

4.5. Identification of Learning Gaps

As in any other formal language, computer programming must be learnt in a systematic way, ensuring that there are no gaps between computational concepts (Rich et al., 2017). If computational concepts are not developed systematically, and if learning gaps are not identified, then misconceptions are likely to appear.

Thus, assistance tools should not only score the presence of certain computational concepts, but also to point out the absence of others in between (Grover & Basu, 2017).

4.6. Identification of Learning Paths

In the same vein, computational concepts can be progressively developed, by means of programming projects with increasing complexity. Current learning paths are monolithic. As shown in (Moreno-León, Robles, & Román-González, 2018, in press), Scratch guides generally begin with programming animations, music and art projects, continues with stories, and finish with games, showing in the process concepts and elements of increasing complexity. This, however, supposes a barrier to those learners who are not interested in games, as their disinterest may lead to not develop higher CT skills.

However, Moreno-León, Robles, & Román-González (2018, in press) also reports that for every category there are projects that show basic, intermediate and advanced CT skills. Thus, it is possible to allow users to set a learning path with the number of phases of their choice and the types of project to include in each level. Future assistance tools should not be limited to receive and assess the projects of the learner, but also to propose him/her feasible and significant learning paths.

4.7. Use of Recommender Systems

Furthermore, the aforementioned learning path can be enhanced by providing the learner with prototypical examples than can be remixed (Dasgupta et al., 2016). Then, assistance tools should not only give feedback about the ongoing programming projects of the learner, but also to propose him/her new projects to be remixed, which are

placed in his/her "Zone of proximal development" (ZPD) (Vygotsky, 1978).

4.8. Other Abilities and Skills

Assistance tools should embrace the analysis and assessment of not-so-objective computational thinking practices based on learner's behavior while programming. High-level skills that should be addressed are reusing, abstracting, modularization, debugging and modeling.

Targeting these skills is not easy as they are tight to the process and obtaining information about them is complex. Nonetheless, we argue that this could be done indirectly by, for instance, the identification of bad smells (see 4.1) and observing how the learner solves them.

4.9. Integrated Instructional Feedback

Currently, a learner interested in receiving automated feedback on a project developed in one of the popular tools (e.g. Scratch, App Inventor or Snap!), needs to export it, and submit it in another tool (e.g., Dr. Scratch or CodeMaster). This tends to difficult the use of such tools and leading the learner to submit his project to analysis only at the end of the development process.

The integration of instructional feedback directly to the development environment could give fast results, as it has been observed in other scenarios (Gonçalves et al, 2017).

4.10. Share and Socialize

Along the formative assessment of the CT skills of the learner, the corresponding assistance tools should not only give feedback to the learner, but also share and socialize his/her achievements with a broader community.

Recent research has demonstrated that individuals who perform more social actions during the learning process, reach higher levels of sophistication in their CT skills and computer programs (Moreno-León, Robles, & Román-González, 2016b). Other research has found that professional developers make a surprisingly rich set of social inferences from the networked activity information, such as inferring someone else's technical goals and vision when they edit code or guessing which of several similar projects has the best chance of thriving in the long term (Dabbish, et al., 2012).

5. CONCLUSION

Computational Thinking is a skill that is vital for the personal and professional development of the citizenship of the 21st century. There are many initiatives that have simplified the acquisition of these skills, mainly by programming in learner-friendly visual interfaces, such as Scratch or MIT App Inventor. In recent times, assistance tools are starting to appear that -on top of the aforementioned platforms- offer assessment and guidance through the learning process. However, at this point these tools are mostly useful for educators. In this paper, we offer some insight of future lines that can make assistance tools better suited for learners. These enhancements range from the introduction of personalized elements that adapt the learning process to the learner, including recommender systems and learning paths, to the evaluation of other skills, such as abstraction, modeling or debugging. We hope to see in the

near future many ideas and implementations targeting these issues to the benefit of educators and learners.

6. ACKNOWLEDGEMENTS

This research was supported in part by CNPQ and in part by the Region of Madrid under project “eMadrid: Investigación y Desarrollo de tecnologías educativas en la Comunidad de Madrid” (S2013/ICE-2715).

7. REFERENCES

- Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer science education*, 15(2), 83-102.
- Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P., & Franklin, D. (2013). Hairball: Lint-inspired static analysis of scratch projects. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 215-220.
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association*, Vancouver, Canada.
- CSTA (2017) K–12 Computer Science Framework. Available at <http://www.k12cs.org>.
- Dabbish, L., Stuart, C., Tsay, J., & Herbsleb, J. (2012). Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*. ACM, 1277-1286.
- Dasgupta, S., Hale, W., Monroy-Hernandez, A., and Hill, B. M. (2016). Remixing as a pathway to computational thinking, in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, ser. CSCW '16, 1438–1449.
- Gonçalves, R. Q., Wangenheim, C A. G, Hauck, J. C. R., Zanella, A. (2017) An Instructional Feedback Technique for Teaching Project Management Tools Aligned With PMBOK. *IEEE Trans. on Education*.
- Grover, S., & Basu, S. (2017). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. In *Proceedings of the 2017 ACM Technical Symposium on Computer Science Education*. ACM, 267-272.
- Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, 52, 200-210.
- Koh, K. H., Basawapatna, A., Nickerson, H., & Repenning, A. (2014). Real time assessment of computational thinking. In *Visual Languages and Human-Centric Computing (VL/HCC), 2014 IEEE Symposium on*. IEEE, 49-52.
- 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.
- Maiorana, F., Giordano, D., & Morelli, R. (2015) Quizly: A live coding assessment platform for App Inventor. In: *Blocks and Beyond Workshop (Blocks and Beyond), 2015 IEEE*. IEEE, 25-30.
- Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia*, 46, 1-23.
- Moreno-León, J., Robles, G., & Román-González, M. (2016a). Comparing Computational Thinking Development Assessment Scores with Software Complexity Metrics. *Proceedings of 2016 IEEE Global Engineering Education Conference*, Abu Dhabi.
- Moreno-León, J., Robles, G., & Román-González, M. (2016b). Examining the Relationship between Socialization and Improved Software Development Skills in the Scratch Code Learning Environment. *J.UCS*, 22(12), 1533-1557.
- Moreno-León, J., Román-González, M., Hartevelde, C., & Robles, G. (2017). On the automatic assessment of computational thinking skills: A comparison with human experts. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 2788-2795). ACM.
- Moreno-León, J., Robles, G., & Román-González, M. (2018). Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing*.
- Rich, K. M., Strickland, C., Binkowski, T. A., Moran, C., & Franklin, D. (2017). K-8 Learning Trajectories Derived from Research Literature: Sequence, Repetition, Conditionals. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 182-190). ACM.
- Robles, G., Moreno-León, J., Aivaloglou, E., & Hermans, F. (2017). Software clones in Scratch projects: On the presence of copy-and-paste in Computational Thinking learning. In *Software Clones (IWSC), 2017 IEEE 11th International Workshop on*. IEEE, 1-7.
- Román-González, M., Moreno-León, J., & Robles, G. (2017). Complementary tools for computational thinking assessment. In *Proceedings of International Conference on Computational Thinking Education (CTE 2017)*. The Educ. Univ. of Hong Kong, 154-159.
- Sherman, M., & Martin, F. (2015). The assessment of mobile computational thinking. *Journal of Computing Sciences in Colleges*, 30(6), 53-59.
- Vygotsky, L. (1978). *Mind in society: The development of higher psychological processes*. Cambridge, MA: Harvard University Press.
- Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). ACM.
- Wolz, U., Hallberg, C., & Taylor, B. (2011). Scrape: A tool for visualizing the code of Scratch programs. In *Poster presented at the 42nd ACM Technical Symposium on Computer Science Education*, Dallas, TX.

计算思维评估的研究现状综述 (2013-2017)

张安琦, 陈桃*, 刘显辛, 程薇

北京师范大学 教育技术学院

zhanganqi19950601@163.com, teastick@gmail.com, 201622010028@mail.bnu.edu.cn, chengweirzh@163.com

摘要

随着计算思维教育的开展, 如何有效地评估学生的计算思维发展水平是近年来国内外教育研究者关注的重点。本研究汇总了2013年至2017年近五年有关计算思维评估的SSCI文献, 并在评估方式、评估对象、评估类别、评估内容等方面进行了对比分析, 结果发现计算思维的评估方式可以分为传统型和基于表现型两大类, 主要对计算思维技能、计算概念、计算实践和计算观念等方面进行评估。希望通过本文的分析给教育研究者在促进学生计算思维的发展、进行计算思维评估设计上提供参考借鉴。

关键字

计算思维; 评估设计; 评估内容

1. 前言

计算思维是所有学习者的宝贵技能, 可以应用于日常解决问题的活动, 并且还可以应用于许多其他学习领域。2006年, 周以真(Jeannette Wing)提出计算思维的概念, 这个概念也逐步受到教育教学领域的广泛重视。2011年国际教育技术协会(ISTE)联合计算机科学教师协会(CSTA)开发了计算思维的操作性定义。近年来各国也开始将计算思维纳入课程学习计划当中, 比如2013年9月, 英国教育部公布全新的以计算思维为核心的计算课程计划, 2014年2月美国College Board以计算思维实践和若干核心概念为主体开发面向高中学生的课程框架。国内外教育研究者开展了一系列有关计算思维教育理论与实践的探索, 评估也成为研究者们关注的重点, 近五年来有关计算思维评估的研究数量也在逐年增长, 本研究对已有文献中计算思维的评估设计展开探索。

2. 文献综述

2.1. 计算思维

计算思维是一种通过计算工具进行信息处理的问题解决过程。计算思维无处不在, 小到日常生活中的琐事, 大到社会问题的处理过程, 都可以抽象为信息处理任务过程中的各种指令或行为。周以真(Wing, 2006)将计算思维界定为一种能力, 这种能力通过熟练地掌握计算机科学的基础概念而得到提高, 计算思维存在于教育教学领域中的多个方面, 不同研究者对计算思维有着不同的认识, 如Flanigan, Peteranetz, Shell, & Soh(2017)认为计算思维和创造性思维是计算机科学内部和外部有价值的工具, 在研究中通过一系列计算创造性练习提高学生计算机科学课程的成绩, 结果表明, 计算创造性练习对成绩有积极影响, 有助于提高学生的计算思维和创造性思维。Jaipal-Jamani & Angeli(2017)

对机器人课程职前教师的自我效能感、对科学概念的理解和计算思维进行前后测, 发现利用机器人进行教学, 能培养对科学概念的理解, 促进计算思维能力的发展。随着计算思维教育的发展, 研究者意识到评估在计算思维教育中发挥着重要作用, 只有了解如何进行评估, 才能把握学生的计算思维发展水平, 从而制定有效的课程计划, 更好地开展计算思维培养方面的课程。

2.2. 计算思维评估

计算思维评估的研究是近几年相关教育研究者关注的重点, 比如Grover等(2017)在提供直观的视觉操作界面的开放式编程活动中, 记录学生在完成编程任务过程中解决问题的行为和表现, 以此为证据衡量和评估学生的计算思维概念和计算思维实践。Korkmaz, Çakir, & Özden(2017)在研究中将计算思维被定义为由ISTE(2015)提出的6项基本技能, 并以此为基础开发计算思维评估量表, 通过问卷、访谈形式来衡量大学生的计算思维水平。Chen等(2017)在机器人课程的学习过程中借助基于计算机科学教师协会标准开发的评估工具, 通过纸笔测验, 观察学生操作机器人的编程过程以及日常事件推理等形式来评估小学生计算思维的应用技能, 结果表明, 该工具具有良好的心理测量特性, 并有可能揭示学生在计算思维学习方面的挑战和发展。(Román-González, Pérez-González, & Jiménez-Fernández, 2017)试图采用心理测量的方法来定义和测量计算思维, 目的在于在提供一个新的计算思维测量仪器, 通过测量发现: 计算思维与空间能力、推理能力和问题解决能力之间存在显著的相关性。Baichang Zhong, Qiyun Wang, Jie Chen, & Yi Li(2016)设计了三维综合评估框架(TDIA), 将方向性、开放性和过程性三个维度整合到评估任务的设计当中, 从计算思维的三个维度: 计算概念、计算实践和计算观念进行全面评估。Choi, Lee, & Lee(2017)开发了基于拼图的算法学习程序(PBAL), 并探究这个程序对学习者的计算思维的影响, 通过观察学习者问题解决的过程、访谈面试等形式评估测量学习者计算思维技能水平; 与传统的算法学习方法相比, PBAL对提高计算思维技能水平有较好的效果。通过文献整理发现, 对于计算思维评估设计的探究是多元而非单一的, 不同研究基于计算思维不同概念框架对学习者的计算思维进行评估, 本文的目的在于研究计算思维评估的现状, 分析整理近五年内在计算思维评估的方式、对象、类别、内容等方面的研究, 为相关教育研究人员提供计算思维评估设计上的参考。

3. 研究问题

通过文献分析，本文提出的研究问题如下：

- (1) 近五年的相关研究中，研究者是如何评估学生的计算思维发展水平？
- (2) 在评估过程中会涉及到计算思维的哪些方面？
- (3) 计算思维的评估方式与评估对象、评估内容之间是否有关系？

4. 研究方法

4.1. 文献纳入标准

基于一定的标准对文献进行评估，选择符合标准的文章纳入本文的研究，本文纳入标准如下：

- (1) 教育教学领域内计算思维的相关研究；
- (2) 研究内容涉及计算思维评估；
- (3) 实证研究；
- (4) 在 SSCI 期刊上发表；
- (5) 在 2013 年至 2017 年发表。

4.2. 文献检索与筛选

文献检索与筛选过程分为以下几个阶段进行：

首先，两位研究者通过 Web of Science 核心合集数据库，以“Computational Thinking”为关键词，对从 2013 年到 2017 年的 SSCI 期刊文章进行了检索，共获得 337 篇文章。

其次，两位研究者对检索到的文章的摘要进行浏览，舍弃不属于教育教学领域内计算思维的相关研究；接下来对剩余的 52 篇文章进行全文阅读，不是实证研究的、研究内容不涉及计算思维评估以及没有具体介绍计算思维评估过程的研究被排除。如果对研究是否保留存在疑问，则两名研究人员独立审查全文，然后一起作出最终决定。

最后，共有 11 项研究符合纳入标准。

4.3. 文献编码

确定符合纳入标准的研究后，本研究的编码方法由两个主要部分组成：

- (1) 基本信息：作者、发表年份、国家或地区。
- (2) 评估设计：评估方式、评估主体、评估对象、评估类别、评估内容。

5. 结果

研究结果从以下四个方面展开叙述：

5.1. 纳入研究的基本信息

从表 1 的统计可以看出，对于计算思维评估的研究逐渐受到各个国家和地区的重视，评估对象涵盖中小学阶段、大学生和职前教师，大多数研究的评估对象是 K-12 阶段的学生。2014 年 2 月美国 College Board 发布了最新版的计算机科学原理 (Computer Science Principles) 课程框架，该课程面向高中学生，以计算思维实践和

若干核心概念为主体，在 K-12 教育阶段注重计算思维的培养与评估更有助于学校了解学生的计算思维现状，为以后开展计算思维的教学实践，提高学生计算思维水平提供参考依据。

表 1 国家或地区、评估主体和评估对象梳理

研究	国家或地区	评估主体	评估对象
(Grover 等, 2017)	美国	研究人员	9、11、12 年级学生
(Korkmaz, Çakir, & Özden, 2017)	土耳其	教师、研究人员	大学生
(Chen 等, 2017)	美国	研究人员	5 年级学生
(Jaipal-Jamani & Angeli, 2017)	美国	研究人员	职前教师
(Tsai, Shen, Tsai, & Chen, 2017)	台湾	教师	大学生
(Basu, Biswas, & Kinnebrew, 2017)	美国	研究人员	6 年级学生
(Choi, Lee, & Lee, 2017)	美国	教师	4-6 年级学生
(Román-González, Pérez-González, & Jiménez-Fernández, 2017)	西班牙	教师	5-10 年级学生
(Atmatzidou & Demetriadis, 2016)	希腊	教师	初中、高职学生
(Baichang Zhong, Qiyun Wang, Jie Chen, & Yi Li, 2016)	中国	研究人员	6 年级学生
(Byeongsu Kim, Taehun Kim, & Jonghoon Kim, 2013)	韩国	研究人员	大学生

5.2. 评估方式

根据文献资料，把文献中出现的评估方式以及篇数整理如图 1 所示，在进行计算思维评估时，研究者主要采用量表、框架等工具来进行评估，评估方式一般分为两类：一类是传统方式，即纸笔测试，另一类是基于学生课堂表现的新型评估方式，如课堂观察、访谈面试、学生编码方案、口头表达、反馈报告、设计方案等。Jaipal-Jamani、Basu 和 Atmatzidou 在研究中都选择用前后测来了解学生已有的知识水平和学习的进步程度，这说明在知识掌握方面，传统的测试方式得到了较好的认可。需要说明的是，Byeongsu Kim, Taehun Kim, & Jonghoon Kim(2013)针对于非计算机专业的大学生，采用纸笔编程的策略，注重于将我们的心智模型转化为落在纸面上的逻辑表示，以提高他们对计算思维的理解和运用，增加学习计算机科学的兴趣，虽然

不借助于计算机，采用传统的纸和笔，但区别于传统评估方式测试题目的形式，其目的在于观察学生的编程过程方案、逻辑思维转换等，没有明确的量化等级评判标准，也看作是一种基于学生表现的评估方式。虽然研究者们都在积极开发新型的评估方式用于计算思维发展水平的评估，但传统的评估方式在评估学生知识基础方面作用毋庸置疑，因此许多研究者采用传统型与基于表现型相结合的多元评估方式评估学生的计算思维发展水平，如表 2 所示。

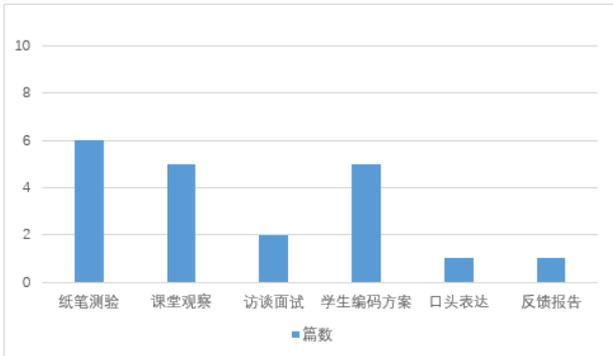


图 1 评估方式篇数统计

5.3. 评估类别

针对 K-12 阶段，表 2 从评估方式、评估类别、评估内容三个维度进行了整理。为了有效评估学生的计算思维发展水平，研究人员、教师运用一套综合的评价体系，提供给学生不同的展示机会，通过各种各样的方式评估学生的知识储备、课堂表现。综合的评价体系包括三大评价类别：诊断性评价、形成性评价和总结性评价。

5.4. 评估内容

如表 2 所示，计算思维技能、计算概念、计算实践和计算观念是研究者进行计算思维评估的主要内容维度。美国麻省理工学院媒体实验室从三个维度上定义分析计算思维，即计算概念、计算实践和计算观念，是大部分研究者进行计算思维评估的概念框架。还有部分研究者评估学生的计算思维技能，但对于计算技能的评估维度确定存在一定的差异，例如 Atmatzidou & Demetriadis(2016)根据五个维度的计算思维概念框架，从抽象、概括、算法、模块化、分解五个方面评估学生的计算思维技能，Korkmaz 等(2017)设计计算思维评估量表，包含六个方面技能的题目，这六个方面分别是交际技能、算法思维、批判思维、合作性、创造力和问题解决技能。还有 Jaipal-Jamani、Chen 在编程环境下，通过记录学生编程过程、布置测试题的形式评估学生计算技能的发展。

此外，通过文献整理，我们发现评估方式与评估对象、评估内容之间有着微妙的联系，对大学生和职前教师的计算思维评估更倾向于使用传统的纸笔测试，而基本上 K-12 阶段的评估都采用基于表现型或传统型与表现型相结合的方式，本研究认为评估方式的选择与评估对象的认知发展水平有一定的联系，随着年龄的增长，学生的认知发展水平提高，对题目的理解程度也随之提升。研究还发现评估与概念相关的内容时，大

多采用传统的纸笔测验，进行前后测的对比来分析学生计算思维知识的发展情况；在评估与实际操作相关的实践类内容时，多采用课堂观察、访谈面试记录学生编程过程等基于表现的方式进行评估。评估对象的定位也是选择计算思维评估方式、评估类型的主要参考依据。

表 2 K-12 阶段计算思维评估方式、类别、内容梳理

研究	评估方式		评估类别		评估内容			
	传统型	基于表现型	形成性评估	总结性评估	计算思维技能	计算概念	计算实践	计算观念
(Grover 等, 2017)	✓	✓		✓		✓	✓	
(Chen 等, 2017)		✓	✓		✓			
(Basu, Biswas, & Kinnebrew, 2017)	✓	✓		✓	✓	✓		
(Choi, Lee, & Lee, 2017)	✓	✓		✓	✓			
(Román-González, Pérez-González, & Jiménez-Fernández, 2017)		✓	✓		✓			
(Atmatzidou & Demetriadis, 2016)	✓	✓	✓	✓	✓			
(Baichang Zhong, Qiyun Wang, Jie Chen, & Yi Li, 2016)		✓	✓			✓	✓	✓

6. 讨论

本研究对计算思维评估方式、类别、对象以及内容的整理分析给缺乏评估经验的研究人员、教师提供一些实践参考。

据了解，澳大利亚、英国、台湾等国家或地区曾举办计算思维挑战赛，目标人群是 K-12 阶段的学生，比赛题目类型多为选择题，考察内容涵盖排程、优化、算法等等，说明计算思维已经受到各国家、地区的重视。在实际教学过程当中，学生作为学习的主体，也可以参与评估体系的建设，增强教育者与受教育者的相互交流，有助于评估体系的建立。评估类别、方式推崇多元化，如传统方式与表现型方式相结合，形成性评

估与总结性评估相结合，更清楚地定位学生的水平和教师的教学结果。

在未来的研究中，如何提高教师、学生计算思维发展水平，如何更有效开展计算思维相关教学活动是计算思维相关研究者持续关注 and 探索的方向。

7. 参考文献

- 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(Part B), 661–670. <https://doi.org/10.1016/j.robot.2015.10.008>
- Baichang Zhong, Qiyun Wang, Jie Chen, & Yi Li. (2016). An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research*, 53(4), 562–590. <https://doi.org/10.1177/0735633115608444>
- Byeongsu Kim, Taehun Kim, & Jonghoon Kim. (2013). Paper-and-Pencil Programming Strategy toward Computational Thinking for Non-Majors: Design Your Solution. *Journal of Educational Computing Research*, 49(4), 437–459. <https://doi.org/10.2190/EC.49.4.b>
- 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(Supplement C), 162–175. <https://doi.org/10.1016/j.compedu.2017.03.001>
- Choi, J., Lee, Y., & Lee, E. (2017). Puzzle Based Algorithm Learning for Cultivating Computational Thinking. *Wireless Personal Communications*, 93(1), 131–145. <https://doi.org/10.1007/s11277-016-3679-9>
- Flanigan, A. E., Peteranetz, M. S., Shell, D. F., & Soh, L.-K. (2017). Implicit intelligence beliefs of computer science students: Exploring change across the semester. *Contemporary Educational Psychology*, 48(Supplement C), 179–196. <https://doi.org/10.1016/j.cedpsych.2016.10.003>
- Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., & Stamper, J. (2017). A Framework for Using Hypothesis-Driven Approaches to Support Data-Driven Learning Analytics in Measuring Computational Thinking in Block-Based Programming Environments. *Acm Transactions on Computing Education*, 17(3), 14. <https://doi.org/10.1145/3105910>
- 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. <https://doi.org/10.1007/s10956-016-9663-z>
- Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, 72(Supplement C), 558–569. <https://doi.org/10.1016/j.chb.2017.01.005>
- 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(Supplement C), 678–691. <https://doi.org/10.1016/j.chb.2016.08.047>
- Wing, J. (2006). Computational thinking. *Communications of the ACM*.

基於專家知識地圖引導慕課學習思維

曾建維^{1*}, 黃能富², 李加安²

¹台灣清華大學教學發展中心

²台灣清華大資訊工程學系

darkdreams0802@gmail.com, nhuang@cs.nthu.edu.tw, ckstar2001@gmail.com

摘要

大規模線上開放式課程 (Massive Open Online Course, 慕課) 已在全球高等教育發展為成熟的學習模式, 如何針對慕課大規模且不同程度、年齡的學習者提供有效的評量機制為發展慕課之新興議題。有別於傳統以教學者為核心的線上課程, 慕課主要以自我調整學習 (Self-regulated Learning) 作為發展模式, 由於學習先備知識 (Prerequisites) 不同導致程度落差極大, 慕課具有根本性的問題: 如完課率低落與缺乏學習指引, 本研究針對慕課之缺陷進行相對應的系統設計。透過專家知識地圖的結構, 建構慕課學習輔助之自動化知識地圖, 將可成為提升課程品質與個人化學習的方法。

關鍵字

慕課; 自我調節學習; 先備知識; 知識地圖。

1. 前言

慕課在全球都相當盛行且對於高等教育造成重要變革, 最為人詬病的就是完課率非常低落 (Freitas, Morgan, Gibson, 2015; Perna, Ruby, Boruch, Wang, Scull, Ahmad, & Evans, 2014), 然而慕課同儕互評的威脅、網絡作弊等相關議題一再挑戰著慕課發展的地位 (Hew & Cheung, 2014)。傳統的線上學習評量機制雖然可以提供有效的指引原則, 如何針對慕課大規模且不同程度、年齡的學習者提供有效的網路化評量機制實為有效發展慕課課程的新興議題。

慕課上的學習者需要根據自身程度, 訂定學習目標、學習策略以精熟課程的內容。透過一系列的教學影片、隨堂練習、討論區以及其他互動功能, 學習者需要能夠發展適切的「自我調整學習」能力引導良好的自主學習。有鑒於自我調整學習對線上學習的重要性及慕課中評量系統的缺陷。本研究將發展「知識地圖」的學習工具, 協助學習者引導慕課上的思維能力, 提升學生學習成效及自我調整能力。

2. 文獻探討

面對慕課如此大規模的學習模式, 應抱持著正面的態度設計、轉化、改變, 以證據為基礎 (evidence-based) 之研究方法改善並提升慕課之相關設計。本研究將針對慕課之缺陷做相對應的完善設計, 透過學習概念的知識結構化提升慕課自主學習者成效, 進而增進其自律學習策略與目標。

2.1. 慕課

慕課特色是修課人數多、學生基礎差異大、全部線上授課 (沒有實體教室)、授課時間較短 (5 至 8 週)、

線上考試、沒有學分。慕課起源於開放教育資源運動和學習連接主義的思潮。強調大規模 (大量學員) 之線上課程, 能提供更多的線上師生互動以及同儕互動學習機制, 同時將學習自主權以及學習的節奏交還給學員 (黃能富, 2015)。

2.2. 慕課評量機制的缺失

許多研究指出慕課具有根本性的疑慮, 最為人熟知的就是慕課完課率非常低 (Freitas, Morgan, & Gibson, 2015) 與高學習流失率 (Daniel, 2012), 由於無法即時監控作答, 潛在許多作弊的缺失, 如此限制慕課無法成為具公正性、甚至授予可信賴的課程學分或是修課證明 (Bady, 2013), 因此, 運用有效的測量工具以檢驗學生在混成、遠距、或虛擬學習環境的學習投入有其需求且十分重要 (Henrie, Halverson, & Graham, 2015)。

2.3. 自我調節學習

自我調節學習為學習者進行自律學習, 於過程中系統化的實現其「學習目標」(Zimmerman & Schunk, 2001), Zimmerman 認為在環境中學習者自我調節學習能力是不可或缺的, 強調於培養利用良好的自我管理技能去因應突發狀況的能力, 過程中個人技能運作的知識與意願更應完備, 同時定義為學習者自身於學習過程中自我的計畫、執行和評價, 其涉及在學習循環過程中持續決定認知、動機和行為 (Zimmerman, 2000)。自我調整學習的觀點注重於學生的知識、後設認知技能、動機和認知, 強調自我調整學習是將「知識」和「技能」進行相互協調; 並將自我調整學習定義為「將個人自動化和控制相連結, 個人呈現自我監控狀態, 調整朝向目標的行動, 發展出類似專家般的知識並且自我改善。」(Patrick & Middleton, 2001)。

2.4. 知識地圖

目前盛行於測驗界針對認知診斷的研究主要是假設認知概念間彼此應視為是相依且依循某種合理的結構 (de la Torre, 2010)。知識地圖 (Knowledge Map, K-Map) 透過圖形化的描述知識分布與結構、知識關聯結構, 余民寧 (2011) 認為認知診斷為「根據某種認知科學的理論為基礎, 以該理論設計診斷測驗試題, 再提出評量該理論的可能知識理論模式, 以驗證該理論下的評量是否成立」。

3. 研究設計

3.1. 研究發展平台

本研究使用台灣清華大學學聯網 (ShareCourse) 平台，ShareCourse 於 2012 年由清華大學創立，根據 2014 年全球最大的慕課課程評比「果殼網 MOOC 學院調查研究報告」，ShareCourse 更在「課程質量最佳」榮獲全球第二的殊榮。目前台灣計有超過 50 個大專院校等單位加入學聯網，總計開設超過 300 門課程，目前已開發行動作業系統 (Android 與 iOS) 的慕課應用軟體 (APP)，並積極提供各種客製化需求，進而符合課程授課教師的作業、考試需求等。

3.2. 研究發展課程

研究預計使用台灣清華大學數學系顏東勇教授所錄製的微積分課程，顏教授具有多年開設慕課課程的豐富經驗，2016 年 9 月開設於「中國大學 MOOC」，總修課人數達兩萬四千多人。微積分課程為奠定一般理工學院所需的基本數學能力，將針對單元主要定義或定理作講解，同時推導定理或公式，並配合例題運用之。任何具備中學數學程度者皆可學習，將可奠定工程數學、複變函數與高等微積分的學習基礎，同時經由演算之過程培養學生邏輯分析之能力，是一門大學生必修的基礎重要課程，下表 1 為微積分十五週課程之教學內容。

表 1 微積分教學內容

週次	預計教學內容
零	前測與線上問卷發放
一	Limit and Continuous function
二	Continuity and Differentiation
三	Differentiation and the Mean Value Theorem
四	Mean value theorem
五	第一次考試
六	Applications of the first and second derivatives
七	Integrations and Fundamental Theorem of
八	Areas and volumes from definite integrals
九	The natural logarithm functions
十	第二次考試
十一	The natural exponential function and the inverse trigonometric functions
十二	Integration by parts and the trigonometric
十三	The trigonometric substitution and the partial
十四	L'Hopital's rule and improper integrals
十五	期末考

3.3. 建構知識地圖

傳統的學習評量只在測驗後提供一個評斷的分數或標準參照，然而慕課有別於一般的線上學習方式，每位學生的學習能力與速度皆不同，慕課的教學設計為教師設定學習步驟，學生依循學習影片、練習、作業與測驗等進度，並無完整知識層面闡述與說明，缺乏學習依歸與指引，因此本實驗課程為教師依照學科專業判斷單元知識結構，依據概念學習順序與其階層，建構出知識地圖，下圖五為台灣清華大學慕課：微積分第一週「極限與連續」知識地圖。

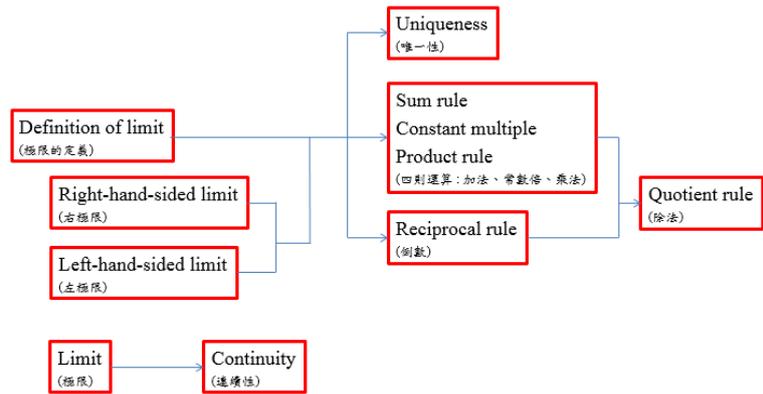


圖 1 微積分第一週知識地圖 (教師編排)

3.4. 自動建構知識地圖

傳統教學設計原則為先行設計課綱，教師依據課綱編排課程素材，慕課教師設計課程以往為遵循實體課堂安排，後續將視頻模組化，因此無法具體將知識結構呈現。教師依照慕課編排反思此知識地圖需要花費相當多心力，因此本模組透過文字探勘 (Text Mining) 技術，萃取教材投影片 (PDF) 文字的上下層級關係、字型大小以及文字斷詞，透過資訊檢索與文字挖掘的常用加權技術 TF-IDF (Term Frequency-Inverse Document Frequency) 將重點關鍵字詞萃取出來，再利用所分析出的上下層級關係，將重點關鍵字建立上下層級的關係，並建立出課程整體架構，最後將資料視覺化，畫出具有方向性的知識結構圖。最終讓教師、教學助理 (TA) 修正此知識地圖，除了可符合慕課的學習順序規劃，也大幅縮短教師與助教製作知識地圖的時間。

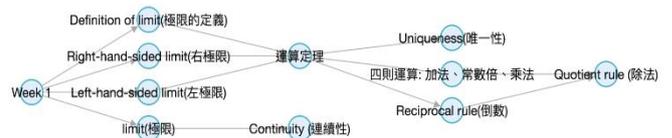


圖 2 微積分第一週知識地圖 (系統生成)

分析講義 PDF 的部分，透過 Open Source: Pdftminer 將資訊從 PDF 文件中萃取出來的一種套件，其主要是專注於 PDF 檔案裡文字資料的取得，對於教材的解析有

著極大的幫助。其作法如下，首先輸入 PDF 文件，文件分析以「頁」為單位，更進一步分析頁面中的所有物件。一頁中，會有許多物件，每個物件的所屬類型不同，有文字、圖片、表格。如下圖 3 為例，頁面具有 5 個物件（如紅框所示）。

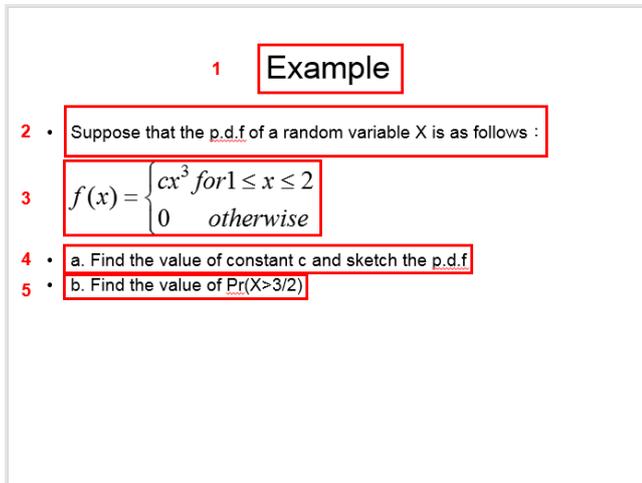


圖 3 知識地圖萃取範例

3.5. 系統架構

本研究建構之診斷教學系統架構圖，將以目前學聯網平台 ShareCourse 為架構進行大規模線上診斷評量系統開發，建構主從架構（client/server）的 internet 作為網路骨幹，由 server 端（web server）負責 client 端（browser）的管理控制，當資料在 client 端做前置處理後，傳回 server 端的題庫系統（MySQL）配合出題。為避免網路傳輸時擁塞的情形，及施測時學生作答的獨立性，以網際網路上能執行運作為主，線上測驗系統以 Linux Cent OS 作為工作平台，PHP 程式語言為基礎撰寫 client 端前置作業之準備，如試題測驗。而在後端處理上，同樣以 PHP 作為 Web server 和 MySQL 溝通的橋樑，負責記錄測驗結果，系統架構圖如下圖 4 所示。

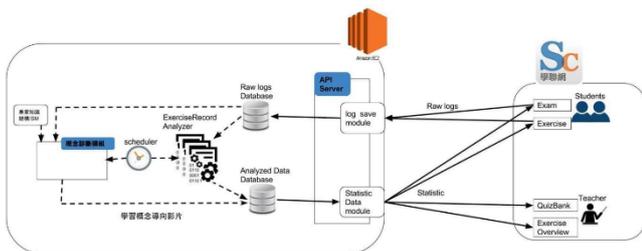


圖 4 系統架構圖

4. 預期成效

(1) 本研究將製作出專家知識結構模組，透過專家判定知識上下位結構與其關聯性，模組可新增單元概念矩陣並依照其關聯，產生專家知識結構圖。

(2) 檢視實驗組與控制組的成績級距與顯著差異，作為教師改進慕課教學的參考指標。

(3) 以既有慕課規劃，設計學習者知識結構圖，省去教師重新錄製課程，並提供未來課程改進的建議。

(4) 透過知識結構提供學員概念指引並導入學習概念影片回顧，協助學員提升慕課上的思維能力，嘉惠平台上廣大的學員。

5. 參考文獻

余民寧 (2011)。教育測驗與評量：成就測驗與教學評量。台北市：心理。

黃能富 (2015)。磨課師 (MOOCs) 與師博課 (SPOCs) 協同授課之翻轉教學法。教育脈動，1，101-110。

Bady, A. (2013). The MOOC moment and the end of reform. *Liberal Education*, 99(4), 6.

Daniel, J. (2012). Making sense of MOOCs: Musings in a maze of myth, paradox and possibility. *Journal of Interactive Media in Education*.

de la Torre, J. (2011). The generalized DINA model framework. *Psychometrika*, 76,179-199.

Freitas, S. I., Morgan, J., & Gibson, D. (2015). Will MOOCs transform learning and teaching in higher education? Engagement and course retention in online learning provision. *British Journal of Educational Technology*, 46(3), 455-471.

Henrie, C. R., Halverson, L. R., & Graham, C. R. (2015). Measuring student engagement in technology-mediated learning: A review. *Computers & Education*, 90, 36-53.

Hew, K. F., & Cheung, W. S. (2014). Students' and instructors' use of massive open online courses (MOOCs): Motivations and challenges. *Educational Research Review*, 12, 45-58.

Patrick, H., & Middleton, M. J. (2001). Turning the kaleidoscope: What we see when self-regulated learning is viewed with a qualitative lens. *Educational Psychologist*, 37, 27-39.

Perna, L. W., Ruby, A., Boruch, R. F., Wang, N., Scull, J., Ahmad, S., & Evans, C. (2014). Moving through MOOCs: Understanding the progression of users in massive open online courses. *Educational Researcher*, 43(9), 421-432.

Zimmerman, B. J. (2000). *Attaining self-regulated learning: a social-cognitive perspective*. In M. Boekaerts, P. Pintrich, & M. Zeidner (Eds.), *Handbook of self-regulation*. (pp. 13-39). San Diego, CA: Academic Press.

Zimmerman, B. J., & Schunk, D. H. (2001). *Self-regulated learning and academic achievement: Theoretical perspectives* (2nd ed.). Mahwah, NJ.: Lawrence Erlbaum Associates.

Computational Thinking and Teacher Development

Computational Thinking Reshapes the Teachers' Perspective on Human Mind towards Teaching and Learning Process

Hew-mee CHEAH
University of Malaya, Malaysia
eleanorcheah.creativeteaching@gmail.com

ABSTRACT

The purpose of this paper is to share on the method used in creating self-realization among Malaysian Educators on the needs of adopting Computational Thinking (CT) Skills. This is an important step to begin the process of change. Once they have accepted the needs of CT, they could be the change agents to shift the Malaysian Education's paradigm by integrating CT into their teaching and learning skills successfully. A total of 21 participants attended a CT training conducted by the author, and the training was aimed to create awareness on (i) educators' perspective on how the human minds work towards the teaching and learning process, (ii) understand that CT is a unique school of thought, and (iii) it is one important skills in this 21st Century. Unplugged activities were being used to create the awareness where the findings clearly showed that the activities used during the training brought much positive results for the whole purpose of this study.

KEYWORDS

Computational Thinking, Unplugged Activity, School of Thought, Reshape Perspective.

1. INTRODUCTION

Malaysia started to promote Computational Thinking (CT) in the year 2016 and integrated it into learning modules especially in ICT subject. This was stated in the 11th Malaysia Plan, which would run from the year 2016 until 2020 (Economic Planning Unit, 2015). The Malaysia Digital Economy Corporation (MDEC) is the sole driving force in making sure the success of this plan ultimately. This is a huge project involving teachers training, alteration of the curriculum, and change management for the stakeholders' readiness.

1.1 Teacher Training

In 2016, MDEC had organized a training programme called "Computational Thinking & Computer Science Teaching Certification Programme" (CT&CS TCP). This certification programme was aimed to build teachers' understanding on CT, and ultimately transfer CT skills to the students in all the schools. The programme started off with the training for selected 100 lecturers from the Teacher Training Institution. Once they had gone through the entire certification process and certified as a Master Trainer (MT), they could start training all the pre-service teachers. In the following years, 36 lecturers from 6 public universities were trained and certified as MT to conduct the same training for all the other in-service teachers.

The CT&CS TCP consists of 3 parts. Part 1 is a 5-day face-to-face training (8 hours a day) conducted by the author for

over one week. Part 2 required participants to submit a programming project. They had to exemplify a range of programming techniques learned during the Part 1 training. They would start this project right after the Part 1 and were given two weeks to complete it. In Part 3, the participants needed to show some particular aspect of CT pedagogy by carrying out a classroom investigation. They had to submit their own video and report on their findings after they had conducted similar lessons in actual classrooms, which demonstrated how the CT pedagogy had effectively helped their weaker students in learning certain difficult topics.

1.2 Curriculum

In order to ease the teachers' implementation of CT education, MDEC had successfully developed a series of teaching modules. These teaching modules covered all the 8 subjects of the primary level, Computer Science Foundation for the Year 7 – 9, and Computer Science for the Year 10 – 11. It gave these teachers some basic ideas on the various ways CT could be integrated into their Teaching and Learning (T&L).

1.3 Change Management

As for the stakeholders' readiness, MDEC works closely with the Ministry of Education (MOE) Malaysia and various State Education Departments to conduct different workshops for the principals, school managements, teachers and students for this CT awareness.

The CT is truly a new concept to the Malaysians, where many teachers were often being confused by this different school of thought. Some of them had thought that CT was actually focused on the engineering thinking, or scientific and mathematical thinking. Some wrongly thought it was just another problem solving skills and couldn't understand why it was being focused on. The investigation by Ling et al in 2017 showed that teachers often related this CT to ICT instead. They thought that one must acquire the ICT knowledge to be able to integrate the CT into teaching and learning (Ling, Saibin, Labadin, & Abdul Aziz, 2017).

2. BACKGROUND OF THE STUDY

There are three main purposes to this study. It is to demonstrate how Unplugged Activities would:

- i. enable educators to reshape their perspective on human mind in T&L process,
- ii. demonstrate that CT is a unique school of thought.
- iii. create realization on the importance of CT.

The three unplugged activities were: 1: Tangram 2: Monster face 3: Algorithm (further discussed in page 3).

2.1 *Reshape the Educators' Perspective on human mind in T&L process*

According to the statistics conducted by the Higher Education Leadership Academy at the Ministry of Higher Education Malaysia in 2011, the results showed that 50% of the lessons delivered by 41 schools across Malaysia were unsatisfactory. The researchers had followed 125 lessons, and most of the lessons did not engage students into learning. These lessons were being conducted using the teacher-centered learning method. Most emphasis was towards memorizing the questions and answering techniques, instead of instilling higher order thinking skills. The assessments were mostly tested on the student's ability in recalling concepts (70% of all the lessons observed) rather than to analyse and interpret data (18%) or synthesize information (15%) (Project Management Office 2012).

Siti Hendon Sheikh Abdullah had conducted a qualitative research in 2013, where she observed the trainee teachers of 9 primary schools who had delivered various Physics topics. The results clearly showed that those trainee teachers had tried to use the inquiry approach, but it was not being conducted effectively. That was due to the trainee teachers' failure in carrying out the teaching and learning constructively. Those trainee teachers were not skillful enough in using the Inquiry-Based Approach to conduct teaching and learning because they had failed to think constructively (Abdullah, 2013).

There is a definite need to build the educators' constructive thinking skills and we must bring this awareness to their conscious level, so that they could recognize this deficit (Adam, n.d.). Once they are fully aware on the areas of their weakness, they could easily adapt and materialize the changes immediately.

2.2 *Computational Thinking as a unique school of thought*

There were famous Mathematicians like John Napier, Charles Babbage, Lady Ada Lovelace and etc who were the pioneer contributors to the formation of computer science as a discipline. The Engineers like Herman Hollerith and Vannevar Bush (just to name a few) had also built punch card and electric motors which became the fundamental architecture design of the modern computers (CMU, n.d.). All these developments had led some people to mistakenly believe that the CT resembles the mathematical, engineering, or Scientific Thinking.

In the book titled "Mastery Algorithms", the author Domingos had written a good description on how the CT is different from these schools of thought. He pointed out that a scientist focuses on theories and the engineer focuses on practical, while the computer scientist actually works on both the theories and practical together (Domingos, 2015). In a more layman understanding, a scientist forms formula while the engineer uses this formula to build things, but a computer scientist needs to come up with formula (example: syntax) and work on the transistors (engineering work).

When someone focuses on the computer science related coding work for a period of time, it will eventually change their thinking patterns. Kim et al had done a research in the year 2013 and discovered that computer programming

enhanced creative problem solving ability for both ordinary and gifted learners (Kim, S., Chung, K., Yu, H., 2013). This implicates that programming activity could enhance a person's thinking.

The author views CT as a collective of schools of thought, which is a deeper and more revolutionary thinking level. Just like dementia is a collective of various symptoms, whereby Alzheimer and Parkinson actually branched out from it; The Computational Thinking is a collective schools of thought, where scientific thinking, engineering thinking, and mathematical thinking are all part of it.

2.3 *The importance of Computational Thinking*

The arrival of the 21st Century, where technology advances exponentially (Nagy, Farmer, Quan & Trancik, 2013), has led to a paradigm shift in education. In order to equip our future generations with problem solving skills to solve complex problems brought by the advanced technology, we need to redesign our educational standard by imparting thinking skills, especially CT in this context, to benefit our young learners.

All the educators today should see the coming of this technology wave, where the Industry Revolution 4.0 would be sweeping around the globe soon. We should therefore prepare our young learners and future leaders with this CT skills.

3. METHODS

The study used qualitative pre-post "self-assessment" approach (Bhanji, F. et al, 2012).

3.1 *Participants*

A total of 21 lecturers from various university faculties who had not attended any CT training had participated in this study. They were selected lecturers from a mixture of the faculties of computer science, engineering, and education.

3.2 *Pre-training assessment*

A pre-training survey was conducted for self-assessment on

- i. understanding on CT,

Q: In your opinion, what is Computational Thinking (CT)?

This question was used to assess the understanding of the participants on CT. By knowing that CT was a new concept to them, author would need to synchronize with all the participants and get them to understand to the importance of CT, before the training of CT could be conducted.

- ii. ability to view from the students' perspective,

Q: I am able to identify the students' thinking pattern and make full use of it for teaching & learning process.

This question eventually led to close the gap between the teacher and students towards the process of teaching and learning.

- iii. problem solving skills.

Q: I think my problem solving skills is _____, because _____.

This question was aimed to demonstrate that CT is a unique problem skills / thought process. The participants would have to assess if CT training had helped to scale up and improve their problem skills.

3.3 Unplugged Activity 1

The training began with Unplugged Activity 1. It was modified from Algorithm Unplugged Activity 6 by Code.org.

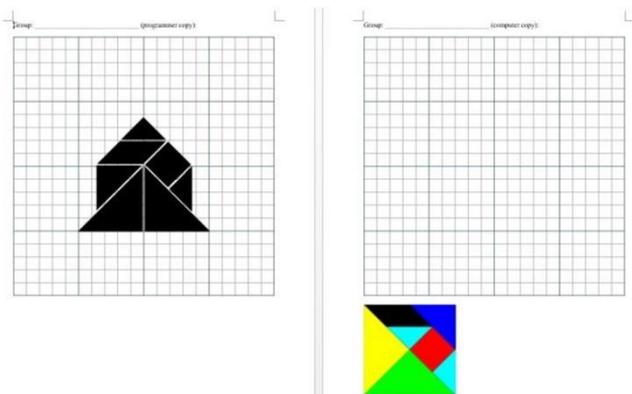


Figure 1. Unplugged Activity 1

All the participants worked in groups of three / four persons. Each group was given a different tangram picture. They had to use their problem solving skills to write instructions for the computer to form the same tangram picture. Once they had written the instructions, two groups were paired to take turns in playing their roles as the computer and also as the programmer. When they were the programmer, they had to read out the instructions. The other group who role played as the computer and sat back to back with the programmer, had to form the picture based on the instructions heard.

This activity demonstrated clearly how the humans need to carefully plan a successful communication with a computer, which also raised up the participant's awareness on its importance to see from the second person's perspective in real-life communications, especially throughout the teaching and learning process.

3.4 Unplugged Activity 2

The next activity was Unplugged Activity 2. It was adopted from Barefoot CAS UK. The original name for this Unplugged Activity was "Crazy Character Algorithms" (Barefoot)

This activity simplified the problem by providing the "ingredients" of the crazy character with simple instructions next to it.

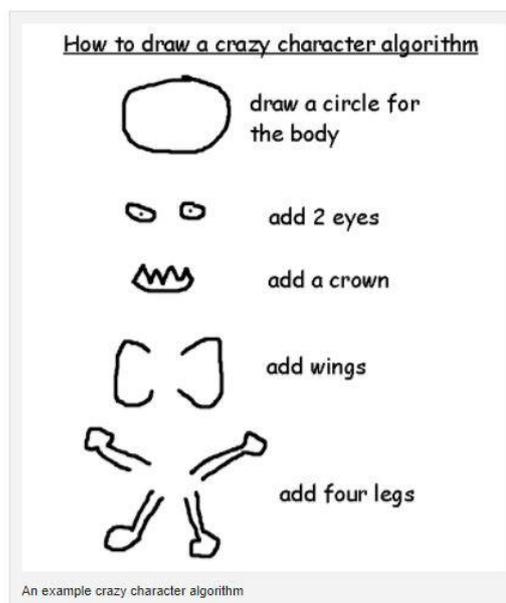


Figure 2. "Ingredients" of the crazy character

All the participants drew different versions for this crazy character by using the "ingredients" in Figure 2. Next, they wrote their instructions for others to draw the same crazy character by just reading the instructions without knowing what was being drawn by that person.

3.5 Unplugged Activity 3

Unplugged Activity 3 showcased how a computer scientist could abstract problems and solutions. It was modified from "Graph Paper Programming Unplugged Activity 4" by Code.org.

This time, the "ingredients" were not being only given for the problems, the solutions were also being abstracted to the simplest way.

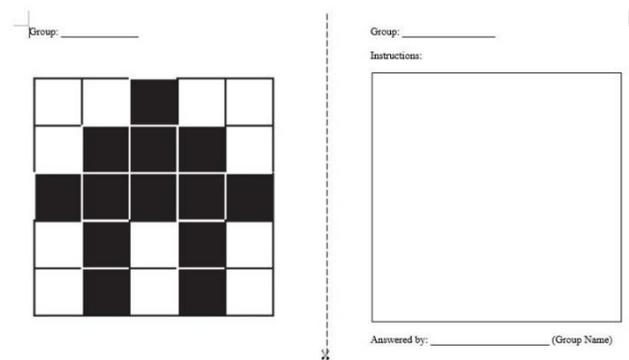


Figure 3. Unplugged Activity 3

PROGRAMMING KEY

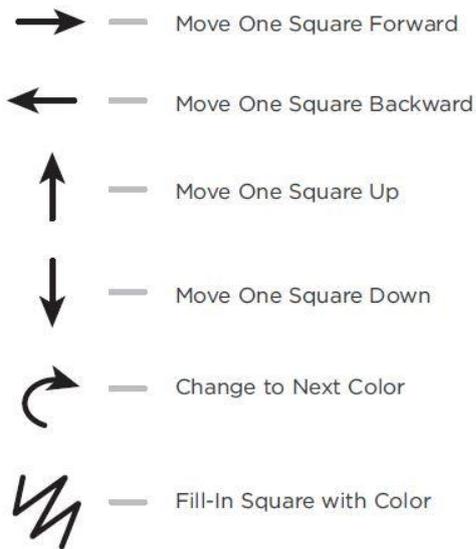


Figure 4. Simplified solutions

All the three Unplugged Activities demonstrated different levels of problem skills, and how this CT could simplify (abstract) the solution (algorithm) by eliminating human errors.

3.6 Post-training assessment

After all the three Unplugged Activities were completed, the participants went through the Computational Thinking training which was designed by the author.

A post-training survey was conducted to assess on participants’ view and understanding on Computational Thinking: Their view on CT as a deeper level of problem solving skills, and does CT help them better observe how students learn and communicate with others.

4. RESULTS

We demonstrated our finding on the participants’

- i. understanding on CT,
- ii. ability to view from the students’ perspective,
- iii. problem solving skills.

4.1 Understanding on CT

Prior to the training, all the participants were asked on their understanding towards this Computational Thinking. Table 1 shows the answers from the 19 (out of 21) participants.

Table 1. Pre-training – Understanding on Computational Thinking.

Question: In your opinion, what is Computational Thinking (CT)?	
Partici pant	Answer
1	Problem solving technique which follows specific steps and procedures / guidelines.
2	I have less exposure on CT, but in general I think it is the way of how we view and solve problems.

3	Logical thinking with use of technology in solving real life problems.
4	Student able to generate new idea using several process and produce the idea using new era computing.
5	Thinking about how to use technology effectively to solve problem.
6	Students know how to use the technology efficiently or in other words, use it with wisdom and have the knowledge on how the process happen (to solve problems using the computer or technology).
7	Breaking down a big problem to smaller pieces and then combine them to get the final solution.
8	Problem solving techniques in CS.
9	Sorry, not really sure. Its may about logic thinking as a coding in computer programming.
10	Using technology as a problem solver.
11	CT (skills and ways of thinking) can be used to support problem solving process when writing computer programs.
12	CT is a set of processes for solving problems in logical way.
13	To provide solution to problem using computer.
14	CT is cognitive an thought processes involved in formulating problems and solutions so that the solutions of the problems could be represented in a form that can be effectively carried out by an information-processing agent.
15	Not sure
16	Logical thinking about how to solve problems.
17	Mind thinking to be as computer thinking.
18	Method used by computer scientist to solve problems.
19	Computational Thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer-human or machine.

Table 1 shows that most of the participants had no prior knowledge on CT. They thought it was the use of technology or computing in solving problems.

4.2 Ability to view from the students’ perspectives

After the participants had gone through the three Unplugged Activities, the participants eventually realized they would definitely need to rethink how they should conduct their teaching lessons from all the students’ perspective.

Table 2. Pre-training self-assessment: Understanding the students’ learning perspective.

Question: During the teaching & learning process, I am able to see from students’ perspective. This is how I do it:	
Partici pant	Answer
N	(Before)

	<p><i>Observing their learning patterns and how they answer assessment questions,</i></p> <p>(After)</p> <p><i>While going through the exercises and the 6 concepts of CT, I realize that as an educator, I should know the prior knowledge that the students have, so that the activities created for them are suitable and they'll be able to gain the CT skills.</i></p>
M	<p>(Before)</p> <p><i>I set my mind that I am a students which is new to the subject.</i></p> <p>(After)</p> <p><i>Don't expect students to think like we think.</i></p>
Q	<p>(Before)</p> <p><i>Observe and evaluate the student performance (results & responses from the students when I asked questions to them)</i></p> <p>(After)</p> <p><i>I have learned that we cannot feel frustrated if students are unable to follow all of our instructions. It is because at sometimes we must see from their perspectives too in order to be get mutual understanding.</i></p>
I	<p>(Before)</p> <p><i>Yes?</i></p> <p>(After)</p> <p><i>Instruction must be clear.</i></p>
	<p>(Before)</p> <p><i>Provide the question to student and ask them to solve it, observe the way how they solve the problem, and then discuss with them if there is any issue.</i></p> <p>(After)</p> <p><i>From today's training, I get to know that the different between instructor and students. Instructor will always think that student understand them, however, that is not 100% true, most of the time, if the instructor did not give them the evaluation, such as provide the exercise, and ask them to try, at the end, the student will totally learnt nothing, as they never try and know their mistake. Thus they don't have chance to correct it.</i></p>
W	<p>(Before)</p> <p><i>Through arguments in their reports</i></p> <p>(After)</p> <p><i>What I know, what my colleagues know, and what the trainer knows is quite different. Therefore, we can not set standards that are too high at first, where we must allow the learning process to change positively over time.</i></p>
S	<p>(Before)</p> <p><i>Through arguments in their reports</i></p> <p>(After)</p> <p><i>What I know, what my colleagues know, and what the trainer knows is quite different. Therefore, we can not set standards that are too high at first, where we must allow the learning process to change positively over time.</i></p>

"Human mind can make assumptions and prediction, but it can get tired and confused."

"Human can predict and make assumption. But they also tend to forget and have negative feelings."

"Human is able to guess, assume and predict when they start to propose a solution. However, they will feel frustrated and sometimes get easily annoyed if they cannot solve the problem using the proposed solution."

"Strength of human mind-can guess, predict, assume, has prior knowledge, and can judge. While, weakness of human mind- get tired and easily disrupted."

"Strength: Human can think wisely, and then improvise. Moreover, human can do the logical reasoning, they able to identify the correct or wrong. However for human's weakness, is they have feeling, have emotion, and sometimes, the bad emotion, will causing them to make the wrong decisions."

"The strength of human mind is able to think logically and creatively while the weakness is lack of focus and concentration."

"Human can do reasoning, they tend to make guess, predict, tired, confused based on their old information."

4.3 Problem Solving Skills – CT is a unique school of thought

All the participants were encouraged to rate their own problem solving skills prior to the CT training. During the training, they would be able to see the three different levels of thought process from the three Unplugged Activities. From these hands-on activities, they eventually realized they still needed to improve their problem solving skills.

These participants provided their thoughts on their own problem solving skills after three Unplugged Activities. It could be summarized into a few format:

"I am more clear about how I think/thinking process while solving problems during training activities."

"I realized my problem solving skills was just moderate."

"I realized my problem solving skills improved after learning all the 6 CT concepts."

"I realized my problem solving skills need to be improved."

"I realized my problem solving skills was just average."

"I realized my problem solving skills can be enhanced by incorporating the computational thinking skill. The skill that I learned the most is abstraction, which is learnt to identify the important features when solving the issue, and also must first to break the problem into the small part which can be manageable."

All the participants were questioned on whether this CT is an important skill to be taught to their students and a high majority of the participants fully agreed that this CT would greatly to help prepare the students to contribute new solutions to the seemingly impossible problems (Figure 5).

The participants were conscious on the strength and weakness of the human mind especially in this area of T&L process after they had gone through Unplugged Activity 1.

"Human mind cannot process too many instruction / complex."

Computational Thinking Skills help prepare students to contribute new solutions to seemingly impossible problems.

21 responses

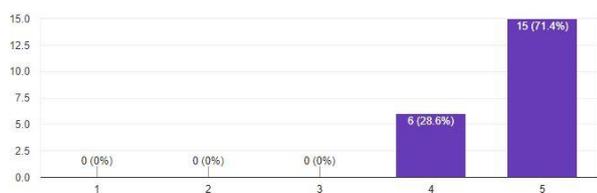


Figure 5. Computational Thinking is important

5. DISCUSSION

The ultimate results showed that Unplugged Activity 1 had successfully created awareness and brought it to the conscious level of all the participants which they could recognize this deficit. They agreed that they had wrongly thought they could easily understand how their students would think but in actual fact there are much room for improvement. This CT has shaped their general perspective on how their students learn after understanding and identifying the ways the human minds work.

Besides this, the results also showed that the participants agreed to the importance of this CT and that it greatly helped to improve their problem solving skills.

6. LIMITATIONS AND FUTURE WORK

The survey and training were conducted entirely in English language. According to the feedback, it showed that the majority of these participants' English level were not at the proficient level. They may have also misunderstood the meaning of some given questions, or were unable to absorb all the information shared during training.

A more carefully planned self-assessment questions for both pre and post trainings should be developed, in order to successfully provoke the participants' thoughts on the core of the questions.

Before we can popularize the CT in Malaysia, we need to create effective awareness to the needs of this CT. It could be a road block for changes to take place if we do not help the educators to unlearn the old concepts, so that they can relearn this CT.

On top of that, the importance of closing the gap between the educators and the learners is very important. It should start from how these educators can view all the students' learning skills from their perspective. From there, these educators could use their CT skills to decompose the lesson towards these students' manageable level, and make learning more fun and achievable.

We need to think of the effective ways to maintain trained educators' thinking pattern, so that they would not fall back to their old patterns too.

7. REFERENCES

- Adams, L. (n.d.). Learning a New Skill is Easier Said Than Done. Retrieved Jan 28, 2018, from: www.gordontraining.com/free-workplace-articles/learning-a-new-skill-is-easier-said-than-done/
- Barefoot CAS. <https://barefootcas.org.uk/programme-of-study/understand-algorithms/ks1-crazy-character-algorithms-activity/>
- Bhanji, F., Gottesman, R., Grave, W.D., Steinert, Y., & Winer, L.R. (2012). The Retrospective Pre-Post: A Practical Method to Evaluate Learning from an Educational Program. *Academic Emergency Medicine*, 189-194. doi:10.1111/j.1553-2713.2011.01270.x
- Domingos, P. (2015) *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York, NY: Basic Books.
- Economic Planning Unit. (2015). *Eleventh Malaysia Plan 2016 – 2020: Anchoring Growth on People*. Putrajaya, WP: Malaysia. Prime Minister's Department.
- CMU. History of Computers. Retrieved Jan 26, 2018, from: <https://www.cs.cmu.edu/~fgandon/lecture/uk1999/history/>
- Kim, S., Chung, K., & Yu, H. (2013). Enhancing Digital Fluency through a Training Program for Creative Problem Solving Using Computer Programming. *The Journal of Creative Behavior*, 47(3), 171-199. doi:10.1002/jocb.30
- Ling, U. L., Saibin, T. C., Labadin, J., & Abdul Aziz, N. (2017). Preliminary Investigation: Teachers' Perception on Computational Thinking Concepts. *Journal of Telecommunication, Electronic and Computer Engineering*, 9, 2-9, 23-29. Retrieved from <http://journal.utem.edu.my/index.php/jtec/article/view/2672>
- Nagy, B., Farmer, J., Quan, B. M., & Trancik, J. E. (2013). Statistical Basis for Predicting Technological Progress, *PLOS ONE*, 8 (2), 1-7. Doi: 10.1371/journal.pone.0052669.
- Project Management Office (2012). *Preliminary Report, Malaysia Education Blueprint 2013-2025*. Malaysia, Ministry of Education.

Teacher's Perceptions and Readiness to Teach Coding Skills: A Comparative Study between China, Finland and Singapore

Chee-kit LOOI^{1*}, Jari MULTISILTA², Longkai WU¹, Pauliina TUOMI¹

¹ National Institute of Education, Nanyang Technological University, Singapore

² Tampere University of Technology, Finland

cheekit.looi@nie.edu.sg, jari.multisilta@tut.fi, longkai.wu@nie.edu.sg

ABSTRACT

While many countries have recognized the importance of computational thinking and coding skills and are implementing curricular changes to introduce coding into formal school education, a necessary and critical success factor involves the preparation of and support for teachers to teach coding. Thus, understanding the perceptions of teachers towards coding is most important, together with knowing the kinds of support they received, and their readiness and challenges to teach. The purpose of the current study is to compare teachers' attitudes towards the importance of ICT skills and coding skills in Finnish, Chinese and Singapore K-12 schools. The findings indicate that Singapore and Finnish teachers believe that coding is useful even if students will not work in ICT jobs while Chinese teachers are undecided. China and Singapore have more positive views towards how to prepare for future-ready learners.

KEYWORDS

computational thinking; coding skills; 21st century skills; primary school; comparative research; attitudes

1. INTRODUCTION

Countries and regions around world, such as Australia, New Zealand, United States, United Kingdom, South Korea, Finland, China and Singapore, have recognized the importance of coding. They are taking rapid measures to introduce it through all levels of the school curriculum. Both Finland, China and Singapore have to date revised national standards and curriculum to focus learning goals on higher-order thinking, inquiry, and innovation, as well as the integration of technology to the curriculum. In these countries, the need for educating students in 21st century skills is commonly acknowledged. These countries (Shanghai region for China) have also been top performers in PISA rankings.

The purpose of the current study is to compare teachers' attitudes towards the importance of 21st Century skills, especially computational thinking (CT) and coding skills in Finland, China and Singapore, in K-12 schools. Specifically we aim to compare teacher's attitudes towards the importance of teaching coding skills already in basic education, the importance of 21st Century Skills in students' future jobs, and preparing students for the digital century. The findings and results of comparative education studies are valuable resources also for the administration of education systems and is one of the main reasons this approach was chosen for this study.

2. ICT AND CODING POLICIES IN THE 3 COUNTRIES

We first provide a backdrop of policies regarding ICT use and teaching of coding the three countries.

Finland

The teaching of ICT started in Finland in 1980s, first in high schools. Official reports and curriculum projects stated clearly that students should learn the basics of this new literacy. However, software support was weak and there was not much in-service teacher training in computing. In secondary schools, the actual subject of ICT was brought into the curriculum between 1987 and 1988, as an optional subject. A few years later, ICT was no longer taught as an individual subject, and ICT skills were integrated into other subjects (Vahtivuori-Hänninen & Kynäslähti 2012).

Since fall 2016, coding is a mandatory, cross-curricular activity that starts from first year of school and spans both primary and lower secondary education. Finland has outlined that coding is one of the learning skills – just like reading, writing, counting and drawing. The Finnish Ministry of Education has outlined that ICT skills, and coding in particular, is a fundamental part of the Finnish National Core Curriculum (FNCC) from 2016 (FNBE, 2016). It is still not an independent subject, but it is integrated into other subjects. The FNCC defines several transversal skills that should be taught and learned in every subject. ICT competence is among these transversal competencies. The FNCC states that pupils should work with digital media and age-appropriate programming tasks. Key content areas related to the objectives of mathematics in grades 1 and 2 state that, “the pupils began familiarizing themselves with the basics of programming by formulating and testing step-by-step instructions” thus supporting the development of logical thinking and problem solving.

China

Computer technology has been utilized in Chinese education since the 1980's (Mok & Leung 2012) According to Niemi and Jia (2016), the growing popularity of the Internet and communication technology from the 1990's onwards brought a wider concept of ICT, which was then introduced into China and Chinese education (Niemi and Jia 2016, 9). In 2010, a national plan for educational reform and development was issued by the central government. It declared that ICT will have a revolutionary impact on education (MoE China 2010). Since that time, there has been a steady increase of government expenditure on education, and vast investment from central and provincial governments has gone to the application of ICT in education (Niemi and Jia 2016, 9; Han and Ye 2017).

In 2016, the National People's Congress approved the 13th five-year plan for national economic and social development which stressed to enhance the educational level of all people and to promote modernization of education. The concrete approaches detailed in this plan include the development of online education and distance learning, the integration of all kinds of digital resources and their service for society as a whole, and the deep integration of ICT with teaching and learning (National People's Congress China, 2016). Based on Jia and Niemi (2016), "The purpose of ICT integration into ordinary teaching and learning is to cultivate students' basic knowledge, skills, and literacy in the information era, to foster their creativity, and to prepare them for the future workplace" (Jia and Niemi 2016, 315).

A new round of high school curriculum reform program has been announced in 2016 and enacted from 2017, which takes CT as one of the four core elements of the discipline of information technology. The move indicates that CT has been given more importance at the national curriculum level which will influence the enactment of new curriculum standards, composition of new teaching materials and guidance of new college entrance examination.

Singapore

Singapore is a small city-state with key national focus on developing human capital, its ICT in Education policies are formulated with the goals of preparing its student citizenry for the knowledge-based economy, and to enhance the learning experiences of students in schools. Since 1997, the government has launched four Masterplans for ICT in Education to equip students with ICT-enhanced approaches to learning.

In 2014, Singapore launched the Smart Nation Programme which is a nationwide effort to harness technology in the business, government and home sectors for improving urban living, building stronger communities, growing the economy and creating opportunities for all residents to address the everchanging global challenges (Smart Nation, 2014). One of the key enablers for the Smart Nation initiative is to develop computational capabilities. Programmes are implemented to introduce and develop CT skills and coding capabilities from pre-school children to adults. To develop CT capabilities and support the Smart Nation initiative, several programmes have been implemented to introduce and develop CT skills and coding capabilities in every Singaporean, from pre-school children to adults (Seow, Looi, Wadhwa, Wu & Liu, 2017).

Singapore's approach is to provide opportunities for students to develop their interests in coding and computing skills through touchpoint activities at various ages. Computing and CT skills are introduced to the children that are age-appropriate and engage them in learning. Children progressively develop interest and skills leading them to offer Computing as a subject for grade levels 9 and 10.

There are major differences between China, Finland and Singapore in terms of their respective populations, languages, history, cultural roots, and educational systems (Jia and Niemi 2016, 318). However, when discussing new ways to teach and learn, these countries face similar opportunities and challenges. In these countries, ICT and

new learning environments are perceived as tools for teaching and learning. These countries emphasize that new digital tools and materials should be pedagogically relevant and that teachers need support and training to learn how to use them.

3. DESIGN OF SURVEY

The survey is designed based on three major guiding questions: 1) What are the perceptions of teachers on ICT use in schools? 2) What are the readiness levels of teachers for teaching coding skills? 3) What are the perceptions of teachers towards teaching coding skills? It comprises 74 questions in total, including 5 questions on teacher profiles, 14 questions on ICT use, 14 questions on teachers' readiness to teach coding skills, and 41 questions on teachers' perceptions and attitudes related to coding skills. The survey questions on perceptions and readiness use a 5-Likert scale (1-Strongly disagree, 2-Disagree, 3-Undecided, 4-Agree, 5-Strongly agree).

4. FINDINGS OF SURVEY

In total there were 702 respondents, 406 from China, 143 from Singapore and 153 from Finland. The teachers from China are all from the Shanghai region. According to Chi-Square test, the gender distribution in the data is statistically different, $\chi^2(2) = 21.26, p < .001$. The majority of the respondents were female teachers (79.4%). In China, there were 84.2% female teachers and in Finland 78.9%. In Singapore, 65.4% of all respondents were female.

According to Chi-Square test, in the age distribution of the respondents there is a significant difference, $\chi^2(16) = 212.04; p < .001$. Respondents in China are younger compared to Singapore and Finland. From an one-way ANOVA test, the teaching experience in school years in Finland, Singapore and China is not statistically different, $F(2,696) = 4.48, p = .012$.

According to one-way ANOVA test, the school level in Finland, Singapore and China there is a significant difference, $F(2,633) = 214.21, p < .001$. From Finland, there were no respondents from early childhood teaching, whereas from China 10.2% of all respondents were in early childhood schools. In Finland, 36.6% of all respondents were in upper primary schools (0% in China). Almost all respondents from Singapore were from secondary school (99.3%).

4.1. Coding skills for all or for some

The question posed is: Coding skills should be taught only to students that are aiming to work on the field of information technology (1 Strongly disagree, 5 strongly agree). The result indicates that there is a significant difference between China, Singapore and Finland. Finnish teachers ($M = 2.46, SD = 1.34$) think that coding skills are needed also for those who are not aiming to be professional programmers while Chinese teachers are undecided ($M = 3.13, SD = 1.21$). The teachers in Singapore ($M = 2.46, SD = 1.13$) think similarly as the Finnish teachers, $F(2) = 37.73, N = 701, p < .001$.

According to one-way ANOVA, there were no differences between the teachers in different age groups, $F(8) = 2.03, p = .041$, or gender, $F(1) = 3.06, p = .080$. In addition, there were no differences between the teachers who had different

amounts of school experience as a teacher, $F(6) = 1.22, p = .292$.

4.2. Best method to learning coding skills

On the question on what is the best method to learn coding skills (1 Strongly disagree, 5 strongly agree), teachers in all countries agree that coding is learned best by writing the code, with visual programming environments, building robots and outside school clubs. Teachers in China agree that coding is also best learned at school with the teacher's guidance, but Finnish teachers are undecided. The difference is statistically highly significant, $F(2) = 80.50, p < .001$. Teachers in China agree that coding is also best learned from books and dedicated websites, but Finnish and Singapore teachers are undecided. The difference is statistically highly significant, $F(2) = 76.58, p < .001$.

		M	SD	N	F(2)	Sig.
At school, with the teacher's guidance	China	4.23	.71	393	80.50	.000
	Finland	3.34	.84	153		
	Singapore	3.6	.97	138		
From books and dedicated websites	China	4.00	.78	393	76.58	.000
	Finland	3.12	.85	152		
	Singapore	3.39	.83	138		
By actually writing/rehearsing the code	China	3.99	.82	391	3.55	.029
	Finland	3.86	.91	153		
	Singapore	4.13	.88	138		
Through visual and graphical coding languages like Scratch	China	3.98	.82	394	2.64	.072
	Finland	3.87	.75	152		
	Singapore	3.81	.78	138		
Through building and programming robots	China	3.85	.85	396	2.01	.135
	Finland	3.95	.80	152		
	Singapore	3.76	.76	138		
In informal activities such as coding clubs, and other outside of school events	China	3.98	.79	394	4.09	.017
	Finland	4.11	.70	153		
	Singapore	3.86	.75	138		

When the gender is used as a factor in the one-way ANOVA, there is a statistical difference only in the item "by actually writing/rehearsing the code", $F(1) = 10.98, p = 0.001$. Male teachers agree that coding should be learned by writing the code ($M = 3.93, SD = .85$) more compared to female teachers ($M = 4.20, SD = .86$).

In addition, according to one-way ANOVA, there is a statistically significant difference in the item "at school, with the teacher's guidance", in different age groups, $F(8) = 4.26, p < .001$. In general, teachers under 45 more that the coding should be learned at school, with the teacher's guidance than the teachers who are over 46.

In addition, according to one-way ANOVA, there is a statistically significant difference in the item "from books and dedicated websites", in different age groups, $F(8) = 4.31, p < .001$. In general, teachers under 45 more that the coding should be from books and dedicated websites than the teachers who are over 46. When the teacher's school experience is used as a factor in the one-way ANOVA, there are no statistically significant differences between the groups.

4.3. ICT used by students in schools

The question posed is: How often your students use the following technologies in your classroom? A four point scale was used, rated from 1 (not at all), 2 (once a month), 3 (once a week) to 4 (daily). The hypothesis we had is: The amounts of use of technologies in the classroom does not differ in China, Finland and Singapore.

The result indicates that computers are used more in China ($M = 3.23, SD = 1.11$) compared to Finland ($M = 2.74, SD = .92$) or Singapore ($M = 2.28, SD = 1.01$). The difference is statistically highly significant, $F(2) = 43.96, p < .001$.

Internet is used in Singapore ($M = 2.69, SD = .95$) less than in China ($M = 3.21, SD = 1.05$) or Finland ($M = 3.14, SD = .85$). The difference is statistically highly significant, $F(2) = 14.68, p < .001$.

Digital cameras and videos are also used more often in China ($M = 2.66, SD = 1.09$) compared to Finland ($M = 1.81, SD = .83$) and Singapore ($M = 1.83, SD = .90$). The difference is statistically highly significant, $F(2) = 57.34, p < .001$. Educational applications and games are used in Singapore ($M = 1.96, SD = .89$) less than in China ($M = 2.71, SD = 1.12$) or Finland ($M = 2.54, SD = .89$). The difference is statistically highly significant, $F(2) = 126.57, p < .001$. Notebooks and tablets and mobile phones are used in classroom similar amounts in both countries.

		M	SD	N	F(2)	Sig.
Desktop/laptop computers	China	3.23	1.11	382	43.96	.000
	Finland	2.74	.92	152		
	Singapore	2.28	1.01	138		
Notebooks/tablets	China	2.19	1.26	390	4.07	.017
	Finland	2.35	1.04	152		
	Singapore	1.95	1.01	133		
Internet	China	3.21	1.05	385	14.68	.000
	Finland	3.14	.85	152		
	Singapore	2.69	.95	138		
Educational applications/games	China	2.71	1.12	393	26.57	.000
	Finland	2.54	.89	152		
	Singapore	1.96	.89	134		
Digital cameras/videos	China	2.66	1.09	386	57.34	.000
	Finland	1.81	.83	149		
	Singapore	1.83	.90	136		
Digital projectors/interactive whiteboards	China	2.98	1.16	389	38.49	.000
	Finland	2.74	1.31	153		
	Singapore	1.93	1.20	134		
Mobile phones	China	2.35	1.30	392	4.34	.013
	Finland	2.69	1.08	151		
	Singapore	2.43	.97	136		

When the teacher's age is used as a factor in the one-way ANOVA, there is a statistically significant difference only in the use of digital cameras and digital videos in the classroom, $F(8) = 4.74, p < 0.001$. The teachers in the age groups 20 to 25 ($M = 2.68$) and 40 to 45 ($M = 2.65$) use digital cameras and videos the most, whereas the teachers from 60 to 65 use the least ($M = 1.67$). There are no such differences in the use of other technologies.

4.4. Teachers' levels of programming skills

The subscale had 2 questions (Cronbach alpha = 0.868): How would you evaluate your own competence on the following skills?

- Programming languages (e.g. Python)

- Visual coding software (e.g. Scratch)

The results suggested that there was no difference in the programming skills of the teachers for China ($M = 3.45$, $SD = 1.90$), Singapore ($M = 3.93$, $SD = 2.56$) and Finland ($M = 3.65$, $SD = 1.87$), $F(2,684) = 2.93$, $p = 0.054$.

According to Kruskal-Wallis test, there is statistically highly significant difference between the male ($M = 4.74$, $SD = 2.48$) and female ($M = 3.29$, $SD = 1.82$) teachers in the programming skills, $H(1) = 44.00$, $p < .001$, $N = 675$. From an one-way ANOVA, there is not a significant difference in the programming skills between the teachers in different age groups, $F(8) = 2.30$, $p = .019$. In general, in the scale from 2 to 10 (a sum of two 5 point Likert items), the programming competence of the teachers is low ($M = 3.59$, $N = 677$, $SD = 2.05$).

4.5. Attitudes towards the importance of the future skills in students' future jobs

The subscale had 8 questions (Cronbach alpha = 0.908): The following skills have a great importance in your students' future jobs: logical thinking, problem solving, creativity, programming, social and collaboration skills, entrepreneurialism, language and communicational skills, analytical thinking.

The results show that there was statistically highly significant difference in the attitudes towards the importance of futures skills of the teachers for China ($M = 37.18$, $SD = 4.03$), Singapore ($M = 35.89$, $SD = 3.65$) and Finland ($M = 35.04$, $SD = 3.96$), $F(2,673) = 17.68$, $p < .001$. The Chinese teachers attitudes towards the importance of future skills are more positive compared to the Singapore and Finnish teachers attitudes.

According to Kruskal-Wallis test, there is statistically significant difference between the attitudes towards the importance of futures skills between male ($M = 35.88$, $SD = 4.20$) and female ($M = 36.60$, $SD = 4.00$) teachers in the skills, $H(1) = 4.49$, $p = .034$, $N = 664$. The female teachers attitudes towards the importance of futures skills is more positive than the male teachers attitudes.

In addition, according to one-way ANOVA, there is a statistically significant difference in the attitudes towards the importance of futures skills in different age groups, $F(8) = 4.22$, $p < .001$. In general, teachers under 45 have a more positive attitude towards the importance of futures skills than the teachers who are over 45.

4.6. Attitudes towards teaching future skills in basic education

The subscale had 8 questions (Cronbach alpha = 0.884): The following skills should be taught to everyone in primary schools: logical thinking, problem solving, creativity, programming, social and collaboration skills, entrepreneurialism, language and communicational skills, analytical thinking.

We found that there was statistically highly significant difference in the attitudes towards the teaching the futures skills already on basic education of the teachers for China ($M = 36.19$, $SD = 5.37$), Singapore ($M = 33.98$, $SD = 4.01$) and Finland ($M = 34.61$, $SD = 4.25$), $F(2,682) = 13.06$, $p <$

.001. The Chinese teachers' attitudes towards the importance of teaching the future skills already in basic education are more positive compared to the Singapore and Finnish teachers attitudes.

According to Kruskal-Wallis test, there is no difference between the attitudes towards the teaching the future skills between the male ($M = 34.96$, $SD = 4.997$) and female ($M = 35.54$, $SD = 4.97$) teachers, $H(1) = 1.52$, $p = .217$, $N = 673$. In addition, according to one-way ANOVA, there is a statistical difference in the attitudes towards the teaching the future skills in different age groups, $F(8) = 3.04$, $p = .002$. In general, teachers under 45 have a more positive attitude towards the teaching the future skills than the teachers who are over 46.

4.7. Attitudes towards the technological change

The subscale had 4 questions (Cronbach alpha = 0.712):

- I believe that almost all businesses will be computerized in the future
- I have a good understanding of the effects of technology on the environment, society, and individuals.
- I think most well-paying technology jobs will require workers who are highly-skilled.
- I think that most jobs in the future that require the use of a computer will require strong thinking skills.

The results show that there was a statistically highly significant difference in the attitudes towards the technological change of the teachers for China ($M = 17.45$, $SD = 2.58$), Singapore ($M = 16.46$, $SD = 1.95$) and Finland ($M = 14.62$, $SD = 2.30$), $F(2,696) = 77.22$, $p < .001$. The Chinese and Singapore teachers' attitudes towards the technological change are more positive compared to the Finnish teachers attitudes.

According to Kruskal-Wallis test, there are no statistical differences between the genders in the attitudes towards the technological change, $H(1) = 1.65$, $p = .199$, $N = 683$.

In addition, according to one-way ANOVA, there is a statistically significant difference in the attitudes towards the technological change in different age groups, $F(8) = 3.77$, $p < .001$. In general, teachers under 45 have a more positive attitude towards the technological change than the teachers who are over 46.

5. DISCUSSION

5.1. Differences between countries

There was not a significant difference in the programming skills of the teachers when we examined both the scripting languages and visual programming languages together. However, the level of programming skills with Python or similar scripting languages was quite low in Finland ($M = 1.58$, $SD = 0.923$) and China ($M = 1.66$, $SD = 0.940$). In Singapore, the programming skills with Python or similar languages level was higher ($M = 2.06$, $SD = 1.382$). In contrast, the skills for using visual programming environments were higher in Finland ($M = 2.08$, $SD = 1.097$) compared to China ($M = 1.79$, $SD = 1.050$) and Singapore ($M = 1.87$, $SD = 1.260$). In the open-ended question, several teachers from Finland and China said that coding is a totally unknown area to them.

In general, Chinese and Singapore teachers' perceptions of their ICT skills are higher compared to the Finnish teachers. The Chinese teachers' attitudes towards the importance of teaching the future skills in basic education and the importance of role the future skills in their students' future jobs are more positive compared to the Finnish teachers. In addition, the Chinese teachers' attitudes towards the technological change are more positive compared to the Finnish teachers' attitudes.

Based on our study, the Chinese and Singapore teachers' perceptions towards the usefulness of ICT in the classroom and school ICT support are more positive compared to the Finnish teachers' perceptions. There are differences in the ICT and programming skills of male and female teachers. In general, male teachers evaluate their ICT and programming skills higher than female teachers. In addition, there is statistically significant difference in the attitudes towards the importance of the future skills in students' future jobs between male and female teachers. The female teachers' attitudes are more positive.

5.2. Differences between genders

Based on our data, there is no gender difference on teachers' perceptions on to whom should be taught coding skills. However, when we asked what the best method to learn coding skills is, there was a difference between male and female teachers. Male teachers agree that coding should be learned by writing the code ($M = 3.93$, $SD = .85$) more compared to female teachers ($M = 4.20$, $SD = .86$).

5.3. Differences between the age groups

There were no difference between the age groups on the item "Coding skills should be taught only to students that are aiming to work on the field of information technology". However, when asked about what is the best method to learn coding skills, teachers under 45 think that the coding should be learned at school, with the teacher's guidance compared to the teachers who are over 45. In general, teachers under 45 think that the coding should be learned from books and dedicated websites compared to the teachers who are over 46 who are less likely to think so.

5.4. Differences between perceptions of computing for all or for some

Singapore and Finland teachers believe that coding is useful even if it is not for ICT jobs; China teachers are undecided.

6. SUMMARY

Teaching coding skills does not happen without the teacher. It is important that teachers are educated, guided, and supported at a practical level to meet the requirements of the coding skills in the curriculum. Many countries are including 21st century skills, computational thinking, and coding skills, as a part of the curricula, but many countries are lacking, at the national level, official and adequate education and training of the teachers on how to implement coding-based activity into their school work.

Singapore and Beijing teachers' preparedness to use ICT is high, compared with Finland. Singapore and Finland teachers believe that coding is useful even if it is not for ICT jobs; Beijing teachers are undecided. Singapore and Finland

have more positive views towards how to prepare future-ready learners

Chinese and Singapore teachers' attitudes towards the importance of teaching future skills already in basic education are more positive compared to the Finnish teachers' attitudes. The Chinese and Singapore teachers' attitudes towards the importance of teaching future skills in basic education, and the importance of the role the future skills will play in their students' future jobs are more positive compared to Finnish teachers. Additionally, the Chinese and Singapore teachers' attitudes towards technological change are more positive compared to Finnish teachers' attitudes.

One of the most striking findings that concern all three countries is the fact that the majority of the teachers in all three countries are not yet competent in any coding languages. While this result is to be expected, that teacher educators cannot expect teachers to effectively teach 21st-century information and media literacy skills that they themselves lack (Fry and Seely 2011, 217). This particular finding clearly suggests adding basic coding skills as a part of the teacher training and in-service, professional development, but also not forgetting the other aspects of teaching the 21st century skills as well. According to Lambert and Gong (2010), there "exists a critical need for suitable curriculum materials to train pre-service and in-service teachers in 21st century concepts related to pedagogy, content, and technology" (Lambert and Gong 2010, 67).

Teachers in all countries agree that coding is learned best by writing the code, with visual programming environments, building robots, and through participation in outside school clubs. Teachers in China and Singapore agree that coding is also best learned at school with the teacher's guidance and from books and websites, but Finnish teachers are undecided. The Chinese teachers consider all presented methods as potential for learning coding. The study indicates that Finnish teachers favour the active learning methods (writing the code in a programming environment, by building robots, and learning in informal learning environments).

The lack of programming and computer education at K–12 level is increasingly recognized as a serious issue in many Western countries (Dagiene et al. 2014; Guerra et al. 2012). Dagiene et al. (2014) states that, "although informatics has been taught as a subject in many European countries as early as in the 1970's, many of these efforts were dropped for various reasons" (Tuomi, Multisilta, Saarikoski, & Suominen 2017, 13). As a result, students graduate from secondary school with a lot of experience using computers and software, but they do not have computational thinking and coding skills, and do not understand the basic principles of how computers and networks operate (Dagiene et al. 2014). This is why it is important to obtain information relating to best practices of having coding as a subject in schools. The best practices could contribute to the modernisation of education and training systems. The results obtained in this study benefit the school principals, teachers, and educational policy-makers. In all, computational thinking and coding skills are challenges that many countries

and schools face. New research that results in providing functional guidelines for teachers, as well as students, to teach and learn coding skills, contributes to the creation of high-quality schools of the future (Tuomi, Multisilta, Saarikoski, & Suominen 2017, 13).

Caveats of this study include: Small sample size from each country, and teachers are not from equivalent school levels (early childhood, primary and secondary from each country). For future research, it is planned to gather similar data from other Asian and European countries and regions such as Hong Kong, the Netherlands, South Korea, Taiwan and USA in order to execute more comparisons and cross analysis between participating countries.

7. REFERENCES

- Christensen, R., and Knezek, G. 1999. "Stages of adoption for technology in education." *Computers in New Zealand Schools*, 11(3), 25-29.
- Dagiene, V. Mannila, L. Poranen, T., Rolandsson, L., and Söderhjelm, P. 2014. "Students' performance on programming-related tasks in an informatics contest in Finland, Sweden and Lithuania." In *Proceedings of the 2014 conference on Innovation; Technology in computer science education, ITiCSE '14*, 153–158. New York, USA, 2014: ACM.
- FNBE (Finnish National Board of Education). 2016. *National Core Curriculum for Basic Education [National Core Curriculum of Basic Education 2014]*. Retrieved from, http://www.oph./download/163777_perusopetuksen_opetussuunnitelman_perusteet_2014.pdf
- Fry, S. and Seely, S. 2011. "Enhancing Preservice Elementary Teachers' 21st Century Information and Media Literacy Skills". *Action in Teacher Education*, 33:2, 206-218, DOI: 10.1080/01626620.2011.569468
- Lambert, J. and Gong, Y. 2010. "21st Century Paradigms for Pre-Service Teacher Technology Preparation". *Computers in the Schools*, 27:1, 54-70. <http://dx.doi.org/10.1080/07380560903536272>
- Jia, J. and Niemi, H. 2016." In search of the future of educational challenges in the Chinese and Finnish context." In *New Ways to Teach and Learn in China and Finland*, edited by Niemi, H., and Jia, J. Bern, Switzerland: Peter Lang D.
- Mok, K. H. and Leung, D. 2012. "Digitalisation, educational and social development in Greater China". *Globalisation, Societies and Education*, 10:3, 271-294, DOI:10.1080/14767724.2012.710118
- National People's Congress China. 2016. *The thirteen Five-Years Plan*. Retrieved from http://www.sh.xinhuanet.com/2016-03/18/c_135200400_2.htm
- Niemi, H., and Jia, J. 2016. "What are the new ways to teach and learn in China and Finland?" In *New Ways to Teach and Learn in China and Finland*, edited by Niemi, H., and Jia, J. Bern, Switzerland: Peter Lang D.
- Smart Nation. (2014). *Why Smart Nation*. Retrieved Feb 13, 2017, from <https://www.smartnation.sg/about-smart-nation>
- Tuomi, P., Multisilta, J., Saarikoski, P., and Suominen, J. 2017. "Coding skills as a success factor for a society." *Education and Information Technologies*. Springer US.
- Vahtivuori-Hänninen, S., and Kynäslahti, H. (2012). "ICTs in a school's everyday life." In *Miracle of education: The principles and practices of teaching and learning in Finnish schools*, edited by H. Niemi, A. Toom, and A. Kallioniemi, 237–248. Sense Publishers.

“It Opens Up a New Way of Thinking, but...”: Implications from Pre-Service Teachers’ Introduction to Computational Thinking

Yu-hui CHANG*, Lana PETERSON

Department of Curriculum & Instruction, Learning Technologies Program,
University of Minnesota, Twin Cities, U.S.
chan1173@umn.edu, pete6118@umn.edu

ABSTRACT

The purpose of this study is to investigate how pre-service teachers perceive and conceptualize computational thinking (CT) concepts within K-12 education. We conducted a pilot case study that was situated in a teacher technology licensure course in the United States. After the CT exposure through a hands-on exploration of programming and robotics as well as an extension research activity, forty-four pre-service teachers’ learning artifacts were collected for a content analysis.

In the initial findings, we found that pre-service teachers were trying to understand practical examples of CT, were inspired by the social justice issues related to computing, and shared CT is in alignment with their educational beliefs. Though a conceptual change of CT occurred among pre-service teachers, there were assumptions and concerns among the pre-service teachers about its application in the classroom.

KEYWORDS

Computational Thinking, Teacher Education, Pre-service Teachers, Technology Integration, Professional Development.

1. INTRODUCTION

Computational Thinking (CT) is becoming a fundamental ability to have in the digital age (Barr, Harrison & Conery, 2011). Despite its importance, most pre- and in-service teachers lack the knowledge and ability to purposefully incorporate CT into classrooms (Freeman, Adams Becker, Cummins, Davis, & Hall Giesinger, 2017; Grover & Pea, 2013). The majority of research on training teachers about CT as a concept and how to integrate CT into the curriculum has focused on in-service teacher professional development (Yadav, Gretter, Good, & McLean, 2017).

Enabling a student to become a “computational thinker” has been added to the International Society for Technology in Education’s Standards for Students (ISTE, 2016). In collaboration with the Computer Science Teachers Association (CSTA), ISTE has suggested that teachers cultivate students’ use of CT as a process for problem solving, algorithmic thinking, and solution building. Shifting CT to a central role within education requires a comprehensive approach including integrating CT into K-12 pre-service education programs (Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014; Yadav et al., 2017).

As both researchers and teacher educators, we found a lack of CT training within our pre-service education program and wished to explore what could be done to change this within our context. To examine this issue, we provided an

opportunity for pre-service teachers to engage in CT experiences in a teacher-licensure course.

This case study is an attempt to introduce pre-service teachers to CT through hands-on exploration. Our driving research questions for this case study are: (a) how do pre-service teachers conceptualize the role of CT within K-12 education? (b) what are implications for teacher educators to support pre-service teachers’ understanding of CT?

2. SUPPORTING LITERATURE

There are myriad of definitions and approaches to the concept of CT. It has been described as a problem-solving process (Balanskat & Engelhardt, 2015), a “cognitive skill” (National Research Council, 2010), a framework of concepts and capabilities (Barr & Stephen, 2011), and an identity (ISTE, 2016). Google (2015) broke down CT into four components for educators: decomposition, pattern recognition, abstraction, and algorithm design. Educators are encouraged to incorporate the four CT components into teaching, such as having students discover the principles of a pattern within learning materials. Brennan and Resnick (2012)’s definition of CT has three components: (1) computational perspectives- how young people identify with computing participation; (2) computational concepts- vocabulary and skills needed to engage in computing; and (3) computational practices- processes used to work with computers. There are clear overlaps with the different definitions of CT but also unique lenses to the various approaches (Voogt, Fisser, Good, Mishra, & Yadav, 2015).

Moving from “what is CT” to “how to use CT”, we look to Yadav et al. (2014) who provided CT modules in a teacher education program to develop pre-service teachers’ understanding of CT and learn more about their attitude towards the concept. As for professional development in general, previous research (Israel, Pearson, Tapia, Wherfel, & Reese, 2015) identified limited instructional time and lack of technology and support as barriers to integrating CT into classrooms. Their findings indicated that supportive resources such as ongoing professional development and coaching play a vital role in increasing teachers’ ability to integrate CT seamlessly. Their findings also showed that struggling learners, students with disabilities, and low socioeconomic status students benefit from building CT skills.

There are more opportunities than ever for students to experience coding and computational thinking through online platforms such as code.org or new robots and tools designed for the K-12 context (Shellenbarger, 2016). Researchers are concerned that CT skills should not be learned only through separate coding programs but

integrated into core content (Barr & Stephenson, 2011). We need to re-design instructional approaches (e.g., problem-solving) and pedagogical strategies (e.g., group collaboration) in curriculum and interdisciplinary subjects to engage learners in practicing CT (Grover & Pea, 2013; Lye & Koh, 2014). Goode, Margolis, and Chapman's (2012) *Exploring Computer Science* teacher professional development is one exemplar model for helping teachers integrate CT into the core curriculum.

3. METHODOLOGY

3.1. Context

This case study was situated in a 1.5 credit, required course for teacher candidates on technology integration in K-12 education, at a Midwest public university in the United States. This teaching-licensure course had to address ten state standards for effective teaching related to topics such as instructional strategies, assessment, and learning environments. The instructor had complete autonomy over pedagogical approaches and the exact curriculum to address the standards. The blended course met seven times face-to-face with online activities between classes. One of the researchers on this study was also the instructor for three sections of this course during the spring and summer of 2017. Pre-service teachers who completed the course session on CT exploration met the criteria in this study for research. They were asked to share their coursework for research purposes after the end of the course. Participants included forty-four pre-service teachers, 5 male and 39 females, who are majoring in elementary education, special education, or early childhood education.

CT is not a topic that is traditionally addressed in this course but given its increased presence in K-12 education and implications for students we felt it was important to expose pre-service teachers to the concept. One barrier to including CT was the lack of time in an already full curriculum. In this case, CT exposure took place three activities: (1) a 10-min introduction about the coding movement, (2) an hour-long unstructured exploration of hands-on tools and resources, and (3) an online extension activity. To be more specific, during the one hour free exploration, pre-service teachers had access to a Makey-Makey, a Blue-Bot robot, a Dash and Dot robot, an Osmo, an Ozobot and a stations of coding platforms designed for elementary students, such as Code.org and Scratch. The classroom was set up by having each resource displayed at one station around the classroom. Participants were encouraged to interact with at least one resource to help them gain new ideas from hands-on experiences. After the in-class CT exploration, teacher candidates were asked to find one online resource related to CT or coding and share their own reflective ideas about CT. Their responses to this activity were posted in the course's online learning management system. Their responses were visible to their peers but it was not a requirement for them to interact with their peers within the discussion thread.

3.2. Data Analysis

Forty-four reflective posts were collected from teacher candidates. To analyze these posts each researcher independently open coded all of the journals. We then compared our initial codes with each other to find alignment

and missed insights. Next, we collaboratively utilized pattern coding to develop major themes from the data. To enhance internal validity, we continued to member check as we tested the themes against the data. Within this stage, five patterns emerged: (1) CT resources, (2) personal meaning of CT, (3) CT and teachers' expertise, (4) conceptual changes of CT, (5) assumptions of CT. During coding process, the researchers also used the constant comparative method to enhance the validity of results.

4. PRIMARY FINDINGS

The pre-service teachers started to gain awareness of CT through their own educational beliefs and teacher expertise. Among all the resources that they shared in the extension activity to support their understanding of CT, pre-service teachers revealed a need to find concrete teaching examples. This included the desire to explore how CT works in classrooms, subjects, and curriculum by utilizing YouTube to view classroom showcase videos and blogs from Edutopia. They also demonstrated their interest in looking for free online coding platforms as a useful teaching resource and learning environment. Practical resources were the most sought after to translate the experience they had just had into a tangible tool for their future students.

Particularly for the pre-service teachers specializing in early childhood and special education, it was difficult to identify specific resources or address concerns in order to support their students' development. Additionally, we found that pre-service teachers described the nexus of CT with their educational beliefs in creativity and constructivist learning theories. They showcased the role CT could play in content and pedagogical style. For example, some teachers mentioned potential ways to design a group discussion such as "*getting students working together to figure out how to create a story using critical thinking and problem solving skills*" (PT#202) and another shared an idea for formative assessment "*track students' progress as well as view student solutions for each level*" (PT#103).

The pre-service teachers communicated a variety of perceptions of the importance of CT. The most influential factor on their conceptualization seemed to be the implication on humanity. Some of the participants talked about the intersection of computing with social justice issues such as gender, race, and socioeconomic status. For them, seeing non-profits focused on addressing representation with computing gave the topic importance. Giving their future students opportunities to succeed and career options was a motivating factor to integrate CT.

During their extension activity research, many of the pre-service teachers referenced campaigns and non-profits such as Hour of Code, Code.org, Made with Code by Google, Girls Who Code, Code2040, or Black Girls Code. The exposure of these social justice centered campaigns prompted reflection on gender, race and socioeconomic issues in the technology field. One pre-service teacher stated that "*I am amazed about the amount of girls that have begun to code, and have become interested in computer science and [I] love [that the coding program] empowers women*" (PT#106). Likewise some pre-service teachers were amazed by free coding resources like Code.org "*can make learning*

computer science more accessible for female and minority students [...] in the hopes of changing [the] narrative” (PT#305), or like Code2040 which “believes that as minorities rise, their presence in technology and innovation related companies needs to increase as well” (PT#313).

One implication of the pre-service teachers referencing the non-profits is the programming that these organizations provide primarily resides in out-of-school time. This seemed to confuse some of the teacher candidates and position CT and coding as an ‘extra’ activity.

Many of the pre-service teachers connected CT with their with their specialty within education. For example, participants discussed the unique role of early childhood, elementary and special education settings. This reflection also brought about questions of how to use CT for students’ learning such as ‘what does this look like for early learners?’ or ‘what is age appropriate?’. The participants also shared confusion on how to use CT pedagogically and how to integrate CT into different subjects. Pre-service teachers demonstrated their student-centered beliefs that align with computational thinking and the teaching sector that they focus on. This implies that we should not lose the insight of teachers’ professional knowledge since this could add pedagogical insights on infusing computational thinking into curriculum.

We found that pre-service teachers strongly demonstrated a positive attitude toward CT after the classroom exposure and the extension activity. However, pre-service teachers started to recognize the value of teaching CT to build problem solving skills within their future students. Some of the teacher candidates also reflected on the alignment of students having these skills and successful technology integration.

Though most participants had positive attitudes about CT, some pre-service teachers shared assumptions about its applicability. For example, a few of the teacher candidates already assumed only particular students would be interested in CT practice. They did not recognize the bias their reflections carried. In addition, while recognizing CT’s value in education, some were inclined not to include CT in their teaching due to perceived age appropriateness, time restrictions, and access to resources.

5. DISCUSSION

The findings from this case study provide an account of a first attempt to integrate CT into a teacher preparation course on educational technology. Our findings showed (a) an initial understanding of how computational thinking can be conceptualized for pre-service teachers’ expertise and (b) a clearer understanding of barriers among pre-service teachers to translate CT into classroom. We will use these findings personally to update the design of the course.

In the context of a 1.5 credit teacher preparation educational technology course with a long list of required contents, we struggled with a limited time frame to expose the pre-service teachers to CT. Our instructional design had positioned the hands-on exploration as the motivational factor within the CT activity. We had not anticipated the pre-service teachers being as interested in the larger societal impacts of

computing. In the future, we intend to outline these social-justice issues as a hook to increase interest at the beginning of the CT activity.

Additionally, it seemed the bulk of their content knowledge on CT came from the extension activity research where the teacher candidates found a resource to share. This self-led online research resulted in a wide variety of conceptualizations of CT. Misconceptions of CT amongst pre-service teachers is a common finding in similar research on training teacher candidates on CT (Sadik, Ottenbreit-Leftwich, Nadiruzzaman, 2017; Yadav et al., 2014). In the future, we will present a specific model of CT and give time for the pre-service teachers to discuss as a group how they could integrate CT into their future instruction.

6. SUBSTANTIATED CONCLUSION

Educational technology courses such as the one in this study need to be updated to build pre-service teachers’ competencies of CT integration in content areas (Yadav et al., 2017). While these teacher candidates are new to the profession they brought to the course knowledge, skills, and beliefs about education that should not be undervalued. Exposing pre-service teachers to CT helped the teacher candidates understand the relationship between ‘what is taught’ (content), ‘how it is taught’ (pedagogy) and ‘why it is taught’ (rationale and relevance) (Yadav, Hong, & Stephenson, 2016).

7. ACKNOWLEDGEMENT

Our thanks to the Learning Technologies Media Lab (LTML) at the University of Minnesota for providing robotics, programming materials and environments to support professional developments. A special thanks to Professor Dr. Cassandra Scharber, for sparking our interest in this subject and her continued advising support.

8. REFERENCES

- Balanskat, A., & Engelhardt, K. (2015). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe*. European Schoolnet, Brussels.
- Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, 38(6), 20-23.
- 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.
- Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (pp. 1-25).
- Freeman, A., Adams Becker, S., Cummins, M., Davis, A., and Hall Giesinger, C. (2017). *NMC/CoSN Horizon Report: 2017 K-12 Edition*. Austin, Texas: The New Media Consortium.
- Goode, J., Chapman, G., & Margolis, J. (2012). Beyond curriculum: the exploring computer science program. *ACM Inroads*, 3(2), 47-53.

- Google. (2015). Computational Thinking for Educators. Retrieved from <https://computationalthinkingcourse.withgoogle.com/>
- Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- International Society for Technology in Education. (2016). ISTE Standards for Students. Retrieved from <https://www.iste.org/standards/for-students>
- Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82, 263-279.
- 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.
- National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press.
- Sadik, O., Leftwich, A. O., & Nadiruzzaman, H. (2017). Computational thinking conceptions and misconceptions: progression of preservice teacher thinking during computer science lesson planning. In *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 221-238). Cham, Switzerland: Springer.
- Saldaña, J. (2016). *The coding manual for qualitative researchers: 3rd edition*. Thousand Oaks, CA: Sage.
- Shellenbarger, S. (2016, February 9). New ways to teach young children to code. *Wall Street Journal*. Retrieved from <https://www.wsj.com/articles/new-ways-to-teach-young-children-to-code-1455049777>
- 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.
- Yadav, A., Mayfield, C., Zhou, N., Hambruch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)*, 14(1), 5.
- 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.
- Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In *Emerging Research, Practice, and Policy on Computational Thinking* (pp. 205-220). Cham, Switzerland: Springer.

The Readiness of Computational Thinking Education in Taiwan: Perspectives from the K-12 Principals in 2017

Ting-chia HSU

National Taiwan Normal University, Taiwan
ckhsu@ntnu.edu.tw

ABSTRACT

This study investigated the perspectives of the K-12 principals who took part in the teacher education of computational thinking. The scales of the readiness survey questionnaire included object readiness, teacher readiness, instructional resource readiness, and leadership support. Moreover, this study also explored the TPACK of teachers for computational thinking education. The integrated questionnaire reports two validity statistics – the acceptable internal consistency (alpha reliability coefficient), and discriminant validity – for the refined 35 items. The results of the survey showed that those principals perceived the present situation which was significantly lower than the degree of importance they preferred. In other words, all the dimensions of the survey will have to be strengthened in the 2 years before conducting computational education for the 12-year compulsory education from August 2019 in Taiwan.

KEYWORDS

Computational thinking, teacher education, leadership, TPACK, readiness

1. INTRODUCTION

Computational thinking (CT) is a necessary form of literacy in the world with digital devices everywhere. CT is not only a kind of expertise which only computer engineers use in our stereotypical thinking. On the contrary, everyone should have an active attitude toward CT in order to understand and make use of this attainment (Wing, 2006). The competence and limitations of CT are both based on the process of operation and computing processing. No matter whether the computational process for solving a problem is executed by the human brain or computed by a computer, we can classify it into the process of CT. For example, through the process of reduction, embedding, transformation, and simulation, the operation of CT can decompose a seemingly complicated problem into several understandable and solvable ones (Wing, 2006). To put it simply, CT is a way of thinking which uses the basic concept of computer science to do problem-solving, system design, and understanding of human behavior. In the meantime, CT makes people adopt a thinking mode which the computer scientists adopt when they encounter difficulties (Grover & Pea, 2013). A previous study has found that most countries have tried to integrate CT courses into K-12 curricula based on a survey of 17 European countries (Balanskat & Engelhardt, 2014). In addition, the elementary and secondary schools in Australia have introduced CT into courses for a period of time, and have placed the literacy of CT in the national education curricula (Falkner, Vivian, & Falkner, 2014). Therefore, currently, many teachers are trying to integrate CT into

various courses (Heintz, Mannila, & Färnqvist, 2016). With the development of digital technologies and the present concerns about CT literacy, how should the teacher education be prepared for CT?

Recently, Orvalho indicated that teachers should follow the methodology for pre-service teachers: Before teaching students how to do CT, teachers should learn knowledge and abilities related to CT first (Orvalho, 2017). Yadav also pointed out that introducing computer science into pre-service courses can efficiently enhance teachers' understanding of CT. Moreover, students' reactions during their learning process tend to be more complicated. Therefore, the teacher can not only learn how to involve the literacy of CT in their courses, but can also help the students cultivate their problem-solving capabilities (Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014). Mouza combined CT with the TPACK (i.e., Technology, Pedagogy, and Content Knowledge) instructional method, and teachers designed CT courses associated with K-8 education when they were trained in teacher education. The result showed that the pre-service training not only had a positive influence on the teachers, but could also help them to develop and practice instructional content embedded with CT (Mouza, Yang, Pan, Ozden, & Pollock, 2017).

As CT is applied to teachers' training, the teachers know what CT is and how to integrate it into their courses. Moreover, the teachers can be earlier confronted with the possible failure that may happen in their teaching process in the future. Israel (2015) has applied CT to teacher education to overcome the obstacles for the teachers to achieve the expertise of the introduction to computer science course. On the other hand, the teachers would realize what difficulties the students with deficient resources may encounter. Through the teacher education for pre-service teachers, those pre-service teachers would benefit a lot and could know how to give support and assistance to their students (Israel, Pearson, Tapia, Wherfel, & Reese, 2015). In addition, when it comes to CT, visual programming cannot be forgotten. When the teachers design CT-related courses, they mostly use Scratch for the basic level. Cetin (2016) considered CT to be the foundation, and applied Scratch to pre-service teachers' training. The result indicates that this did indeed help teachers in arranging beginner courses, and the visual programming environment could help teachers better understand CT (Cetin, 2016).

The current study applied the same course mentioned above before research questions one to three (i.e., visual programming for mathematical learning unit) in the teacher training for the K-12 newly appointed principals. We then investigated the readiness of their schools via four scales:

technology readiness, teacher readiness, instructional resource readiness, and leadership support. In addition, we also investigated the technology, pedagogy, content, knowledge and the overall TPACK of CT based on the real conditions they perceived at present. At the same time, we also surveyed the preferred importance which the principals revealed for the same eight scales. We then explored the difference between the perceived present situation and the preferred importance demonstrated by the principals.

2. METHOD

2.1. Sample

There are 24 newly appointed principals participating in the teachers' training course for cultivating their literacy of computational thinking.

2.2. Questionnaire and Reliability Analysis

There are eight scales in the questionnaire. The first four scales were revised from the readiness questionnaire of mobile learning (Yu, Liu, & Huang, 2016). That questionnaire was named the support-object-personnel (SOP) m-learning readiness model, and was developed to assess the capacity for mobile learning readiness in primary and secondary schools in the previous study (Yu, Liu, & Huang, 2016). Darab and Montazer (2011) proposed an eclectic e-learning readiness scale which includes object readiness, software readiness, and leadership support (Darab & Montazer, 2011). In addition, Machado (2007) emphasized the importance of teacher readiness such as the professional application capabilities for e-learning. Cheon (2012) proposed the higher education m-learning readiness model based on the theory of planned behavior (TPB), and found that the attitude of a school had impacts on the undergraduates' perspectives on mobile learning (Cheon, Lee, Crooks, & Song, 2012). Accordingly, object readiness, teacher and instructional readiness, and leadership support are important scales for evaluating the readiness for putting something into practice at school, such as e-learning, mobile learning, or computational thinking, and so on.

Table 1. Descriptive Information for the first four Scales: Readiness

Scale Name	Description	Sample Item
Object readiness	For the current situation of equipment in the school, please answer the following questions.	There are enough information appliances such as computers for learning in the school, providing resources for technological courses.
Teacher readiness	For the current condition of teachers in your school, please answer the following questions.	There are full-time Information Technology teachers in my school.
Instructional resource readiness	For the arrangement of teaching materials for Technology domain, please answer the following questions.	The teachers in my school have capabilities to employ the official textbooks in the information technology courses.
Leadership support	For the attitude of school management,	School management proposes visions, policies, or projects that support and

please answer the following questions. encourage the teaching as well as learning in the technological domain.

Scholars have revised the model of pedagogical, content and knowledge (PCK) and proposed the model of TPACK (i.e., Technological Pedagogical Content Knowledge) (Mishra & Koehler, 2006). The framework clearly pointed out the relationship between the technological, pedagogical, and content knowledge of the teachers. Therefore, many studies have employed the TPACK model to evaluate the professionalism of teachers or the effectiveness of teacher education (Chai, Koh, & Tsai, 2010; Koehler, Mishra, & Yahya, 2007). Moreover, another study has introduced this model for the teachers to do self-assessment (Schmidt, Baran, Thompson, Mishra, Koehler, & Shin, 2009). This study also employed the TPACK model for the principals to do self-description for the school teachers in the technology domain at their schools.

Table 2. TPACK for Computational thinking teachers

Scales	Questionnaire items
Knowledge of technology	TK1-Our teachers know how to solve their own technical problems.
	TK2-Our teachers can learn new technology easily.
	TK3-Our teachers have the technical skills and use the technologies appropriately.
	TK4-Our teachers are able to use computational thinking tools or software to do problem-solving.
Knowledge of pedagogy	PK1-Our teachers can adapt their teaching style to different learners.
	PK2-Our teachers can adapt their teaching based upon what students currently understand or do not understand.
	PK3-Our teachers can use a wide range of teaching approaches in a classroom setting (collaborative learning, direct instruction, inquiry learning, problem/project based learning etc.).
	PK4-Our teachers know how to assess student performance in a classroom.
Knowledge of content	CK1-Our teachers have various ways and strategies of developing their understanding of computational thinking.
	CK2-Our teachers can think about the subject matter like an expert who specializes in computational thinking.
	CK3-Our teachers have sufficient knowledge about computational thinking.
TPACK	TPACK1-Our teachers can teach lessons that appropriately combine computational thinking, technologies and teaching approaches.
	TPACK2-Our teachers can use strategies that combine content, technologies and teaching approaches.
	TPACK3-Our teachers can select technologies to use in the classroom that enhance what they teach, how they teach and what students learn.
	TPACK4-Our teachers can provide leadership in helping others to coordinate the use of content, technologies and teaching approaches at my school.

Table 3 reports two validity statistics – namely, the internal consistency (alpha reliability coefficient), and discriminant validity – for the refined 35 items, including 20 items for readiness and 15 items for TPACK. Data are reported

separately for the perceived and preferred versions. The reliability data suggest that the refined version of each scale for readiness and TPACK has acceptable internal consistency. The reliability data suggest that the refined version of each scale has acceptable internal consistency.

Table 3. Internal Consistency (Cronbach Reliability Coefficient), and Discriminant Validity (Mean Correlation with other Scales), for Perceived and Preferred Versions.

Scale	Form	Alpha Reliability	Number of items	Mean Correlation
Technology readiness: The educational hardware of technology domain at school	Perceived present situation	0.701	5	0.17
	Preferred importance	0.728	5	
Professional development of the teachers in Technology domain	Perceived present situation	0.673	5	0.17
	Preferred importance	0.804	5	
The resource of instructional material	Perceived present situation	0.646	5	0.20
	Preferred importance	0.759	5	
Leadership support	Perceived present situation	0.835	5	0.28
	Preferred importance	0.766	5	
Knowledge of technology	Perceived present situation	0.840	4	0.28
	Preferred importance	0.876	4	
Knowledge of pedagogy	Perceived present situation	0.884	4	0.27
	Preferred importance	0.795	4	
Knowledge of content	Perceived present situation	0.943	3	0.42
	Preferred importance	0.869	3	
Overall TPACK of computational thinking	Perceived present situation	0.908	4	0.38
	Preferred importance	0.939	4	

3. DIFFERENCE BETWEEN PERCEIVED AND PREFERRED SITUATION

The results found that the principals perceived that their teachers had not fully prepared 2 years before conducting the 12-year compulsory education for computational thinking. The 12-year compulsory education will be carried out in August, 2019 while the investigation was done in 2017. From Table 4, it could be found that the preferred situation was significantly higher than the present situation in each dimension.

In terms of readiness, the technology readiness should be improved in 2 years. This part seems to be the easiest part to achieve in the future, but the teachers have to be trained at the same time. Otherwise, they may not know how to use the new equipment in their teaching.

Table 4. Paired Sample *t* test between perceived present situation and preferred importance.

Scale	Forms	N	Mean	SD	t	p
Technology readiness: The educational hardware of technology domain at school	Perceived present situation	24	3.00	0.82	-7.52***	.000
	Preferred importance	24	4.46	0.51		
Professional development of the teachers in Technology domain	Perceived present situation	24	3.28	0.93	-5.99***	.000
	Preferred importance	24	4.54	0.48		
The resource of instructional material	Perceived present situation	24	3.18	0.80	-4.19***	.000
	Preferred importance	24	3.94	0.59		
Leadership support	Perceived present situation	24	3.77	0.74	-4.00**	.001
	Preferred importance	24	4.39	0.50		
Knowledge of technology	Perceived present situation	24	3.95	0.69	-3.94**	.001
	Preferred importance	24	4.58	0.43		
Knowledge of pedagogy	Perceived present situation	24	3.83	0.87	-4.11***	.000
	Preferred importance	24	4.64	0.44		
Knowledge of content	Perceived present situation	24	3.74	0.99	-4.01**	.001
	Preferred importance	24	4.56	0.50		
Overall TPACK of computational thinking	Perceived present situation	24	3.58	0.97	-4.51***	.000
	Preferred importance	24	4.55	0.54		

***p<0.001; **p<0.01

In terms of TPACK, it was worrying to find that the principals tended to not have confidence in their teachers as they perceived that their knowledge of technology, pedagogy and content had not achieved the degree they expected. Therefore, the teacher education institutes have to put more effort into the development of instructional material and train the teachers to have the capabilities to develop their own material for computational thinking in the near future.

4. CONCLUSIONS

Based on the results of this investigation of the K-12 principals in Taiwan, there are some suggestions to enhance the preparation for involving computational thinking education in the 12-year compulsory education.

For object readiness, which refers to the educational hardware of the technology domain at school, it looks like it is the easy part if the government devotes money to the K-12 schools. However, the teachers have to be trained to know how to operate the new equipment, regardless of whether they are maker environment or computer technology products; otherwise, the payment for the hardware will be wasted. That perfect environment which is supposed to be constructed in the 2 years could not work without professional teachers. Therefore, future studies could further analyze the regression between the readiness of the teachers in the technology domain and the readiness of the hardware, and find direct evidence for this inference.

Unfortunately, the participants perceived that the leadership and management levels have not provided enough support for conducting computational thinking education. In other words, many people agree that computational thinking education is important; nevertheless, the leadership has not put enough emphasis on it. This study infers that the reason for this strange situation is that the literacy of computational thinking will not be regarded as one part for the senior high school or college entrance examination. However, normal education is also important. Schools should not only pay attention to the subjects related to the senior high school or college entrance examinations. Liberal education should be encouraged more.

In the 2 years, the related institutes have a large amount of work to do. The most important part is teacher education. The teachers in the technology domain should be trained to afford the requirements of instruction in the technology domain.

ACKNOWLEDGEMENTS

This study is supported in part by the Ministry of Science and Technology in Taiwan under contract number: MOST 105-2628-S-003-002-MY3.

5. REFERENCES

- Balanskat, A., & Engelhardt, K. (2014). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*: European Schoolnet.
- Cetin, I. (2016). Preservice Teachers' Introduction to Computing: Exploring Utilization of Scratch. *Journal of Educational Computing Research*, 54(7), 997-1021.
- Chai, C. S., Koh, J. H. L., & Tsai, C. C. (2010). Facilitating Preservice Teachers' Development of Technological, Pedagogical, and Content Knowledge (TPACK). *Educational Technology & Society*, 13(4), 63-73.
- Cheon, J., Lee, S., Crooks, S. M., & Song, J. (2012). An investigation of mobile learning readiness in higher education based on the theory of planned behavior. *Computers & Education*, 59(3), 1054-1064. doi:10.1016/j.compedu.2012.04.015
- Darab, B., & Montazer, Gh. A. (2011). An eclectic model for assessing e-learning readiness in the Iranian universities. *Computers & Education*, 56(3), 900-910. doi:10.1016/j.compedu.2010.11.002
- Falkner, K., Vivian, R., & Falkner, N. (2014). *The Australian digital technologies curriculum: challenge and opportunity*. Paper presented at the Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148.
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42(1), 38-43.
- Heintz, F., Mannila, L., & Färnqvist, T. (2016). *A review of models for introducing computational thinking, computer science and computing in K-12 education*. Paper presented at the Frontiers in Education Conference (FIE), 2016 IEEE.
- Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, 82, 263-279.
- Koehler, M. J., Mishra, P., & Yahya, K. (2007). Tracing the development of teacher knowledge in a design seminar: Integrating content, pedagogy and technology. *Computers & Education*, 49(3), 740-762.
- Machado, C. (2007). Developing an e-readiness model for higher education institutions: Results of a focus group study. *British Journal of Educational Technology*, 38(1), 72-82. doi:10.1111/j.1467-8535.2006.00595.x
- Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, 108(6), 1017-1054.
- Mouza, C., Yang, H., Pan, Y.-C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, 33(3).
- Orvalho, J. (2017). *Computational Thinking for Teacher Education*. Paper presented at the Scratch2017BDX: Opening, Inspiring, Connecting.
- Schmidt, D. A., Baran, E., Thompson, A. D., Mishra, P., Koehler, M. J., & Shin, T. S. (2009). Technological Pedagogical Content Knowledge (TPCK): The development and validation of an assessment instrument for preservice teachers. *Journal of Research on Technology in Education*, 42(2), 27.
- 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 (TOCE)*, 14(1), 5.
- Yu, Y.-T., Liu Y.-C., & Huang, T.-H. (2016). Support-Object-Personnel Mobile-Learning Readiness Model for Primary and Secondary Schools. *Journal of Research in Education Sciences*, 61(4), 89-120.

Two Studies of Perceived and In-Situ Readiness for Implementing the Computing Education in Singapore

Longkai WU*, Chee-kit LOOI*, Meng-leong HOW, Liu LIU
National Institute of Education, Singapore

longkai.wu@nie.edu.sg, chee-kit.looi@nie.edu.sg, mengleong.how@nie.edu.sg, liu.liu@nie.edu.sg

ABSTRACT

Computing education is garnering more attention from policy makers and educators both locally and globally. In Singapore, nineteen schools are beginning to offer the computing curriculum at the GCE “O” level, that is, for grades 9 and 10. If it is the case that computing education is standing on the verge of being formalized and offered as a mainstream subject, it will be important to understand teacher and student readiness towards the status quo of computing education in schools. This paper describe two studies: a survey study of computing teachers’ from the nineteen schools on their perceived readiness towards implementing computing curriculum; and an ethnographic study of four secondary schools with different degrees of in-situ readiness for both teachers and students during their implementation of the computing curriculum. Based on the two studies, we propose more systematic ways of preparing teachers to teach and students to learn the computing subject.

KEYWORDS

Computing Education, Computational Thinking, Teacher Readiness, Student Readiness

1. INTRODUCTION

In 2017, Singapore’s Ministry of Education (MOE) implemented a new curriculum for the Computing subject for grade 9 and 10 students in 19 schools. The new curriculum is a distinct shift teaching students from informal activities (infocomm clubs, code for fun, extracurricular activities et al.) to formal school education in development of students’ Computational Thinking (CT) skills and programming competencies. O Level MOE curriculum has provided guiding framework for computing teachers and students to take up their practices, but it takes time for them to build capacities and alignment in enactment. This paper tries to investigate different degrees of teacher and student perceived and in-situ readiness in local secondary schools prior to and during their implementation of computing curriculum and address the issue of formalization for teaching CT and programming in K-12 schools.

2. PREPARE TEACHERS AND STUDENTS FOR COMPUTING CURRICULUM

It is paramount to prepare in-service and future teachers to face the challenges of teaching Computational Thinking (García-Peñalvo et al., 2016). Hodhod, Khan, Kurt-Peker, and Ray (2016) argue that for students to acquire this important skill, teachers must acquire in-depth knowledge of the problem-solving strategies that utilize CT, and the strategies for integrating CT into their lesson plans. Some CT training workshops for teachers focus on K-12 students,

such as the one offered by Franklin et al. (2015) which provides advice for best practices in curriculum, content delivery, interfacing with schools, and classroom layout.

In the preparation of teachers for the teaching computing, Voogt, Fisser, Good, Mishra, and Yadav (2015) suggest adopting a multi-perspective approach, because many EU countries have computing teachers at the upper secondary school level, but too few at the lower secondary and primary school levels. At the primary school level, Voogt et al. (2015) assert that it is imperative for teacher education programs to recruit computer science specialists who can at least teach the basic notions of computing. In Israel, there is a shortage of computing specialists to teach in high school and it was necessary to train teachers of other subjects to teach CS by training them through a crash course which was comprised of about ten courses that form the basics of computer science (Gal-Ezer & Stephenson, 2014). Lepeltak, the director of learning focus in the Council of European Professional Informatics Societies (CEPIS), calls for a professionalization of teachers who are asked to impart CS lessons, even in other non-CS classes. Further, both Voogt and Lepeltak concur that teacher training could be pushed at the EU level to embark on the professionalization and the training of teachers (Bocconi et al., 2016).

3. STUDY 1: A SURVEY STUDY OF COMPUTING TEACHERS’ PERCEIVED READINESS TOWARDS IMPLEMENTING COMPUTING CURRICULUM

In Dec 2016, prior to the formal implementation of computing curriculum, we conducted a survey on computing teachers to seek their degrees of understanding, interest levels, capacities and challenges regarding the teaching of computing. 36 computing teachers (27 male and 9 female) from 19 schools participated in our survey.

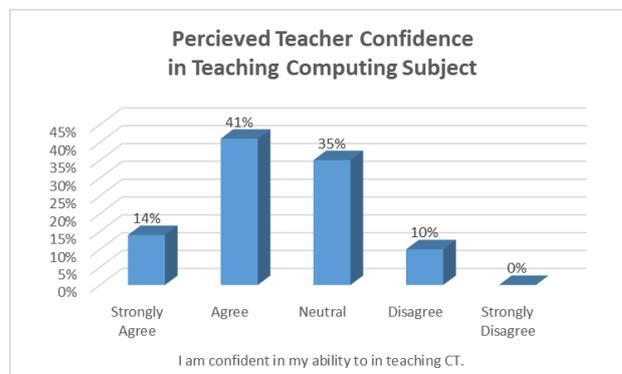


Figure 1. Perceived Confidence in Teaching Computing Subject

As to perceived teacher confidence to teach in computing subject (Figure 1), 56% teachers agreed that (14% strongly agree) they are confident to teach and implement CT in their classes. 24% are neutral while 14% consider that they are not ready.

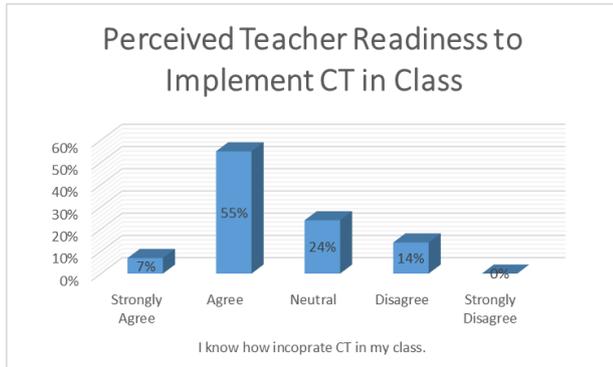


Figure 2. Perceived Readiness to Implement CT in Class

As to perceived teacher readiness to implement CT in Class (Figure 2), 63% teachers agree (7% strongly agree) they have been ready to incorporate and implement CT in their classes. 24% are neutral while 14% consider they are not ready.

As to perceived student readiness to learn computing in Class (Figure 3), 52% teachers considers (4% strongly considers) their students have been ready to learn computing. 28% are neutral while 21% consider they are not ready and CT is too complex to learn at the level of their students.

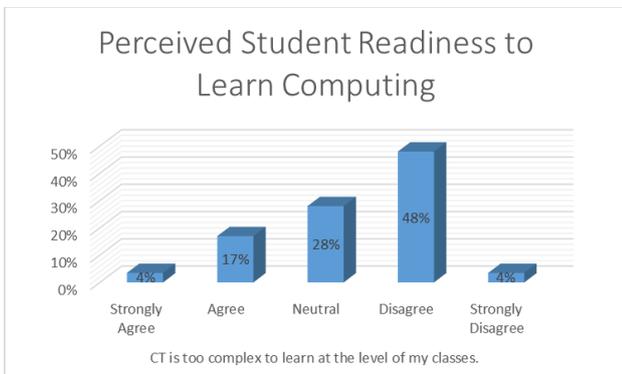


Figure 3. Perceived Student Readiness to Learn Computing

Thus, a discrepancy on the confidence and readiness in computing subject and to incorporate CT into the teaching and learning is observed among the computing teachers. A considerable portion of teachers lacks confidence in teaching CT and is unclear on how to bring out expected learning outcomes.

As to the challenges in teaching computing, lack of teaching resource (94%) ranks the first and lack of pedagogical knowledge (83%) ranks the second among the seven options (Table 1). When responding to open-ended questions, they also mention that they would need shared lesson plans and best practices by other schools to help their teaching. It is obvious that teachers are much more concerned about the resource for teaching and how to teach rather than what to teach, i.e. content knowledge.

Table 1. Perceived Challenges in Teaching Computing.

	Percentage	Count	Ranking
Teaching Resources	94%	34	1
Pedagogical Knowledge	83%	30	2
Ways to Motivate and engage students	69%	25	3
Instructional Skills	67%	24	4
Community Support	64%	23	5
Content Knowledge	61%	22	6
Computing Infrastructure in school	42%	15	7

4. STUDY 2: AN ETHNOGRAPHIC STUDY OF TEACHERS' IN-SITU READINESS IN IMPLEMENTING COMPUTING CURRICULUM

The literature has been mainly focused on preparing in-service and future teachers through professional development programs or workshops before computing is introduced into the curriculum at schools. It has been rare or lacking to develop an in-situ view to understand the readiness for computing of in-service teachers, as well as their students, as computing curriculum have been implemented in authentic classrooms.

To this end, we have adopted an ethnographic approach to conduct a field study in four local secondary schools which have implemented computing curriculum during the whole year of 2017. The researchers participated in the building and enactment of computing curriculum as active participants and took extensive field notes to record the observations, surveys and interviews. After the whole year implementation, we differentiate the four schools considering their different degrees of readiness to implement computing curriculum with respect to teachers and students.

4.1. School A – Basic Student and Teacher Readiness

As the teacher is new to teaching the computing subject, School A does not have a specific plan about what they are going to teach for the following weeks although provided with the Scheme of Work (SoW) by MOE. The topic and content may just be decided just before the class. The reasons could be the inexperience of teaching computing, as well as unfamiliarity with the computing curriculum.

Meanwhile, quite a large number of students respond to our survey revealing that they have chosen computing subject under the circumstance that they are not able to be enrolled into additional mathematics for the O-Level, which could be a better choice for them. Besides low motivation, it is also

noticed that this batch of computing students may not have sufficient English proficiency to do well in computing as language proficiency is considered by the teachers as important in articulating their answers in paper exam.

Thus, the degree of preparedness of the teachers and the students in School A to implement the computing curriculum can be further improved. More engaging activities can be incorporate into the learning of computing which the teachers are starting with be more familiar with over the year. Over the school year, the teacher is seen to be gaining more proficiency in teaching.

School B – Basic Student Readiness, High Teacher Readiness

In School B, the teacher has a high passion for teaching computing as he is highly interested in computing related knowledge or gadget. He advocates the coupling of 5E framework with the unplugged activities and argues that unlike the scientific inquiry process, computing subject can develop a cross-disciplinary mindset stressing for logic and conceptualization. He also believes that CT is not only about coding but also high-level planning that involves designing, decomposition and implementation. The students, in his opinion, should not become mere coders or coding workers. Instead, they should be equipped with a systematic mindset to solve complex problems.

During the class observation, researchers find that the teacher's scaffolding plays a significant portion in guiding students' actives. However, the students are not passionate or active in computing classroom as we have expected or comparing to other schools. They are also not quite used to teacher's scaffolding. The teachers explain that school B is a neighborhood school which the enrolled students are likely to be considered as low achievers since they have not performed well in The Primary School Leaving Examination (PSLE).

Therefore, school B teacher has developed a high degree of readiness in teaching computing in terms of beliefs and strategies. But the enactment has not been quite satisfying in the classroom with a relatively basic degree of readiness of students towards computing subject.

School C – High Student Readiness, Medium Teacher Readiness

In school C, the researchers conduct a focus group interview with seven Sec 3 students and find that all of them chose the subject out of their interests in technology. They believe the computing subject has met their interest and satisfied their curiosity to technology after taking the subject for the whole year. More surprisingly, the school actually does not provide any computing related course at levels of Sec 1 or Sec 2. These students' interests derive more from their parents' impact or future job considerations. Most of these students claim that they would continue to study computing when they are to be enrolled in polytechnic or university. They have also been very active and highly motivated in computing class. They tend to work in groups and initiate their own discussions about the computational problems. Peer learning has been undergoing when the high-achieving students actively help the low-achieving students. Their

proficiency and creativity in coding has also been surprisingly high as exhibited in their mini projects.

To meet the students' need, the two teachers who co-teach in this class intentionally enact their computing lessons at a difficulty level a bit higher than the O-level computing syllabus. However, they fell that their competency regarding content and pedagogy is not sufficient to teach this group of students as students always ask questions beyond their capacities. They also found difficulties in designing suitable practice tasks and exam questions for students. They rely a lot on an online learning system for homework assignment and grading which would save them time in grading students' codes. They complain that they do not have the one-year training in teacher training institute on computing like other subject teachers. Thus, they have to learn and teach at the same time all by themselves.

Thus, school C has a situation where the students are more ready to learn computing based on their own interests to an extent whilst the teachers are not sufficiently ready to teach.

4.2. School D – High Student Readiness, High Teacher Readiness

School D has two teachers and sixteen students in the computing class. Both of the teachers have gone through a computing education training course, which focuses mostly on content knowledge rather than the pedagogy or the knowledge about how to teach a specific computing topic. The teachers have to enhance content delivery with their own experiences in pedagogical aspects. Through the class enactment, the leading teacher creates his own version of unplugged activities (e.g., kinesthetic activities) to introduce computing topics and motive students to explore, corporate and present. He believes that communication and presentation is key in computing subject instead of being a silent coder who cannot make the design and solution to be understood. As to the student feedback, they have developed their interests in computing subject mainly because the teachers are highly enlighteningly in helping them to realize computing is to affect everybody's life and what they have learnt can be linked with real applications.

The 16 students in the computing class are mostly considered by teachers as high achievers. They like the computing subject since the interactive and immersive process has made it more interesting and attractive comparing to other O' level subjects. In the focus group discussion, they are confident and determined to be "A" scorers in the coming O' Level exam for computing subject. Their concern are more with the opaque opportunities to continue to learn computing subjects after secondary school level.

Therefore, both teachers and students in School D have a high level of confidence and competency in computing. Comparing to other schools, they are capable to implement computing curriculum mainly with their resources and capacities.

5. DISCUSSIONS AND CONCLUSION

In this paper, we describe a survey study and an ethnographic study on both perceived and in-situ readiness in the implementation of the computing curriculum. We find

that the degrees of perceived and in-situ readiness of teachers and students to teach and learn in computing subject vary among the different schools. The factors influencing students' readiness have mainly been their interests, motivation, and learning competencies. For teachers, their degrees of readiness are more related to their beliefs, teaching strategies, pedagogical preparations and available teaching resources. Readiness towards computing of teachers and students seem to be more self-initiated, rather than school-initiated. A lack of systematic ways to prepare more teachers and students to be enrolled in computing subject, is perceived. Students need more resources to cultivate their interests in computing, whilst teachers require more training and teaching resources to develop adaptive expertise to instruct different groups of students. The schools also need to adopt more adaptive strategies for different computing teachers and different groups of students to maximize learning effectiveness.

6. REFERENCES

- Bocconi, S., Chiocciariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education-Implications for policy and practice*. *EdMedia* 2016. <https://doi.org/10.2791/792158>
- Gal-Ezer, J., & Stephenson, C. (2014). A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States. *ACM Transactions on Computing Education*, 12(2), 8:1–8:18.
- García-Peñalvo, F., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). TACCLE 3, O5: An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers KA2 project " TACCLE 3 – Coding " (2015-1-BE02-KA201-012307), 72. <https://doi.org/10.5281/zenodo.165123>.
- Hodhod, R., Khan, S., Kurt-Peker, Y., & Ray, L. (2016). Training Teachers to Integrate Computational Thinking into K-12 Teaching. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16* (pp. 156–157). <https://doi.org/10.1145/2839509.2844675>
- 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. <https://doi.org/10.1007/s10639-015-9412-6>

Acknowledgements:

The work reported in this paper is funded by NIE grant OER 04/16 LCK. We thank Peter Seow and Wendy Leong for their assistance in this research.

中學資訊科技教師運算思維學科教學能力調查

蔡旻穎, 吳正己*, 游志弘

國立臺灣師範大學

myt@ntnu.edu.tw, chihwu@ntnu.edu.tw, chihung@ntnu.edu.tw

摘要

臺灣 K-12 課程中的「資訊科技」科目新課綱是以培養學生運算思維為主軸, 預定於 2019 學年度實施, 現職資訊科技教師 (computing teachers) 能否掌握「資訊科技」學科教學知識是新課綱實施成功的關鍵因素。本研究以線上問卷調查方式讓中學資訊科技教師自我評量其學科教學知識, 共回收 50 份有效問卷。結果顯示, 整體上, 教師自評之 PCK 足以勝任新課綱教學。在「程式設計與運算思維」面向中的自評結果, 資訊系所畢業教師優於非資訊系所, 高中教師優於國中教師, 而非師範體系畢業教師優於師範體系畢業教師。未來建議擴大研究樣本, 並透過訪談對教師的 PCK 做更深入的分析。

關鍵字

資訊科技課程; 學科教學知識; 程式設計; 運算思維

1. 前言

面對全球化競爭、培育國家人才需求及因應資訊科技的迅速發展, 包括美國、英國、德國、以色列和澳洲等先進國家皆已重新調整資訊科技課程之理念與架構, 除獨立設立科技領域亦致力於推動資訊教育課程改革, 顯見資訊教育的發展已日趨重要。當前臺灣時值新課綱的編修階段, 在因應資訊科技日新月異、國際教育發展潮流和強化臺灣未來競爭力之迫切需求下, 已在新課綱中增設「科技領域」, 並規劃由「資訊科技」與「生活科技」兩項學科組成, 而資訊科技科目也被列為中學教育階段之必修課程 (教育部, 2016)。此外, 有鑑於數位時代中運算思維於生活中密不可分, 臺灣將培養運算思維能力納入新制訂的資訊科技課程中, 期能有效培養學生運算思維, 而學習程式設計為實踐目標行之有效的根本途徑 (林育慈、吳正己, 2016)。

Shulman (1986, 1987) 將學科教學知識 (pedagogical content knowledge, PCK) 界定為學科內容知識與教學知識二者的融會, 是教師教學專業的重要知識。各方學者也針對教師應具備之 PCK 的知識種類、內涵、定義及發展脈絡也分別提出不同闡釋 (Grossman, 1988; Tamir, 1988; Cochran, DeRuiter, & King, 1993)。PCK 共同核心為教師在面對各教學情境時, 須能運用有效的表徵方式和統整資源來呈現特定學科主題內容, 同時亦須掌握學生先備知識與學習困難, 並據以轉化為學生易理解之方式教學。NARST (National Association for Research in Science Teaching) 與美國 NRC (National Research Council) 所提出之國家科學教育標準 (National Science Education Standards, NRC, 1996) 亦

標榜 PCK 之重要性, 並將之列為教師專業能力的評鑑指標。

PCK 為特定學科教師應具備的知識, 不同學科領域之教師所須具備的 PCK 迥然有別。近年 PCK 的相關研究早已在各學科領域中蓬勃發展, 有些研究以質性方式探討教師 PCK 的發展、PCK 對教學的影響、或比較資深與新手教師間 PCK 的差異 (Grossman, 1988; 邱美虹、江玉婷, 1997); 有些研究以量化方式, 讓學生填寫量表或問卷, 並輔以相關質性資料來了解教師的 PCK 表現 (Lederman, Gess-Newsome, & Latz, 1994; 王國華、段曉林、張惠博, 1988)。陳彥廷 (2014) 則以自行編製的 MPCK (Mathematics Pedagogical Content Knowledge) 量表提供國小數學教師自我評量 PCK 能力。有關「資訊科技」科目 PCK 的研究較少, 相關文獻僅發現對資訊科技教師專業能力模型 (Competences for Teaching Computer Science Model) 及資深與新手資訊科技教師分別應具備的 PCK 能力進行探討 (Berges et al., 2013; Margaritis & Magenheimer, 2015), 且研究內容多側重於一般教師的教學知識要求, 並非針對資訊科技教師所持 PCK 進行探討。

面對新課綱的推展, 課程中新的教學理念、教材與教法, 對教學現場的資訊科技教師均是新的挑戰和要求。資訊科技教師必須熟悉資訊科技學科內容知識, 了解學科獨有的 PCK 來進行資訊科技教學。目前臺灣缺乏資訊科技教師 PCK 的相關研究, 對於現職資訊科技教師 PCK 能力現況仍不明; 進一步的, 對教師「程式設計與運算思維」面向 PCK 之掌握更是刻不容緩。本研究旨在編製「資訊科技教師 PCK 自我評量問卷」, 調查臺灣現職中學資訊科技教師 PCK 之現況, 並分析不同的背景變項對教師在「程式設計與運算思維」面向 PCK 的影響, 以提供因應新課綱實施資訊科技師教學專業發展的參考。新課綱僅於中學有資訊科技課程, 本研究以中學教師為研究對象, 茲列研究目的如下:

- (一) 了解現職資訊科技教師學科教學知識現況。
- (二) 探討不同背景的資訊科技教師在「程式設計與運算思維」面向學科教學知識之差異。

2. 研究方法

2.1 研究參與者

本研究透過寄發電子郵件發送給各校資訊科技教師填答, 共有 74 位中學教師填答, 61 份為完整填答問卷, 其中共有 50 份有效問卷。填答者背景資料如表 1, 包括 28 位男性及 22 位女性教師, 任教學校教育階段分布平均, 國中教師 (48%)、高中教師 (52%); 填答者大

部分任職年資 6 年以上 (86%)，且大都畢業於資訊相關系所 (80%)。

表 1 研究參與者背景資料

背景變項	屬性	N	%
性別	男	28	56.0
	女	22	44.0
年齡	40 歲 (含) 以下	16	32.0
	41 歲 (含) 以上	34	68.0
任教年資	1-5 年	7	14.0
	6-15 年	7	14.0
	16-25 年	20	40.0
	25 年以上	16	32.0
畢業系所	資訊相關系所	40	80.0
	非資訊相關系所	10	20.0
修畢教育專業課程的學校	師範或教育大學	37	74.0
	非師範或教育大學	13	26.0
修畢資訊專門科目的學校	師範或教育大學	21	42.0
	非師範或教育大學	29	58.0
任教學校教育階段	國中	24	48.0
	高中	26	52.0
學校位於臺灣所在區域	北部	27	54.0
	中部	11	22.0
	南部	12	24.0
學校位於所在縣市之	都會市區	30	60.0
	一般鄉鎮地區	20	40.0

2.2 實施程序與工具

本研究分為四個階段進行：

(1) 定義資訊科技教師 PCK 內涵

參考相關文獻後將 PCK 歸納為六類知識面向，分別為課程、學習者及其背景知識、學科整體知識、學科教學方法與情境、程式設計與運算思維及學科學習成效評估知識。

(2) 編製問卷題目初稿

以臺灣教育部 (2016) 「科技領域師資職前教育專門課程規劃計畫」報告為主要架構，並參考「國小教師數學教學知識 (MPCK) 知覺量表」(陳彥廷, 2014)，針對各知識面向編製問卷題目初稿，共 40 題。

(3) 問卷定稿

邀請資訊教育專家學者及四位中學資訊科技教師，檢視問卷內容並提供修改建議，修訂成正式問卷，定名為「資訊科技教師 PCK 自我評量問卷」。問卷內容除詢問教師基本資料外，包含六類知識面向的 PCK，共有 38 個自評問題。自評採 Likert 五點量表形式，以 1 至 5 分分別表示：非常不同意、不同意、普通、同意及非常同意等自評結果。另有一題開放式問題，調查教師期望研習之教學專業發展課程主題。

(4) 實施問卷調查

蒐集中學資訊科技教師電子郵件地址，寄發邀請填寫問卷函，請教師於線上填寫。撰寫本文時，共有 74 位教師填寫問卷。

3. 結果與討論

資訊科技科目新課綱著重培養學生運算思維，本文結果與討論將著重在資訊科技教師於「程式設計與運算思維」面向之自評結果為主 (3.2 及 3.3)，惟仍將概略呈現資訊科技教師在各知識面向之自評結果 (3.1)。

3.1 各知識面向 PCK 自評結果

資訊科技教師 PCK 自評結果如表 2。整體上，教師認為他們具備的 PCK 能夠符合新課綱的教學需求，各知識面向平均分數皆在 4 分 (同意) 左右；其中學科整體知識 (M = 4.09) 得分最高，其次為學習者及其背景知識 (M = 4.03)。顯示教師認為自己已充分掌握資訊科技學科知識體系，並瞭解學生學習相關的知識。平均最低的項目為學科學習成效評估知識 (M = 3.85)，顯示教師們認為自己在學習評量方面的知識略顯不足。

表 2 各知識面向 PCK 自評結果 (N = 50)

知識面向	題數	Mean	SD
1. 課程	5	3.95	.55
2. 學習者及其背景知識	3	4.03	.53
3. 學科整體知識	6	4.09	.49
4. 學科教學方法與情境	14	3.88	.53
5. 程式設計與運算思維	7	3.89	.50
6. 學科學習成效評估知識	3	3.85	.60

3.2 「程式設計與運算思維」PCK 自評結果

教師於「程式設計與運算思維」面向各題自評結果如表 3 所示。在程式設計部分，第 2 題到第 4 題平均分數皆達到 4 分，表示教師普遍了解各類型程式設計學習工具的特性，並能選用適合的學習工具，同時也理解學生學習程式設計的困難。另外，「能否運用適當的策略來教授程式設計」的得分相對較低 (第 1 題，M = 3.76)，這意味著教師對目前使用的教學法能否有效幫助學生學習程式語言可能較無把握。由教師於開放問題的填答亦獲得相呼應，多數教師表示希望有關單位能提供程式設計教學法增能課程研習機會。

表 3 「程式設計與運算思維」面向自評結果 (N = 50)

「程式設計與運算思維」面向	Mean	SD
1. 我知道運用適當的策略來教授程式設計。	3.76	.66
2. 我可以用不同的程式設計工具 (如：文字式、積木式、流程圖等) 來解決問題。	4.08	.75
3. 我能依學生程度選擇合適的程式設計學習工具 (如：文字式、積木式、流程圖等)。	4.06	.68
4. 我知道學生學習程式設計時容易遭遇的困難。	4.04	.75
5. 我了解運算思維的意涵。	4.00	.61
6. 我會設計培養學生運算思維的學習活動。	3.84	.68
7. 我能結合其他學習領域於運算思維教學。	3.44	.76

有關運算思維方面，教師認為自己是了解運算思維內涵（第5題， $M = 4.00$ ），但在設計培養運算思維的學習活動（第6題， $M = 3.84$ ）及結合其他學習領域於運算思維教學（第7題， $M = 3.44$ ）的信心與教學知識略顯不足。推論是由於運算思維是近年才被提出的概念，過去未納入在師資培育課程中，顯然現職教師過去所學未足能應付課程內容的變革，而此結果也反應在開放式問題的回饋中，多數教師指出運算思維課程設計為亟須增辦的研習主題。

除此之外，於「學科學習成效評估知識」的知識面向中，其中一題項與運算思維相關，題目內容為「我能選用合適的評量方法評出學生整合運算思維與資訊科技來解決問題的能力」（ $M = 3.78$ ），是該面向中平均得分最低的項目，顯示教師對如何評估運算思維與問題解決能力較無把握。何榮桂（2015）指出新課綱強調兼重資訊科技之學科理論與實作，而兩者的評量方式不盡相同，從實務面該如何評估運算思維能力，是新課綱內容應具體示例說明，提供給教學現場教師有所依循。

3.3 教師背景與程式設計與運算思維面向自評差異

此部分資料使用單因子變異數分析，探討資訊科技教師不同背景變項對其「程式設計與運算思維」PCK的影響。

依「任教學生年段」背景變項的分析結果顯示，高中教師在運用程式設計教學策略（ $F = 5.58, p < .05$ ）、使用程式設計工具（ $F = 14.36, p < .05$ ）、依學生程度選用程式設計學習工具（ $F = 8.18, p < .05$ ）、了解學生學習時容易遭遇的困難（ $F = 19.11, p < .05$ ）、設計培養運算思維的學習活動（ $F = 4.97, p < .05$ ）和結合其他學習領域於運算思維教學（ $F = 6.66, p < .05$ ）上的自評結果高於國中教師。

其次，在「畢業系所是否為資訊相關科系」背景變項分析結果顯示，畢業系所為資訊相關科系之教師在依學生程度選用程式設計學習工具（ $F = 6.29, p < .05$ ）與結合其他學習領域於運算思維教學（ $F = 4.49, p < .05$ ）上的自評結果高於非資訊相關科系畢業教師。

於「修畢教育專業課程之學校是否為師範或教育大學」的背景變項結果顯示，非師範或教育大學修畢教育專業課程之教師在結合其他學習領域於運算思維教學（ $F = 5.47, p < .05$ ）上的自評結果高於在師範體系修畢教育專業課程之教師。

然而，不同性別、年齡、任教年資、修畢資訊專門科目的學校屬性、任教學校位於臺灣所在區域與位於所屬縣市之屬性等背景變項對教師在「程式設計與運算思維」面向的自評結果無顯著的影響。

4. 結論與建議

由前述調查結果顯示，整體而言，資訊科技教師認為他們能夠勝任新課綱的教學需求，尤其在資訊科技的「學科整體知識」面向最佳，但在「學科學習成效評估知識」面向則略顯不足。

在「程式設計與運算思維面向」之自評結果則顯示，教師已掌握各類型程式設計工具特性和學生學習程式設計時易遭遇的困難，且能依學生程度來選擇適合的學習工具；而教師自我感覺在教授程式設計運用的教學策略與方法為其次，運算思維是新興概念，雖然結果顯示教師已了解運算思維的內涵，但過去未能在師資職前課程中獲得有關教學專業知識，在運算思維課程設計與評量工具的部分是亟待有關單位能增辦研習活動，幫助教師專業成長。

不同背景之教師對「程式設計與運算思維」各題 PCK 之自評，因任教學生年段、畢業系所是否為資訊相關科系、及是否為師範或教育大學畢業而有顯著差異。

建議未來研究應擴大問卷填答人數，並進行訪談，以深入分析與釐清教師所具備的 PCK 全貌，據此針對不同背景教師提供增能研習，以提升教師學科教學知識。

5. 參考文獻

- 王國華、段曉林、張惠博（1998）。國中學生對科學教師學科教學之知覺。*科學教育學刊*，6(4)，363-381。
- 何榮桂（2015）。試論十二年國民基本教育「資訊科技」課程綱要規劃草案。*科學教育月刊*，250，48-64。
- 邱美虹和江玉婷（1997）。初任與資深國中地球科學教師學科教學知識之比較。*科學教育學刊*，5(4)，419-459。
- 林育慈和吳正己（2016）。運算思維與中小學資訊科技課程。*教育脈動*，(6)，5-20。
- 陳彥廷（2014）。國小教師數學教學知識（MPCK）知覺量表發展之探究。*測驗學刊*，61(1)，51-78。
- 教育部（2016）。*中華民國師資培育統計年報*。臺北市：教育部。
- 教育部（2016）。科技領域師資增能研析計畫結案報告。未出版。
- 教育部（2016）。十二年國民基本教育課程綱要國民中小學暨普通型高級中等學校科技領域（草案）。2017年2月1日，取自：國家教育研究院 http://www.naer.edu.tw/ezfiles/0/1000/attach/92/pta_10229_131308_94274.pdf
- Berges, M. et al. (2013). Developing a competency model for teaching computer science in schools. In *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 327-327). ACM.
- Cochran, K. F., DeRuiter, J. A., & King, R. A. (1993). Pedagogical content knowing: An integrative model for teacher preparation. *Journal of teacher education*, 44(4), 263-272.
- Grossman, P. L. (1988). *A study in contrast: Sources of pedagogical content knowledge for secondary English*. Unpublished doctoral dissertation, Stanford University, Stanford, CA.
- Lederman, N. G., Gess-Newsome, J., & Latz, M. S. (1994). The nature and development of preservice science teachers' conceptions of subject matter and

pedagogy. *Journal of Research in Science Teaching*, 31(2), 129-146.

Margaritis, M., & Magenheim, J. (2015, March). Pedagogical content knowledge a comparative study between CS pre-service teachers and experienced teachers. In *Global Engineering Education Conference (EDUCON)*, 2015 IEEE (pp. 102-111). IEEE.

Shulman, L. S. (1986). Those who understand: Knowledge growth in teaching. *Educational researcher*, 15(2), 4-14.

Shulman, L. (1987). Knowledge and teaching: Foundations of the new reform. *Harvard educational review*, 57(1), 1-23.

Tamir, P. (1988). Subject matter and related pedagogical knowledge in teacher education. *Teaching and teacher education*, 4(2), 99-110.

基于计算思维培养的教师培训课程设计与实践

刘显辛, 陈桃*, 查思雨, 张安琦

北京师范大学 教育技术学院

201622010028@mail.bnu.edu.cn, teastick@gmail.com, 18110027072@163.com, zhanganqi19950601@163.com

摘要

在新一轮修订的中国大陆高中信息技术新课标中,界定了计算思维作为信息技术学科的核心素养要素而应该被重点培养。但是就计算思维的培养而言,教师起着至关重要的作用。本文在前人研究的基础上提出了计算思维教师培训课程的四阶段模型,并根据此模型设计开发了为时4天的workshop活动课程,探索了教师在参与课程前后对计算思维的感知上的变化。结果发现,通过此次培训,教师对计算思维的理解、自我效能感发生了显著的变化,而内在动机和计算思维的课程整合则无显著影响。

关键字

计算思维; 教师培训; 感知;

1. 前言

计算思维是当前国际计算机界广为关注的一个重要概念,也是当前计算机教育需要重点研究的重要课题,是当前一个颇受关注的涉及计算机科学本质问题和未来走向的基础性概念。1996年,麻省理工学院(MIT)的西蒙派珀特最早提出这一概念,但在2006年3月,美国卡内基梅隆大学的周以真教授在ACM会刊《Communications of the ACM》上第一次将计算思维推向前台。她认为,学会计算思维是在信息社会中创新的需要。如同所有人都具备“读、写、算”(简称3R)能力一样,计算思维是必须具备的思维能力(Angel, 2016)。目前,计算思维已经引起了计算机科学家和教育界人士的广泛关注。在新一轮修订的中国大陆高中信息技术新课标中,界定了计算思维作为信息技术学科的核心素养要素而应该被重点培养。但是就计算思维的培养而言,编程教育是培养学生计算思维的有效方式,对学生在计算思维的培养方面有着得天独厚的优势,有利于培养学生的问题分析能力和问题解决能力。

2017年7月,国务院印发《新一代人工智能发展规划》,明确指出人工智能成为国际竞争的新焦点,应逐步开展全民智能教育项目,在中小学阶段设置人工智能相关课程、逐步推广编程教育、建设人工智能学科,培养复合型人才,形成中国人工智能人才高地。可见,无论是国家的政策层面要求还是计算思维的内涵所在,培养学生的计算思维都很有意义且迫在眉睫。在我国,信息技术课主要承载着培养学生计算思维的主要任务。但是本研究前期对北京市海淀区的信息技术教师(含电教教师)进行了有关计算思维的调查(问卷详见附件),结果显示虽然95%的教师都认为很有必要为学生开设计算思维相关的课程,但是89.5%的教师没有接受过这方面的学习以及不知道如何将其整合到自己的

课程中,基于此,本研究对如何开展计算思维的教师培训课程进行了深入探究。

2. 文献综述

2.1. 计算思维

计算思维是计算机科学实践的核心,是21世纪数字公民的一项基本素养。2006年,美国卡内基梅隆大学周以真(Jeanette Wing)教授提出,计算思维是运用计算机科学的基本概念进行问题解决、系统设计与人类行为理解的过程。2010年,她再次补充定义计算思维是一种解决问题的思维过程,能够清晰、抽象地将问题和解决方案用信息处理代理(机器或人)所能有效执行的方式表述出来(Wing, 2006)。计算思维提供了一种能够广泛应用于工作、学习和生活中的组织与分析问题的新视角,同时它可以连结计算机科学与其他学科知识领域,突破了专业知识技能与思想的局限,促使学习者进行技术使用者到创造者的角色转变。她还提出计算思维包括算法、分解、抽象、概括和调试五个部分。

2011年,美国国际教育技术协会(International Society for Technology in Education, ISTE)联合计算机科学教师协会(Computer Science Teachers Association, CSTA)基于计算思维的表现性特征,给出了一个操作性定义:“计算思维是一种解决问题的过程,该过程包括明确问题、分析数据、抽象、设计算法、评估最优方案、迁移解决方法六个要素”。

虽然关于计算思维的定义,目前还没有达成共识,但是周以真提出的计算思维所包含的5个要素得到了很多研究者的认可。抽象,是指在解决问题时去掉次要的细节;概括,是指总结、发现规律或相似点的过程,发现整个大任务中已经熟知的部分,或者在其他地方已经了解过的部分,进而使算法变的简单;分解,就是将大的问题分成较小的部分,进而简化问题,更易于向别人解释;算法,就是通过设计一系列具体有序的步骤来解决问题;调试,就是不断发现错误、改正错误的过程。综上所述,本研究认为计算思维是一种包含抽象、概括、分解、算法和调试5个要素的问题解决过程。

2.2. 计算思维的研究现状——K-12 阶段

计算思维往往和编程同时出现,Lye(2014)认为编程是一项需要将问题进行抽象和分解的活动,使得学生进行思考并因此促进计算思维能力的发展。Werner, Denner, Campe, & Chizuru Kawamoto(2012)通过让学生使用编程软件 Alice 完成特定的任务来衡量学生计算思维能力的发展。很多研究者(eg. Wilson, Hainey, & Connolly(2013); Pardamean & Suparyanto(2015))认同这

一观点，同时也促进了编程工具在教育领域的发展。目前比较流行的是图形化编程环境，如 Scratch, Alice, Game Maker 等，这些工具对于初学者来说简单易用，只用拖拽相应的模块就可以实现一些功能(Grover & Pea, 2013)。除此之外，美国《中小学计算机科学标准》分阶段设计了计算思维的教学实施方案，建议在 K-6 年级，将学习内容设计成创造性和探究性活动，嵌入到社会科学、语言艺术、数学和科学课程中；7-9 年级，学校可以根据情况开设独立的计算机课程，也可以整合学科内容到其他课程中；10-12 年级以必修课的方式达成学习目标。

与美国混合式教学不同，英国则采取独立开课模式。自 1988 年以来，信息技术课程就一直作为英国中小学生的必修课程。为顺应时代发展，该课程先后经历了从信息技术(Information Technology, IT)到信息通信技术(Information and Communication Technology, ICT)，再到计算(Computing)的变革。英国教育部于 2014 年 9 月引入新的计算机教学大纲，将课程要求划分为四个阶段：K-2 年级，理解算法概念，能够创建和调试简单的程序等；3-6 年级，编程解决实际问题，了解计算机网络，有效使用搜索技术等；7-9 年级，理解几个反映计算思维的关键算法，掌握 1-2 门程序设计语言解决计算问题，熟悉计算机组成等；10-11 年级，培养计算机科学、数字媒体和信息技术的知识、能力和创造力，发展问题分析、解决、设计和计算思维能力等。

2.3. 计算思维的教师培训

教师学会将计算思维整合到课堂实践中对于培养学生的计算思维是非常重要的 (Prieto, 2014)。有研究者尝试对职前教师和在职教师进行培训，如 Blum (2007) 通过历时一周的工作坊活动，向教师们介绍计算思维以及计算机科学与其他学科的联系，研究了工作坊活动是否影响教师对计算机科学的认识，结果发现教师对计算机科学的认识发生了显著地变化，并且愿意将计算思维的相关内容整合到自己的教学实践中来，研究者还发现在职教师想要更多的实践层面的指导以及更多的教学资源。虽然很多研究都关注在职教师的专业发展，但是很少呈现教师培训课程如何设计及实施状况。Angeli 等(2016)提出了 k-6 阶段的计算思维教学通用框架，并从 TPCK 的角度出发，探讨了教师在进行计算思维教学时应具备的基于计算思维的学习者知识、教学法知识、技术知识和学科知识 (见图 1)。

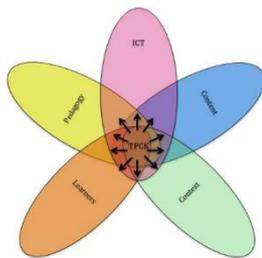


图 1 TPCK 模型

3. 研究目的

本研究设计教师培训课程帮助在职教师学习如何开展计算思维课程，并且在课程实施的过程中，探索教师在参加培训课程前后对计算思维的态度变化及自己的理解。

4. 研究过程

4.1. 参与者

本研究的研究对象为北京市海淀区几所学校的在职教师。这门课程是由海淀区教育科学研究院发起的，旨在培训教师有关计算思维及相关课程的设计和实 施，推进人工智能的发展。选课人数为 27 人，去除未填写前测问卷或后测问卷的教师，最终样本为 16 人，95% 的教师都在原学校教授信息技术或通用技术课程，其中男生 9 人，占 56%；女生 7 人，占 44%。

4.2. 研究流程

本研究的流程如图 2 所示。首先，研究者根据教师培训的框架及计算思维的特性设计并开发出培训课程；然后，在课程实施之前，对教师进行前测，包括计算思维的理解、自我效能、参与培训的内在动机及对计算思维的 课程整合的看法等；接下来，实施培训课程，整个培训课程历时四天，以 workshop 的形式开展，不仅有个人的学习，也有小组的挑战任务；最后对教师进行后测及访谈。

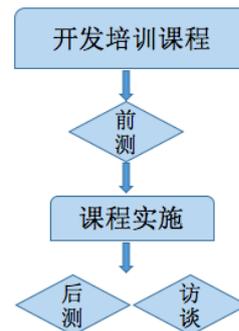


图 2 研究流程图

4.3. 课程设计

本课程设计基于的学习工具为 Swift Playgrounds，这是一款在 iPad 上使用的 app，让编程变得更加轻松、灵活。学习者可以一边写代码，一边看到代码造就的成果，并且大量运用所熟知的单词和词组，学生通过钻研并打通一个又一个关卡，不断完善编程技巧，逐步夯实编程知识的基础。但是，在完成项目的时候，编程只是所需能力中的一小部分，更重要的是实施的策略、寻找错误和解决问题的能力、分享和协作的能力以及面对挑战的能力等，这些也是非常重要的。

本研究在 TPCK 模型的基础上，结合计算思维的特点，提出了计算思维教师培训的四阶段模型，见图 3。

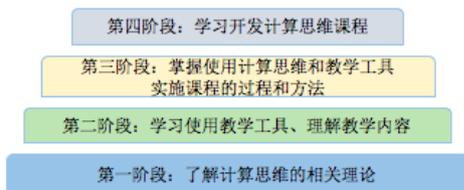


图3 教师培训四阶段模型

第一阶段是让教师了解计算思维的相关理论，包括计算思维的概念、计算思维对于学生发展的重要意义等；第二阶段，让教师学会使用 Swift Playgrounds，知悉其中所包含的编程的基本知识和方法，如算法、序列、条件、判断和调试等；第三阶段，让教师掌握使用该工具开展课程教学的策略、过程和方法等；第四阶段，也即培训过程中需要达到的最高目标，让教师学会开发计算思维课程（基于任一学习工具），四个阶段的难度是逐级递增的，因此达到不同阶段目标的教师的数量也是不同的。

需要特别说明的是，对编程基本知识学习的过程中，教师学习知识只占其中的一小部分，更多的是了解课程是如何设计的、是怎样培养计算思维的不同要素以及思考怎么将此课程带到原学校实施。第四天的课程开发阶段，教师需要选取以往自己设计的教案，带到培训课堂上，用培训课程中学到的相关知识重构自己原来的课程，通过分享和交流实现碰撞。

4.4. 研究工具

本研究所用的计算思维感知量表改编自(Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011)和(Shim, Kwon, & Lee, 2017)的量表。该量表主要测量学习者对计算思维的理解、自我效能、参与培训的内在动机以及对计算思维的课程整合的看法四个维度，共 15 道题，采用李克特5点量表，包括“完全不同意”、“比较不同意”、“同意”、“比较同意”、“完全同意”五个选项，分别计 1-5 分，分数越高表明学习者的计算思维感知越好，问卷的前后测信度系数分别为 0.838 和 0.876，表明信度良好。

5. 研究结果

本研究采用 SPSS20.0 对数据进行处理和计算，采用配对样本 t 检验对教师参加培训前后对计算思维的感知结果进行分析，以检验培训课程对教师计算思维的的理解、自我效能、内在动机、课程整合等的影响。表 1 是计算思维感知量表四个维度前后测量的结果。在对计算思维的理解上， $\text{sig}=0.001(<0.05)$ ；在自我效能感这一维度上， $\text{sig}=0.006(<0.05)$ ；而在内在动机和课程整合方面， sig 值分别为 0.931 和 0.188。可以看出在参与培训前后，教师对计算思维的理解和自我效能感发生了显著的变化，显著提高；而在内在动机及计算思维的课程整合方面没有显著的影响。

表 1 计算思维感知前后测配对样本 t 检验结果

维度	类别	人数	均值	标准差	t	Sig.(双侧)
CT 的理解	前-后	16	-0.334	.322	-4.142	.001
	自我效能	前-后	16	-0.562	.704	-3.195
内在动机	前-后	16	-0.019	.856	-0.088	.931
	课程整合	前-后	16	-0.188	.544	-1.379

6. 讨论

本文探索了如何设计有关计算思维的教师培训课程，企图让教师明确计算思维的要素以及整合到课堂中来，是培养学生计算思维的重要一步。通过结果分析，初步证明了该培训课程对于教师的计算思维的感知上的影响，教师对计算思维的概念和对学生的意义有了更加深刻的理解、通过短期的培训使得教师相信自己能做好此方面的工作，但是内在动机没有变化，可能跟培训的时长较短有关系。其次，就计算思维与其他课程的整合而言，由于目前国内缺少优秀的整合案例，不能在培训过程中给教师以更加针对性地指导，只能教师之间讨论重构课程的难处和方法，这在一定程度上限制了教师对于课程整合的态度和能力发展。

通过访谈可知，大多数教师更喜欢参与性、互动性比较强的培训课程，如果在学习的过程中，动手实践并有适当的产出，则会大大激发教师们的学习热情；他们非常认可培训课程中所用到的教学工具—Swift Playgrounds，认为其比较适合中小学生学习编程，弥补了目前可视化、拖拽式工具的不足，为学生之后过渡到纯代码层面的编程很有帮助。虽然国家在大力提倡人工智能、强调计算思维对于学生发展的重要性，但是就计算思维的培养而言，目前大多数教师的知识储备和认识都有待更新，只有通过系统地学习，才能更好地推动落地。除此之外，一些教师表示，领导不重视、课时有限以及缺少相应的软硬件的支持，都是限制因素。就培训课程而言，提高教师的参与热情、提供更多的教师交流平台和相关的教学资源是改进此课程的方向之一。针对软硬件皆能保障的学校，本着教师自愿报名参与的原则，发展第一批“种子教师”，试点尝试利用本次培训中学到的有关计算思维课程实施的策略和方法进行开课，并且将学生参与课程的反馈、教学反思和经验等分享给其他教师，这是后续培训课程的发展方向，也是本研究未来的研究方向。

7. 參考文獻

- 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.
- Blum, L. (2007). CS4HS: An Outreach Program for High School CS Teachers. 收入 *Teachers*”, *Proceedings 38th ACM Technical Symposium on Computer Science Education*.
- Lye, K. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61.
- Prietorodriguez, E., & Berretta, R. (2014). Digital technology teachers' perceptions of computer science: It is not all about programming. *Frontiers in Education Conference* (pp.1-5). IEEE.
- Werner, Denner, Campe, & Chizuru Kawamoto. (2012). The fairy performance assessment: Measuring computational thinking in middle school. SIGCSE'12 - Proceedings of the 43rd ACM Technical Symposium on Computer Science Education.
- Wilson, A., Hainey, T., & Connolly, T. (2013). Using Scratch with Primary School Children: An Evaluation of Games Constructed to Gauge Understanding of Programming Concepts. *International Journal of Games-based Learning*, 3, 93–109.
- Wing, J. (2006). Computational thinking. *Communications of the ACM*.
- Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. 收入 *ACM Technical Symposium on Computer Science Education* (頁 465–470).

國小師資生 Code.org 運算思維課程實作與成效探討

李政軒^{*}，楊智為，郭伯臣
教育資訊與測驗統計研究所
臺灣臺中教育大學

chenghsuanli@gmail.com, yangcw@mail.ntcu.edu.tw, kbc@mail.ntcu.edu.tw

摘要

本研究在師資培育大學中，實施了運算思維課程並探討師資生的學習成效。運算思維是 21 世紀中，教師必須要學習和教育兒童的重要技能之一。在臺灣十二年國教中，也強調其重要性並制定對應的能力指標。故影響了近來老師培訓和師培教育。然而，懂得如何教運算思維的教師甚至能將運算思維融入課程的教師仍為少數。本研究利用 Code.org 運算思維課程，培訓師資生 12 小時，讓其了解運算思維定義，並透過可視化代碼和圖形編程模塊進行練習。根據 15 位師資生前測與後測結果，發現前後測成績有顯著差異。因此應用 Code.org 運算思維課程能有效地提高運算思維的學習成效。

關鍵字

運算思維；Code.org；師資生；國民小學。

1. 前言

運算思維(Computational Thinking)是一種問題解決的思考過程，透過「問題分解(Problem Decomposition)」來將複雜和開放性問題分解成數個較小但較容易處理的小部分來闡述問題，並透過「演算法設計和程序(Algorithm Design and Procedure)」的開發，解決這些小部分問題的系統方式與電腦可以處理的步驟，告訴電腦如何去處理並協助解決問題。運算思維幾乎可以協助所有學科的解決問題，包括數學，科學和人文科學。在課程中學習運算思維的學生可以開始看到學科之間以及學校和課外生活之間的關係(Wing, 2006; Wing, 2011; ISTE & CSTA, 2011; Barr & Stephenson, 2011; Lee, Martin, Denner, Coulter, Allan, Erickson, Malyn-Smith, & Werner, 2011; Aho, 2012; Grover & Pea, 2013; Google, 2016)。

國際教育技術教育協(The International Society for Technology in Education, ISTE)，計算機科學教師協會(Computer Science Teachers Association, CSTA)和英國計算學校(The UK Computing at School, CAS)工作組與教育和工業代表合作，為教育工作者開發運算思維的教學資源(ISTE & CSTA, 2011; CAS, 2014; ISTE, 2016; Google, 2016)。美國總統歐巴馬也在 2014 年親自錄製影片宣傳「Hour of Code」，期藉由程式設計輔助運算思維所需的認知任務工具並展現運算思維能力(林育慈、吳正己, 2016)。由此可知，許多國家都紛紛推展運算思維，顯示運算思維的重要性。

在臺灣國家教育研究院「十二年國民基本教育課程綱要國民中小學暨普通型高級中等學校科技領域草案」

中也提到，資訊科技課程發展需關照科技與科學、數學、社會、藝術領域間的統整，課程設計將以「運算思維」為主軸，透過電腦科學相關知能的學習，培養邏輯與系統化思考等「運算思維」能力，並透過資訊科技之設計與問題實作，提升學生「運算思維」的應用能力、問題解決能力、團隊合作能力與創新思考能力(國家教育研究院, 2016a)。

故本研究在臺灣師資培育大學中，利用 Code.org 運算思維課程，針對師資生(具有修習師資職前教育課程資格的學生)進行 6 週共 12 小時的課程，再搭配前測與後測，來了解師資生在運算思維技能的提升狀況。

2. 文獻探討

2.1. 運算思維

隨著電腦與網路科技的快速進步，目前資訊的大量流通與改變快速，培養學生擅用電腦和網路，是新時代教育的一個重要內容。今天的資訊科技，透過機器學習、人工智慧的技術，打造出具有智慧的機器人，可以回答艱澀問題，甚至 AlphaGo 以 4:1 的成績戰勝人類頂尖棋手李世石。但這些技術是人類科學家演算法和智慧的結晶，電腦通過深度學習技術，在大數據的支撐下，可以做出更理性和精準的判斷，然而這一切，都是利用程式設計來實現。程式設計思維就猶如智力體操，並非要栽培未來的程式設計師，而是為了培養孩子的運算思維，開拓更寬廣的學習途徑，學習創意思考、有系統的推論、團隊合作並借助電腦實作解決問題能力，這些技能不僅在各專業領域都受用無窮，更是生活中不可或缺的能力。(劉耘, 2013; 洪士灝, 2016; 葉丙成, 2016; 尚吉剛, 2016)。

Zhong, Wang, Chen, & Li (2016)分析了運算思維定義提出三個觀點，分別為運算思維是問題解決的過程、形式的表達與三維架構(Brennan & Resnick, 2012)包含「概念(Concepts)」、「實踐(Practices)」與「視野(Perspectives)」(圖 1)。

A. 概念：學生在設計演算法會用到的程式概念，包含「物件(Objects)」、「指示(Instructions)」、「序列(Sequences)」、「迴圈(Loops)」、「事件(Events)」、「條件(Conditionals)」與「運算(Operations)」。

B. 實踐：學生在利用概念設計與測試演算法會用到的步驟，包含「抽象化(Abstraction)」希望能從具體問題或特定實例中萃取理解和解決問題的相關資訊、基本要素、共同特徵或動作來簡化問題。(Wing, 2006; ISTE & CSTA, 2011; Barr & Stephenson, 2011; Lee et al, 2011; Grover & Pea, 2013; Google 2016)、「演算法設計和程序

(Algorithm Design and Procedure) 」透過建立一系列有序的指令建立解決問題的工具來執行任務以達成部分目標(ISTE & CSTA, 2011; Grover & Pea, 2013; Google, 2016)、「資料表示(Data Representation) 」指能了解不同格式資料,有邏輯地組織和分析資料,並採用適當的圖形、圖表、文字或圖像來描述和組織數據(ISTE & CSTA, 2011; Barr & Stephenson, 2011; Google, 2016)、「問題分解(Problem Decomposition) 」能將資料、處理過程或問題分解成數個較小可管理且容易處理的部分(Wing, 2006; Barr & Stephenson, 2011; Google, 2016)、「模式辨識和一般化(Pattern Recognition and Generalization) 」透過觀察資料模式,找出資料趨勢和規律,來建立模型、規則、準則或理論,仿照現實狀況建立模擬資料,並使用條件、迴圈、遞迴或迭代方式來藉此驗證一般化的預測結果與解決問題(ISTE & CSTA, 2011; Google, 2016)。

C. 視野：學生透過表達、連結與詢問來形成對於現實世界問題的觀點,包含「創意和表達(Creative and Expressing)」、「溝通和合作(Communicating and Collaborating)」、「理解和質疑(Understanding and Questioning)」。

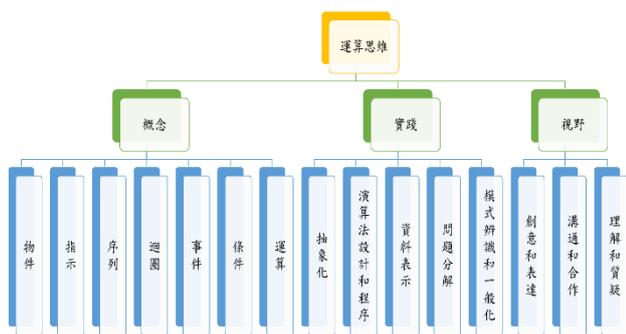


圖 1 本計畫採用之運算思維三維度架構

2.2 Code.org 課程

美國非營利組織 Code.org 發起的編寫程式活動「Hour of Code」(圖 2),讓學童透過活潑有趣的方式,在程式遊戲中嘗試、探索與創造,培養運算思維、想像力以及解決問題的能力,並在熟悉程式設計的基本概念同時,讓學童提早具備面對國際競爭的能力。小朋友可以用視覺程式語言,透過簡單的搬移拖拉方式,來建立演算法的區塊,學習用邏輯性的語言來思考與解決問題(Code.org Teacher Community, 2016; Google, 2016)。



(a) 課程操作畫面



(b) 堆疊後的程式碼

圖 2 Hour of Code 中的 Write your first computer program 課程

Code.org 針對不同年齡層的學生提供完整的運算思維課程,其中「課程 1」是針對年齡 4-6 歲的學生,裡面的程式區塊,大多是採用可視化的圖形介面,降低學生的閱讀負擔。課程 2 是適用於年齡 6 歲以上的學生,除了學生須具備基本閱讀能力外,也提高了區塊的使用難度,例如增加轉向的概念。課程 3 則是課程 2 的延續課程,其年齡設定為 8~18 歲的學生。課程 4 則是課程 2 與課程 3 的進階課程,裡面包含更複雜的程式設計概念,還包括帶有參數的函式。



圖 3 Code.org 課程 1 的部分課程介面

除此之外,Code.org 也針對教師角色進行介面設計,學生可以透過課程代碼,直接與老師指定課程連結。老師因此可以看到學生的學習進度,作答狀況,甚至給予學生測驗或額外課程。

目前已經有研究顯示,在 K12 學生採用 Code.org,並發現學生在使用 Code.org 課程訓練運算思維能力時,對於程式語言會抱持比較正面的態度,且男女生之間的能力相當(Kalelioglu, 2015)。有鑑於 Code.org 的課程已經提供 4-18 歲學生相關的課程,因此第一線的老師可以很容易利用相關課程來提升 K1~K12 學生運算思維能力。因此,本研究認為讓目前師資生了解何為運算思維、提升其運算思維能力、使用 Code.org 教授運算思維利,是目前師資培育大學重要的課題之一。

3. 研究方法

本研究的參加對象為臺灣中部某一所師資培育大學,在教育學群的選修課程「教學媒體與運用」導入 6 週共 12 小時的運算思維課程,共有 15 位學生完整參與課程

與前後測驗。運算思維課程是根據 Code.org 課程 3 提供的單元內容，包含「運算思維介紹」、「函式」、「條件判斷」、「巢狀迴圈」、「條件迴圈」與「除錯」等。

由於本課程之目標除了加強師資生運算思維技能外，也需要培養師資生教授運算思維課程的能力。因此，老師在授課時，除了讓學生進行 Code.org 課程 3 課程外，也需而外花時間教導學生如何利用 Code.org 開設課程，與觀看學生學習狀況，如學生學習進度、回應、評量/調查等。

表 1 部分前後測試題與其對應的到運算思維概念與實踐技能對照表

試題編號	概念								實踐			
	物件	指示	序列	迴圈	事件	條件	運算	抽象化	演算法設計和程序	資料表示	問題分解	模式辨識和一般化
1			V	V								
2										V		
4						V						
5									V			
8				V								V
9				V			V					
10				V		V	V					
11				V		V	V					
13								V				
14									V			
15											V	
16						V	V					
22						V	V					
23												V
24				V								
25									V			

4. 實驗結果

本研究使用之前測與後測試卷的信度 Cronbach α 值分別為 0.800 與 0.835，皆高於 0.8，故前後測試卷屬於高可信度。因此，本研究採用 15 位學生在 25 題式題的前測與後測總分來進行成對 t 考驗分析，藉此探討師資生透過 Code.org 課程 3 學習後，其運算思維概念與實踐技能是否有提升。

在課程實施前，本研究根據運算思維定義中的概念與實踐(圖 1)，並參考國際運算思維挑戰賽，設計了兩套互為複本的測驗。其中一套為前測測驗，待完成 6 週課程後，在進行後測測驗。每套測驗共有 25 道式題，作答時間設定為 60 分鐘。為了能夠判斷學生運算思維成績的題生狀況，會要求學生認真作答，若不會作答，請跳過該題，不要硬猜答案。表 1 為本研究開發部分前後測試題與其對應的到運算思維概念與實踐技能對照表。

表 2 為 15 位學生的前後測成對 t 考驗分析表，其中 15 位學生的前測平均為 57.6 分，後測平均為 63.2 分，所以後測平均與前測平均差為 5.6 分，其 t 值為 1.86， p 值為 0.04 小於顯著水準 0.05。

表 2 前後測之成對 t 考驗分析表

前測平均	後測平均	後測平均 - 前測平均	t	顯著性 (單尾)
57.6	63.2	5.6	1.86*	0.04

5. 結論

本研究將 Code.org 運算思維課程導入師資培育課程，希望能藉此提升師資生運算思維能力，並讓師資生了解如何透過 Code.org 課程來教授學生運算思維技能。透過 6 週共 12 個小時的教學，初步分析結果顯示，師資生在運算思維的能力有顯著進步($t=1.86^*$)。故建議師資培育大學可以考慮將 Code.org 導入課程之中，除了能夠教導師資生運算思維概念與提升師資生運算思維能力外，也可以培養師資生透過 Code.org 課程來教授運算思維的能力。

這個研究為初步研究，在課程實施過程中，一開始師資生接觸可視化代碼和圖形編程模塊進行運算思維課程時，是具有高度興趣的。但隨著重複在電腦上的課程訓練，師資生在學習上會漸漸對這樣的課程失去興趣，產生疲乏。故未來建議可以在課程當中增加一些不插電的活動，提高老師與師資生之間的互動，維持師資生的學習興趣。

6. 致謝

本研究由臺灣科計部計畫編號 MOST 106-2511-S-142-003-MY3 補助支持，特此誌謝。

7. 參考文獻

- 尚吉剛(2016年12月6日)。從《我的世界》到《西部世界》，代碼將構建未來。取自 <http://it.sohu.com/20161206/n475107684.shtml>
- 林育慈、吳正己(2016)。運算思維與中小學資訊科技課程。國家教育研究院教育脈動電子期刊，6。取自 <http://pulse.naer.edu.tw/Home/Content/6fe1eedf-10a1-4e1e-890e-dbbec8ce0647?paged=1&insId=40977899-d342-4f01-94a7-66d446c9d3bb>

- 洪士灝(2016年4月24日)。**創客時代的計算思維與科技教育**。火箭科技評論。取自 <https://rocket.cafe/talks/75328>
- 國家教育研究院(2016a)。**十二年國民基本教育課程綱要國民中小學暨普通型高級中等學校科技領域(草案)**。教育部。
- 葉丙成(2016年6月)。**葉丙成：學程式者，能成麒麟之才？親子天下**。
- 劉耘(譯)(2013年2月14日)。**人人都該學程式設計**。TEDxTaipei。
- Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, 55, 832-835.
- 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 archive, 2(1), 48-54.
- 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.
- CAS (2014). Computational Thinking. Retrieved from <http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/>
- Code.org Teacher Community (2016, July 13). Try Pair Programming—track the progress of multiple students using one computer! Retrieved from <http://teacherblog.code.org/post/147349807334/try-pair-programmingtrack-the-progress-of>
- Google (2016). Games for tomorrow's programmers. Retrieved from
- Grover, S., & Pea, R. (2013). Computational thinking in K-12: Review of the state of the field. *Educational Researcher*, 42(1), 38-43.
<https://blockly-games.appspot.com/?lang=en>
- ISTE & CSTA (2011). *Computational thinking: Teacher resources (2nd ed.)*. Retrieved from http://csta.acm.org/Curriculum/sub/CurrFiles/472.11C_TTeacherResources_2ed-SP-vF.pdf
- ISTE (2016). COMPUTATIONAL THINKING FOR ALL. Retrieved from <https://www.iste.org/explore/article/detail?articleid=152>
- Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior*, 52, 200-210.
- Lee, L., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32-37.
- Wing, J (2011). *Research notebook: Computational thinking-What and why?* The Link Magazine, Spring. Carnegie Mellon University, Pittsburgh.
- Wing, J. (2006). *Computational thinking*. *Communications of the ACM*, 49(3), 33-35.
- Zhong, B.C., Wang, Q.Y., Chen, J., & Li, Y. (2016). An Exploration of Three-Dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research*, 53(4), 562-590.

Designing Computational Thinking Assessment: A Case Study of a Pre-Service Teacher Course in Korea

Mi Song KIM^{1*}, Hyungshin CHOI²

¹University of Western Ontario

²Chuncheon National University of Education

mkim574@uwo.ca, hschoi@cnue.ac.kr

ABSTRACT

This case study reports a pilot study of designing computational thinking (CT) assessment instruments in a pre-service teacher course in Korea. We describe the implementation of a CT course for pre-service teachers who did not major computer science. We report two instruments: a survey and a team project guideline. The results suggest that two assessment instruments have the potential to help pre-service teachers gain self-confidence and become motivated to incorporate CT concepts across all disciplines.

KEYWORDS

computational thinking, assessment, pre-service teachers, multimodal representation

1. INTRODUCTION

With the ever increasing need for teaching computational thinking (CT) to learners of the digital age, teacher educators need to develop a curriculum to enable teachers and teacher candidates “to better conceptualize, analyze, and solve complex problems by selecting and applying appropriate strategies and tools” (Computer Science Teachers Association, 2011, p. 9). In this light, much attention has been paid to the design of K-12 CT curricula in many countries including South Korea (Heintz, Mannila, & Farnqvist, 2016) as we are discovering the positive effects of computer programming in K-12 education. However, it has been a challenge to better prepare pre-service teachers to embed CT activities across subjects and contexts (Kazakoff & Bers, 2012). To address this challenge, this case study aims to design and implement CT assessments for Korean pre-service teachers who did not major computer science.

2. LITERATURE REVIEW

Computational thinking (CT) was first used by Papert (1996) in an article about mathematics education. However, a definition for this term was not provided until years later when Wing (2006) mentioned it to entail “solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (p. 33). With the clear rise in the importance of CT, many countries are introducing computing as a core curriculum subject (Heintz, Mannila, & Farnqvist, 2016).

However, bringing CT into teacher education is at its early stages of development and lacks curriculum studies to design teacher education (Yadav et al., 2011) and assess the development of CT (Brennan & Resnick, 2012). Although CT is considered to be critical 21st competencies, little is known about how to assess CT expertise development (Lye & Koh, 2014).

3. THE STUDY & METHOD

This case study was part of a series of design-based research, and in this paper, we report only designing CT assessment instruments for pre-service teachers at a national university of education in Korea. In order to design a survey instrument, we have incorporated the five sub-components of CT derived from a meta-analysis conducted by Selby and Woollard (2010). We have also added categories to make it applicable for pre-service teachers’ CT courses (i.e., programming course, problem solving via CT, etc.).

Further, we have developed ‘a team project guideline’ for pre-service teachers when they present their team projects (i.e., animations, games, quizzes, etc.) based on their understanding of core CT concepts. This team project guideline aimed to help pre-service teachers to reveal their comprehension of CT explicitly and to collaboratively reflect on their CT team projects.

4. RESULTS

The survey instrument with 15 items on a 4-point Likert scale (1 = strongly disagree, 2 = disagree, 3 = agree, 4 = strongly agree) was developed to assess CT skills. There are three categories in the survey: the degree of experiencing CT during the course, self-efficacy of teaching CT, and CT transfer. Each category has five items pertaining to five sub-components of CT: algorithmic thinking, evaluation, problem decomposition, abstraction, generalization. For example, self-efficacy includes “if I teach elementary students Scratch programming in the future, I would be able to help them to solve problems with algorithmic thinking”.

Overall, pre-service teachers reported positive experiences in terms of high level of CT concepts, self-efficacy and prospective use of CT. More detailed results will be reported somewhere else. A team project guideline was developed for pre-service teachers to reveal their ability to think computationally while preparing for a presentation of their programming projects.

The guideline included the five sub-components of CT (algorithmic thinking, evaluation, problem decomposition, abstraction, generalization) as well as a description of the problem, sprites (or images), background, variables, roles of team members and reflection. Figure 1 shows an example created by a team from a preliminary study using Scratch programming: breaking the problems into smaller problems and defining each smaller problem.

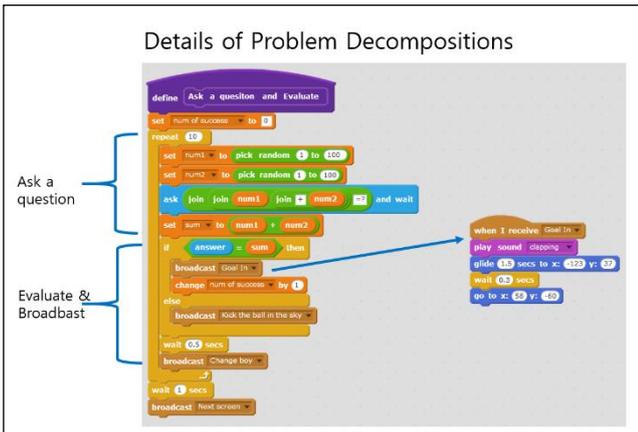


Figure 1. Scratch team projects.

5. CONCLUSION & DISCUSSION

The results suggest that two assessment instruments have the potential to help pre-service teachers gain self-confidence and become motivated to incorporate CT concepts across all disciplines. The instruments were designed to assess the impact of the CT instruction using Scratch programming for pre-service teachers who did not major computer science. In particular, collaboratively incorporating the team project guideline in a team allowed pre-service teachers to critically reflect on their learning progress and intensify collaborative efforts. Further, in addition to written language, they effectively incorporated multimodal representation (e.g., visual images) to communicate their CT concepts. Drawing upon this finding, we will continuously design another cycle of design-based research to initiate a student-generated rubric for CT assessment to promote student agency, collaboration, and multimodal representation.

6. REFERENCES

- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Paper presented at the American Educational Research Association. Canada: British Columbia
- Computer Science Teachers Association. (2011). *CSTA K-12 computer science standards*. Retrieved from https://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf
- Heintz, F., Mannila, L., & Farnqvist, T. (2016). *A review of models for introducing computational thinking, computer science and computing in K-12 education*. Paper presented at 2016 IEEE Frontiers in Education Conference.
- Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the Kindergarten Classroom. *Journal of Educational Multimedia and Hypermedia*, 21(4), 371-391.
- 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*, 21, 51-61.
- Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1). doi: 10.1007/bf00191473
- Selby, C. C., & Woollard, J. (2010). *Computational thinking: The developing definition*. SIGCSE 2014.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
- Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). *Introducing computational thinking in education courses*. Paper presented at the Proceedings of the 42nd ACM technical symposium on Computer science education.

Which Parts of Computer Science Concepts Do Future Teachers Identify? First Results of a Part-Whole-Thinking Analysis in Computer Science Education

Nils PANCRATZ*, Ira DIETHELM
Department of Computing Science
University of Oldenburg, Germany

nils.pancratz@uni-oldenburg.de, ira.diethelm@uni-oldenburg.de

ABSTRACT

The ability to detect Part-Whole-Relationships and to interconnect these to an organized structure is one of the core cognitive processes through which knowledge is acquired. However, the sharing of this capability, which belongs and relates to Computational Thinking skills and is called Part-Whole-Thinking, is lining up behind the conveyance of content in Computer Science classes and courses still. In order to support a more vigorous inclusion of Part-Whole-Thinking into Computer Science Education, various aspects need to be considered and investigated in the first place. The Model of Educational Reconstruction for Computer Science Education illustrates the elements to be taken into account when designing and arranging Computer Science lessons and courses. One of the elements under consideration is the investigation of the teachers' perspectives. The contribution at hand presents first results of an analysis of future teachers' Part-Whole-Thinking of Computer Science Concepts.

KEYWORDS

Part-Whole-Thinking, Computational Thinking, Computer Science Education, Teachers' Perspectives, Model of Educational Reconstruction

1. INTRODUCTION

Part-Whole-Relations play a decisive role in cognitive processes that are inevitably involved in understanding various objects, systems, processes, definitions, and concepts (Gerstl and Pribbenow, 1995). The essential ability of *Part-Whole-Thinking* (PWT) belongs to core concepts of *Computational Thinking* (CT) as originally defined by Wing (2006). Since many Information Technology devices make use of Part-Whole-Relationships, these need to be adequately included in explanations in *Computer Science* (CS) classes. Rao and Shafique (Rao, 2005; Shafique and Rao, 2006) could already successfully improve their students cognitive learning processes by including PWT in their CS courses. The benefits they noticed mainly included improved thinking skills in the students and an improvement in teaching skills (Rao, 2005). But as the lack of publications on this subject since 2006 shows, their attempt "to bring these issues to the notice of the computer science community" (Rao, 2005, p. 173) has been in vain. Many different aspects have to be considered when supporting a more vigorous inclusion of PWT or "the transfer of knowledge from research to the classroom" (Diethelm, Hubwieser, and Klaus, 2012, p. 164) in general. To illustrate these facets for *Computer Science Education* (CSE), Diethelm, Hubwieser, and Klaus (2012) extended the *Model of Educational Reconstruction*. One of the aspects they

included concerns the *investigation of teachers' perspectives*. This issue is especially important for the design and arrangement of CS lessons and courses, since CS teachers generally have very different educational backgrounds and qualifications (ibid., p. 167). They "regard the teachers' perspective as a key factor for the design of lessons as well as for educational research" (ibid., p. 167). One question they ask for is, which conceptions "the teachers actually apply to explain the chosen phenomena themselves" (ibid., p. 167).

The contribution at hand and the belonging poster present first results of an analysis of PWT in CSE. Questionnaires were filled out by 21 students of a CSE lecture at the University of Oldenburg, Germany. The students were asked which parts they identify of eight typical CS concepts. The following research questions were pursued during this specific research approach:

1. Which parts do future teachers identify of common CS concepts? To which extend are the parts identified correctly?
2. To which extend is the used method of asking for parts of concepts through questionnaires suitable for the purpose of investigating PWT?

2. METHODOLOGY

In this pilot study, questionnaires were designed to investigate the future teachers' perceptions. After three closed questions on the biographical background of the participants, an everyday example (parts of cars: tires, wheel, engine, bonnet, doors, ...) on the following task ("In the following you have to identify Part-Whole-Relationships of Computer Science concepts") was presented in the questionnaire. The CS concepts under consideration (cf. Tab. 1) were chosen through an analysis of the core concepts that are included in the CS curriculum of Lower Saxony, Germany (Niedersächsisches Kultusministerium, 2014). The participants had 25 minutes to answer the questions. In order to analyze the questionnaires, the answers were digitalized, translated from German to English, and normalized in the first place. The normalization included a combination of all abbreviations (e.g. combining the answers "PSU" and "power supply unit" to "power supply unit (PSU)") and synonymous listings (e.g. combining "provider" and "Internet provider" to "(Internet) provider") and a re-movement of plurals. Afterwards, the occurrences of identical listings of parts for each investigated concept were counted. In addition to that it was counted, how many parts each participant identified of each concept and an average for each concept was calculated (cf. Sec. 3). After this descriptive statistic analysis, the answers were checked

for content-related correctness on the meta-level by the authors (cf. Sec. 4).

3. RESULTS

An overview on the CS concepts under investigation, the amount of parts that each participant identified in average ($\varnothing_{\text{parts/person}}$), the number of various identified parts in total ($\#_{\text{various Parts}}$), and the number of parts that were identified by at least two respondents ($\#_{\text{id. sev. times}}$) is given in the following Tab. 1.

Table 1. Overview on the results

CS concept	$\varnothing_{\text{parts/person}}$	$\#_{\text{various parts}}$	$\#_{\text{id. sev. times}}$
Computer	4.3	29	20
Internet	4.0	40	16
Email	3.7	41	13
Automaton	3.4	40	13
Website	3.0	43	14
Algorithm	2.8	44	11
Database	2.5	29	10
Data	1.8	28	5

A detailed overview on the answers is presented in the poster, which interested readers of this contribution gladly will be provided with by request via email.

4. CONCLUSION AND DISCUSSION

In Tab. 1, the concepts are sorted in descending order according to the amount of parts that each participant identified in average. It can easily be seen that the less parts are identified by the students the more abstract, complex, and theoretical the concepts are: While “computer” — a physical device and concrete product — is the concept that the students identified the most parts of in average, they had issues with finding parts of “data” — which is a very theoretical and abstract concept in contrast. While this fact alone might not be that surprising, there is another interesting aspect that needs to be mentioned at this point: While all of the repeatedly identified parts of “computers” are completely reasonable, comprehensible, and correct, there are huge mistakes in the main parts that the students identified of the more theoretical and abstract concepts. For example, it is an obvious error that “information” is a part of “data”. Instead, data requires some sort of interpretation to get information. Similar obvious mistakes can be found for “algorithms” and “websites”: While “algorithms” are parts of “applications” and “methods” instead of the other way around — as identified by many students —, it is also wrong to say that “the Internet”, “servers”, and “browsers” are parts of “websites”. Another interesting aspect is the fact that the more complex the concepts are — excepting “database” and “data”¹ — the more various parts are identified. By analogy, the amount of several times identified parts decreases with increasing complexity and abstractness of the concepts. To describe it differently, these two facts mean that the students are more disagreeing on what parts the more complex and theoretical concepts consist of.

Generally speaking it seems as if the students of the investigated introductory CSE lecture had huge problems with the task of finding parts of complex CS concepts. Many students listed elements as parts that simply do not fit. Without a doubt, it is way more difficult to identify parts of “data” than “computers”. So, it is not at all remarkable, that the students listed less parts of the more abstract concepts than the concrete ones. However, it is quite worrying that they tended to give wrong answers when they were asked to identify parts of more complex CS concepts to a not negligible extent. At this point it is mentionable, that this lecture is intended to be attended by students in their fourth bachelor semester. So, a lack of knowledge on CS concepts is probably not the reason for the deficits that were found out in this study.

As already mentioned, Part-Whole-Relationships play a huge role in CS. PWT (mostly subconsciously) helps to understand objects, systems, processes, definitions and concepts. But surprisingly there is almost no literature available on infusing it into CSE (Rao, 2005). The only way to achieve this infusion is through the CS teachers. So, this study aimed at an investigation of future teachers’ PWT to make a start. Future work will lie on a deeper investigation of PWT in CSE alongside the Model of Educational Reconstruction. Therefore, a suitable research method will be designed in the first place, since deficits were seen with naively asking for an identification of parts of wholes through questionnaires.

5. REFERENCES

- Diethelm, I., Hubwieser, P., and Klaus, R. (2012). *Students, Teachers and Phenomena: Educational Reconstruction for Computer Science Education*. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*. ACM, 164–173.
- Gerstl, P. and Pribbenow, S. (1995). *Midwinters, end games, and body parts: a classification of part-whole-relations*. In *International Journal of Human-Computer-Studies*, 43(5), 865-889
- Niedersächsisches Kultusministerium (2014). *Kerncurriculum für die Schulformen des Sekundarbereichs I Schuljahrgänge 5 – 10*. Hannover, Germany: Niedersächsisches Kultusministerium
- Rao, K. (2005). *Infusing Critical Thinking Skills into Content of AI Course*. SIGCSE Bull. 37, 3. 173–177
- Shafique, M. and Rao, K. (2006). *Infusing Parts-whole Relationship Critical Thinking Skill into Basic Computer Science Education*. In *Proceedings of the FECS 2006, Las Vegas, Nevada, USA, June 26-29, 2006*. 287–292
- Wing, J. M. (2006). *Computational Thinking*. In *Communications of the ACM, March 2006*, 49 (3), 33-35

¹ This mainly results from the fact that every fourth student did not find any parts of these two concepts at all, though they both were positioned in the middle of the questionnaire.

General Submission to Computational Thinking Education

Developing a Framework for Computational Thinking from a Disciplinary Perspective

Joyce MALYN-SMITH¹, Irene A. LEE², Fred MARTIN³, Shuchi GROVER, Michael A. EVANS⁴, Sarita PILLAI¹

¹Education Development Center

²Massachusetts Institute of Technology

³University of Massachusetts Lowell

⁴North Carolina State University

jmsmith@edc.org, ialee@mit.edu, fred_martin@uml.edu,
shuchig@cs.stanford.edu, michael.a.evans@ncsu.edu, spillai@edc.org

ABSTRACT

This paper describes progress towards the development of a Framework for Computational Thinking (CT) from a Disciplinary Perspective. The work aimed at discovering how CT can be encouraged, taught and practiced within disciplines throughout primary and secondary education. It identifies an initial set of “elements” describing CT practices that bridge learning and working in highly sophisticated STEM environments and shares examples of these practices used by STEM professionals at work and developed by students in schools. It is hoped that this paper will provoke dialogue among educators advocating for CT as a core skill for all and will contribute to breakthroughs in thinking about how CT should be learned and assessed in and out of school.

KEYWORDS

Computational thinking, K-12 education, workforce development, human-technology frontier.

1. INTRODUCTION

The proliferation of new technologies has changed the way we live, learn, and work. Although the future of work is unclear, experts envision a new machine age, where technologies (sensors, communication, computation, and intelligence) are embedded around, on, and in us; where humans will shape technology and technology will shape human interaction; and where technologies and humans will collaborate to discover and innovate. In short—the Human-Technology Frontier.

Without question, the global workforce will need a new set of skills and competencies to succeed in the future work environments on this frontier—that feels closer with each new technological advance. A recent report by EDC’s STELAR Center (Malyn-Smith et al., 2017) identified computational thinking as one of the essential skills needed by future workers for success in work at the Human-Technology Frontier. As our society works to understand and identify strategies to overcome these complex and interrelated challenges, important questions include: What can we do to prepare today’s students to succeed in work at the Human-Technology Frontier? and What steps can we take to make this happen? If we are to believe that the Human-Technology Frontier is upon us, we need to reconsider how computational thinking is taught in order to advantage our students, not only in developing CT skills, but also in developing the CT practices used in STEM workplaces (EDC, 2011).

2. BACKGROUND

Since noted computer scientist Jeannette Wing (2006) proposed CT as a new “core skill” various groups have tried to define CT for education and training purposes (e.g. Grover & Pea, 2013, 2018). CT (focusing on problem-solving, algorithms, data representation, modeling and simulation and connections to other fields) is a prominent strand of the K-12 Standards for Computer Science developed by the Computer Science Teachers Association (CSTA, 2011). Individual states (including Massachusetts and New Jersey, USA) have instituted computer science (CS) and digital literacy standards that use the term CT. Next Generation Science Standards (NGSS Lead States, 2013) include computational thinking in one of their eight scientific practice standards. National Science Foundation (NSF) funded projects are conducting research on several different approaches to CT. Data practices, modeling and simulation practices, computational problem solving practices and systems thinking practices are proposed by Weintrop et al. (2016). Lee et al. (2011) propose that youth develop CT skills as they use, modify and create with digital tools and technologies. While these initiatives signal a broad based, grassroots interest in computational thinking, their simultaneous development and independent implementation leaves us without consensus on a precise definition of CT. (Barr & Stephenson, 2011; Voogt, Fisser, Good, Mishra, & Yadav, 2015; Weintrop et al., 2016). Most agree, however, that Computational Thinking is formulating problems and their solutions in a way that a machine (computer) can be used to represent the problem and carry out its solution.

What has emerged from these varied research and practice efforts aimed at CT is a debate over how CT is best taught and learned. Many computer science educators believe that CT is best taught through programming where students’ development of CT can be ensured and uniquely observed. Others believe that to best prepare today’s youth for tomorrow’s world, CT should be taught/learned in the service of disciplines. While many of the efforts described above define CT by dissecting it into its component parts, little has focused on what results from integrating CT and disciplinary learning. To guide teaching and learning of CT within the disciplines, a new kind of computational thinking framework was needed – one which captured and clarified what students were able to do using CT – and unable to do without CT.

3. DEVELOPING A FRAMEWORK

A group consisting of principal investigators, researchers, and educators from National Science Foundation funded ITEST (Innovative Technology Experiences for Students and Teachers) and STEM+C (STEM+Computing) projects convened in August and November 2017 to explore the development of an Interdisciplinary Framework for Integrating CT in K-12 Education. Their goal was to draft a framework defining computational thinking from a disciplinary perspective. The 54 workshop participants provided a good balance of researchers and practitioners, who represented grade spans Kindergarten-2nd grade, 3rd-5th grade, 6th-8th grade, and 9th-12th grade, as well as disciplines including science, mathematics, engineering, social science, computer science and the humanities. In total there were 31 researchers, 18 teachers / practitioners, 3 participant observers, and 2 staff members. (13 of the participants were from colleges/universities, 15 from schools, 15 from non-profits, 1 from business, 3 from foundations including the NSF). The primary goals were to develop a framework for computational thinking from a disciplinary perspective that built on the work of the foremost researchers and practitioners focused on helping youth develop CT skills. Progress towards the goals was guided by some of the foremost CT thought leaders in the U.S. including Irene Lee of Massachusetts Institute of Technology, Shuchi Grover, Fred Martin of University of Massachusetts Lowell and CSTA, and Michael Evans of North Carolina State University.

As a first step, participants were asked to submit examples of their work to share with other participants prior to the workshops. Educators/practitioners shared curriculum and activities that illustrated CT in action in their classrooms. Researchers shared their lessons learned through research on various aspects of CT skill development and integration. Together the group explored these examples and found that a number of common “elements” emerged. During the workshops, participants were asked to provide additional examples of CT integration by grade level and discipline. These examples were subsequently reviewed and discussed within the emerging framework of common elements.

Thought about the goal of developing a framework for CT in the service of disciplines crystallized around the larger goal of education – that of preparing youth for success for living, learning and working *after* compulsory education. Thus, focusing on building a bridge between the CT skills developed in school and the professional practices involving CT, particularly those in scientific workplaces became paramount.

A traditional way CT is integrated is shown at the bottom of Figure 1 illustrated with the Massachusetts digital learning and computer science (DLCS) standards component areas of abstraction, algorithms, programming and software development, data collection and analysis, and modeling and simulation. Typically, individual CT components are taught then linked in pairs and clusters leading up to potentially more powerful CT activities at with older age groups.

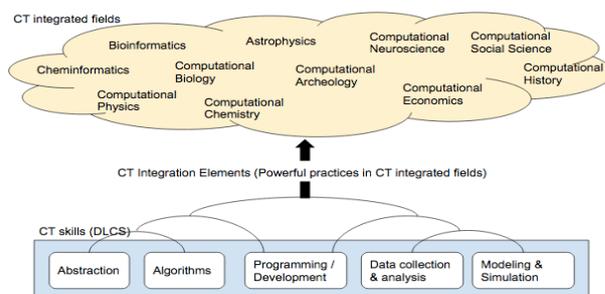


Figure 1. Bridging between traditional teaching of CT and CT as used in CT integrated fields.

Stronger connections between these CT components and the powerful practices used by professionals in CT-integrated scientific fields (e.g. computational biology, bioinformatics, cheminformatics, computational economics and others) were sought. The aim in making these connections was to ensure that the CT integrated in K-12 concept areas provided a strong foundation for the computational thinking used by practicing scientists and would bridge the skills transition from school to work.

4. CT from a Disciplinary Perspective – examples from STEM workplaces

To further explore the elements that might form a framework for CT from a disciplinary perspective, examples of CT commonly used by practicing scientists specifically, examples of what can be accomplished using CT that would be difficult, if not impossible, without CT were gathered. From these examples of CT used by practicing scientists in CT integrated fields, the elements emerged and were tested as organizers for other examples of CT. The initial examples considered follow.

4.1. Ensemble modeling

Scientists use multiple models are used to predict the behavior of complex systems. For example, weather forecasting now uses ensembles of models to understand weather patterns (Gneiting & Raftery, 2005; Krishnamurthy et al., 2000). Each model in an ensemble simulates the global weather system taking different sets of parameters or initial conditions into account. Instead of making a single forecast of the most likely weather, a set (or ensemble) of forecasts is produced. This set of forecasts aims to give an indication of the range of possible future states of the atmosphere.

4.2. Computational chemistry

Scientists innovate with computational representations - For example, the SMILES (simplified molecular-input line-entry system) notation is a representation for describing the structure of chemical compounds using short ASCII strings (O’Boyle, 2012). This revolutionized computational chemistry and drug design by enabling computers to read and operate on chemical sequences (including searching and database indexing).

4.3. Bioinformatics

CT is used in bio-informatics workplaces. In Next Generation Sequencing Data Analysis, dozens of whole genomes can be sequenced in rather short time, producing huge amounts of data (McKenna et al., 2010; DePristo, et

al., 2011). Complex bioinformatics analyses are required to turn these data into scientific findings. To run these analyses quickly, automated workflows on high performance computers are state of the art. Scientists design processes to achieve high throughput processing of genomic data.

4.4. Environmental science

Environmental scientists use crowd-sourced data in water management (Fienen & Lowry, 2012; Stepenuck & Green, 2015; McKinley et al., 2015). When considering water management strategies for a region, data for various communities with different water usage and needs (for example, for growing different crops or industrial uses) is necessary to understand the larger picture of water usage and needs, as well as the local variations.

4.5. Machine learning

To a larger and larger extent, scientists are using machine learning to make predictions. In supervised machine learning, scientists build models by running algorithms on “training sets” of inputs matched with correct responses (Srivastava et al., 2014; Lecun, Bengio, & Hinton, 2015). These models can then be used to offer predictions (or responses) when given new inputs. Changes in the training set data can have implications on the machine learning model built and can introduce biases if the training data is not representative of the target.

5. The Elements of CT integration from a Disciplinary Perspective

The examples from advisors and researchers along with lessons and activities provided by educators were examined. Evidence was found that K-12 subject area teachers were integrating CT in ways that were consistent with its use in CT-integrated fields. The following five Elements of CT Integration from a Disciplinary Perspective that emerged from the reviews and discussions were:

1. Understand (complex) systems.
2. Innovate with computational representations.
3. Design solutions that leverage computational power/resources.
4. Engage in collective sense making around data.
5. Understand potential consequences of actions.

5.1. Understand complex systems

Modeling how interactions of many individuals or components in a system lead to aggregate level emergent patterns is difficult to do without CT. Complex systems in particular are not amenable to traditional mathematical analysis. Simulating a system’s change over time and real-time feedback in the form of simulations help scientists visualize complex systems dynamics. These systems are often hard to predict due to having a multitude of interrelated factors and levels. In K-12 education, computer modeling and simulation of these systems offers a way to see how the systems behave under different circumstances, with different inputs.

5.2. Innovating with computational representations

The design and development of innovations is made possible through CT. New ideas, conceptualizations, representations,

and processes can be thought of and developed as computations. For example, thinking of the brain as a network and creating neural networks as artificial brains has led to advances in artificial intelligence and cognitive science. In K-12, students can be introduced to computational representations by learning about how colors are represented on computers as RGB values.

5.3. Design solutions that leverage computational power and resources

Scientists working with large data sets or on computationally intensive calculations design solutions that leverage the efficient use of resources and computational power to optimize their time. In some cases, distal collaborators can pool and share computational resources and in other cases co-located collaborators can access distributed resources to achieve their goal. Some speedups are achieved by decomposing datasets and/or processes to run in parallel. In K-12 settings, educators can challenge students to think about how they would solve a problem differently if the input set was of large scale. For example, rather than developing processes to assemble 10 finished copies of an item, how would students go about assembling 10,000 copies?

5.4. Engage in collective sense making around data

Data sets can be amassed through crowd-sourcing or collection by multiple individuals or sensors. These data can be analyzed to uncover patterns. Visualization of multidimensional data enables students to see patterns that might not otherwise be apparent. When possible in the K-12 education setting, teachers can ask small groups of students to run simulations on a subset of the inputs, then share their output data and analyses. Gathering and analyzing the combined data illustrates how each part of the data contributes to the understanding of the whole.

5.5. Understand potential consequences of actions

Scientists envision the future through simulation and use machine learning to make predictions. Using parameter sweeping, the space of all possible combinations of inputs can be tested to see the variety and probability of outcomes. In K-12, students can learn how cause and effect relationships can be used to predict outcome. Students can also begin to understand the space of inputs created by parameterizing models.

Notably, these elements of CT integration go *beyond* the mechanics of learning to program a computer. They form a bridge between CT as it has traditionally integrated in K-12 classrooms (through the introduction of computer programming activities) and professional practices.

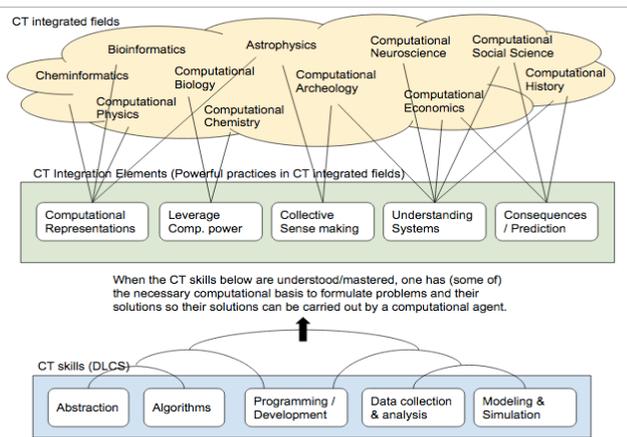


Figure 2. CT integration elements as a bridge between traditional CT integration in K-12 education and CT as powerful practices used in CT integrated fields.

Figure 2 illustrates how the thinking progressed from the idea of direct teaching of CT skills through programming - to a realization that to help students develop CT skills through STEM disciplinary learning, their education needs to include a stronger focus on computational tools, techniques, and processes used in the CT integrated fields.

6. CT from a Disciplinary Perspective – examples from K-12 classroom teachers

Through the examination of lessons provided by K-12 educators, it was determined that a subset of the disciplinary teachers were already integrating CT within K-12 that aligned with the elements presented above. Several lessons and activities teachers provided from their curricula illustrate how these elements can be introduced in K-12 to help students develop CT skills aligned with professional practices.

6.1. Middle school science

In middle school ecosystems lessons (Lee, 2011; Project GUTS, 2014) using the StarLogo Nova modeling and simulation environment, middle school students in science classrooms used, modified and created computer models and ran simulation to understand complex systems; multiple models were produced and compared; students engaged in collective sense making around data (by crowdsourcing data generated from multiple runs of each of the models); and students learned about potential consequences of actions (such as the impact of removing a top predator).

6.2. Elementary school mathematics

In a 5th grade mathematics classroom, students were asked to generate a language to describe a minimal set of actions to be performed by robots tasked to build a tower. Within this activity students were innovating with computational representations, and designing solutions that leverage how computers process data (in this case, instructions).

6.3. High school engineering

In a high school engineering classroom, a teacher used a multi-step physical construction task to illustrate domain vs. task decomposition as method of parallel processing in high performance computing. Students designed processes to make many copies of a Lego figure that leveraged

“processing” resources (other students) then optimized the design based on collective sense making from data on time to complete the task.

6.4. Middle school mathematics

In a middle school mathematics classroom, students using the iSENSE data-sharing platform were able to collect and add locally generated data to a large student-generated data set. They could then analyze their data and compare it to data provided from other classrooms (Willis et al., 2015).

6.5. Across subject areas

There is a large window of opportunity for K-12 students to learn about consequences of actions, in areas ranging from cause and effect in programming to decision-making and prediction in machine learning.

7. CHALLENGES

While the path towards CT integration from a disciplinary perspective is growing clearer, many challenges remain. First, we acknowledge that the majority of K-12 teachers are still struggling with the integration of CT in terms of teaching the basics of computer programming. Introducing the elements of CT integration can be viewed as a conflicting definition instead of a further elaboration on a trajectory of CT from K-12 to professional practice.

Another challenge is the rate at which fields are innovating with CT. The examples of CT integrated fields presented in this paper are only a few of the many fields that have been greatly impacted by CT. Many additional fields are incorporating computational tools, techniques, and practices. Across fields, innovations and discoveries made possible by the integration of computational tools, techniques, and practices are increasing.

The rapid rise of machine learning raises yet another challenge. Across disciplines, the need for analysis of computational systems, especially those used to make predictions that greatly impact human life, is paramount. The inclusion of the CT integration element “*Understanding potential consequences of actions*” addresses this important need.

8. CONCLUSION

The authors believe that learning CT needs to extend beyond learning to program. It must include engagement in computational practices used in the sciences that harness the power of computers to enhance scientific discovery. The CT Integration Elements presented here provide a framework for foundational learning of CT within disciplines beginning in elementary school and extending through high school and beyond. Examples provided by K-12 teachers shed light on ways K-12 educators have integrated powerful practices from professional CT integrated fields. It is hoped that the framework can aid teachers in the development of CT lessons, and ensure that the CT that teachers promote has links to the CT used in scientific workplaces. Still, this Framework is a work-in-progress. It is hoped that it will evolve as researchers continue to examine—and K-12 educators increasingly engage in—CT integration in the classroom.

9. ACKNOWLEDGMENTS

We gratefully acknowledge the support of the “Workshop to Develop an Interdisciplinary Framework for Integrating Computational Thinking in K-12 Science, Mathematics, Technology, and Engineering Education” project from the National Science Foundation (DRL award# 1647018). We’d also like to thank the many participants in the workshop series who generously shared their curricula, research, and insights.

10. REFERENCES

- 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.
- Computer Science Teachers Association (2011). K-12 computer science standards. <http://csta.acm.org/Curriculum/sub/k12standards.html>
- DePristo, M. A., Banks, E., Poplin, R. E., Garimella, K. V., Maguire, J. R., Hartl, C., Philippakis, A. A., del Angel, G., Rivas, M. A., Hanna, M., McKenna, A., Fennell, T.J., Kernytsky, A.M., Sivachenko, A.Y., Cibulskis, K., Gabriel, S.B., Altshuler, D., & Daly, M. J. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nature Genetics*, 43(5), 491–498.
- EDC (2011). *A Profile of a Computational Thinking Enabled STEM Professional in America’s Workplaces – Research Scientists / Engineers*. (revised 2013). Waltham, MA: EDC.
- Fienen, M.N., & Lowry, C.S. (2012) Social Water—A crowdsourcing tool for environmental data acquisition, *Computers & Geosciences*, 49(1), 164-169.
- Gneiting, T., & Raftery, A. E. (2005, October). Weather Forecasting with Ensemble Methods. *Science*, 310(5746), 248-249.
- Grover, S. & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*. 42(1), 38-43.
- Grover, S. & Pea, R. (2018). Computational Thinking: A competency whose time has come. In *Computer Science Education: Perspectives on teaching and learning*, Sentance, S., Carsten, S., & Barendsen, E. (Eds). Bloomsbury.
- Krishnamurti, T. N., Kishtawal, C. M., Zhang, Z., Larow, T., Bachiochi, D., Williford, E., Gadgil, S., & Surendran, S. (2000). Multimodel Ensemble Forecasts for Weather and Seasonal Climate. *Journal Of Climate*, (13). 2000 American Meteorological Society.
- Lecun, Y., Bengio, Y., & Hinton, G. (2015, May). Deep learning. *Nature*, 521, 436–444.
- Lee, I., Martin, F. Apone, K. (2014). Integrating Computational Thinking Across the K-8 Curriculum. *ACM Inroads*, 5(4): 64-71.
- Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., and Werner, L. (2011). Computational thinking for youth in practice, *ACM Inroads*, Vol. 2 No.1.
- Malyn-Smith, J., Blustein, D., Pillai, S., Parker, C. E., Gutowski, E., & Diamonti, A. J. (2017). *Building the foundational skills needed for success in work at the human-technology frontier*. Waltham, MA: EDC.
- McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., & DePristo, M. A. (2010). The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20(9), 1297–1303.
- McKinley, D. C., A. J. Miller-Rushing, H. L. Ballard, R. Bonney, H. Brown, D. M. Evans, R. A. French, J. K. Parrish, T. B. Phillips, S. F. Ryan, L. A. Shanley, J. L. Shirk, K. F. Stepenuck, J. F. Weltzin, A. Wiggins, O. D. Boyle, R. D. Briggs, S. F. Chapin III, D. A. Hewitt, P. W. Preuss, and M. A. Soukup. (2015). *Investing in citizen science can improve natural resource management and environmental protection*. USGS Publications Warehouse. <http://pubs.er.usgs.gov/publication/70159470>
- NGSS Lead States. (2013). *Next Generation Science Standards: For States, By States*. Washington, DC: The National Academies Press.
- O’Boyle, N. M. (2012). Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *Journal of Cheminformatics*, 4(22).
- Project GUTS CS in Science curriculum (2014). Ecosystems as Complex Systems. Downloaded at <http://www.teacherswithguts.org> .
- Stepenuck, K.F., and Green. L. (2015). Individual and community level impacts of volunteer environmental monitoring: a synthesis of peer-reviewed literature. *Ecology and Society*, 20(3):19.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014, June). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- 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. Retrieved from <http://link.springer.com/article/10.1007/s10639-015-9412-6>
- 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.
- Willis, M. B., Hay, S., Martin, F. G., Scribner-MacLean, M., & Rudnicki, I. (2015). Probability with Collaborative Data Visualization Software. *Mathematics Teacher*, 109(3), 194–199.
- Wing, J. (2006, March). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

Virtuality Literacy: On the Representation of Perception

Andreas DENGEL
University of Passau, Germany
Andreas.Dengel@uni-passau.de

ABSTRACT

Immersive media such as virtual reality (VR) and augmented reality systems provide new ways of experiencing digital environments. Connecting the sense of presence to ones and zeros leads to questions on how we perceive digitally created content for it to become our subjective reality. Computational Thinking (CT) merges human abilities with computer affordances, already covering aspects ranging from data representation to the critical handling of and reflective attitude towards different forms of information and media. Combining the existing CT skills with the information and media literacy approach in terms of VR leads to the requirement of *Virtuality Literacy* as the critical reflection and production of the representation of human perception through immersive digital media. Virtuality Literacy as a new CT skill covers the thematic fields of the *representation of sensory stimuli, immersion and presence* as well as *virtual information and media literacy*. Enhancing Virtuality Literacy at an early age may lead to a better understanding of why and how immersive media can influence peoples' perceptions of various aspects of reality. Future studies will have to investigate the implementation of Virtuality Literacy in different learning environments.

KEYWORDS

Virtual Reality, Computational Thinking, Representation of Information, Information and Media Literacy

1. INTRODUCTION

Virtual Realities (VRs) as completely synthetic and immersive digital environments (Milgram, Takemura, Utsumi, & Kishino, 1994) are currently in the public eye following the latest technological developments. In contemporary Computer Science Education (CSE), the process of virtualizing information from the real world is characterized by the concept of data representation (Atchison et al., 1968; Brinda, Puhlmann, & Schulte, 2009) but efforts to combine these aspects with concepts of perceptual psychology are still lacking. As life becomes more digitized, it has become particularly important to acquire a better understanding of how different stimuli, transmitted by human sensors (visual, auditory, tactile, etc.) affect our perception of reality. This article focuses on the concept of Virtuality Literacy as the ability to critically reflect and produce human perception through immersive digital media.

2. CONNECTED CONCEPTS OF COMPUTATIONAL THINKING

2.1. Representation of Information

The process of encoding information into data structures has been recognized as an important part of CSE. Hubwieser and Broy (1999, p. 166) describe the process of representation of information: "In order to make information accessible to any kind of processing it has to be transformed into a physical

representation according to the rules of a more or less formal language". Relating this to CT, understanding the concept of computational abstraction using various forms of data representations has been identified as a fundamental CT skill (Barr, Harrison, & Conery, 2011; Wing, 2006). Together with abstraction, efficiency and heuristics, information representation has emerged as a perspective in ordinary human activities on a daily basis (Lu & Fletcher, 2009). As the concept of the representation of information underlies every form of digital data processing, it incorporates all kinds of immersive electronic media, including Virtual and Mixed Realities.

2.2. Information and Media Literacy

The requirement of knowing how to 'read' media in terms of a critical understanding as well as knowing how to 'write' in order to be able to produce them leads to a form of media literacy. Combining the different concepts of (digital) media literacy with the requirement of 'reading' and 'writing' information in a critical way the concept of information and media literacy becomes a fundamental 21st century skill for everyday and working life (Hobbs, 2010). As an unthinking use of immersive media would be critical due to the many possibilities of influencing users through simulating virtual and mixed realities (Fox, Bailenson, & Binney, 2009), information and media literacy must be the basic framework of every work with immersive virtual environments (VEs). Hence a Virtuality Literacy results when combining CT skills with information and media literacy in terms of virtual and mixed realities.

3. VIRTUALITY LITERACY

The term literacy includes reading and writing skills, whereas Virtuality Literacy (as a CT skill) addresses the abilities and competencies of analyzing, reflecting and producing information in immersive VEs. Wing describes CT as a thought process that formulates problems and their solutions by means of abstraction and decomposition in such a manner that a computer can effectively process the given problem (Wing, 2006). Virtuality Literacy focuses on the transfer process of information from the real or fictional world into a virtuality and vice versa. To split Virtuality Literacy into teachable segments, we distinguish the *Representation of Sensory Stimuli, Immersion and Presence* as well as *Virtual Information and Media Literacy* as partial competences of the transdisciplinary CT concept of Virtuality Literacy.

3.1. Representation of Sensory Stimuli

Representation of Information as a part of CSE maps the transformation of information to ones and zeroes. This classical element of the CSE curriculum is an important part of the creation of VEs as some are meant to represent a credible version of the real world. What this CT skill does not cover is the perception behind an abstraction of real world concepts. The model neglects completing the process

of the transmission of information to the recipient's brain through perception. This is essential to understanding immersive media since our perception of reality is the product of our brain's preselection and rearrangement of sensory stimuli. Figure 1 shows the *Representation of Perception* model as an extension of the *Information-Oriented Concept* from Breier and Hubwieser (2002).

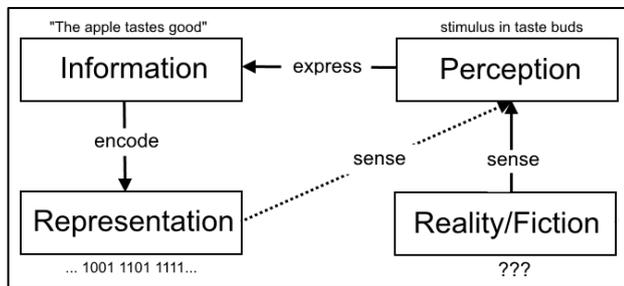


Figure 1. The Process of the Representation of Perception

The representation does not become information before being perceived by the user. Instead, it is transmitted directly to the human senses. In this model, real and fictional world are seen as a black box, as we only perceive reality through our senses. Only if the first representation (the encoded expressed perception of the experienced real or fictional world issue) equals the second representation (the encoded expressed perception of the experienced representation), this form of representation is valid. The ability to understand and apply this kind of abstraction is the main CT skill in Virtuality Literacy.

3.2. Immersion and Presence

Presence “refers not to one’s surroundings as they exist in the physical world, but to the perception of those surroundings as mediated by both automatic and controlled mental processes” (Steuer, 1992, p. 76). The different types of presence are physical, social and self-presence (Biocca, 1997). With an understanding of the *Representation of Perception*, it is possible to examine how these types of presence as the feeling of *being there* arise. While representations of physical objects have a long history in CS, representing social feelings and self-identification in a VE through ones and zeros are a CT skill of abstraction that has not yet been explored. Immersion as “a quantifiable description of a technology” (Slater, Linakis, Usoh, & Kooper, 1999, p. 3) is what turns the ones and zeros into perceived reality. The linking of the subjective feeling of presence and the technological immersion of human sensors (addressing the visual and auditory senses) and actuators (collecting data from gyro sensors for head tracking or different types of positional tracking) comprises the process of retrieving and sending data and human-computer-interaction as central CT skills.

3.3. Virtual Information and Media Literacy

As the representation of social feelings and self-identification in terms of social and self-presence is possible in immersive media, a critical reflection on these perceptions is needed. Even though *Virtual Information and Media Literacy* would be a media educational or media semiotic skill rather than a CT skill, it requires a CSE foundation. Virtual Information and Media Literacy covers aspects of

‘reading’ virtual information critically with the background knowledge of its possible influence on social feeling and self-identification. Thus, in order to obtain an overall understanding of information and media in immersive VEs using the *Representation of Perception* approach, one has to combine technological insights from a CSE perspective, apply a media educational and media semiotic angle and also view the subject through the lens of cultural and historical views and pictorial science research. The same goes for the ‘writing’ skills that allow the production of one’s own immersive information and media content.

4. REFERENCES

- Atchison, W. F., Schweppe, E. J., Viavant, W., Young, D. M., Conte, S. D., Hamblen, J. W., . . . Rheinboldt, W. C. (1968). Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science. *Communications of the ACM*, 11(3), 151–197.
- Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*. (38), 20–23.
- Biocca, F. (1997). The Cyborg's Dilemma: Progressive Embodiment in Virtual Environments [1]. *Journal of Computer-Mediated Communication*, 3(2), 0.
- Breier, N., & Hubwieser, P. (2002). An information-oriented approach to informatical education, 1, 31–42.
- Brinda, T., Puhlmann, H., & Schulte, C. (2009). Bridging ICT and CS. In P. Brézillon (Ed.), *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education* (p. 288). New York, NY: ACM.
- Fox, J., Bailenson, J., & Binney, J. (2009). Virtual Experiences, Physical Behaviors: The Effect of Presence on Imitation of an Eating Avatar. *Presence: Teleoperators and Virtual Environments*, 18(4), 294–303.
- Hobbs, R. (2010). *Digital and Media Literacy: A Plan of Action. A White Paper on the Digital and Media Literacy Recommendations of the Knight Commission on the Information Needs of Communities in a Democracy*: ERIC.
- Hubwieser, P., & Broy, M. (1999). Educating Surfers or Craftsmen: Introducing an ICT Curriculum for the 21st century. In *IFIP WG 3.1 and 3.5 Open Conference “Communications and Networking in Education: Learning in a Networked Society* (pp. 163–177).
- Milgram, P., Takemura, H., Utsumi, A., & Kishino, F. (1994). Augmented Reality: A class of displays on the reality-virtuality continuum. *SPIE Vol. 2351, Telemicroscopy and Telepresence Technologies*, 282–292. Retrieved from
- Steuer, J. (1992). Defining Virtual Reality: Dimensions Determining Telepresence. *Journal of Communication*, 42(4), 73–93.
- Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33–35.

{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/cte2018

Email

cte2018@eduhk.hk



Created and Funded by



香港賽馬會慈善信託基金
The Hong Kong Jockey Club Charities Trust
齊心同力 勇攀高峰 RACING HIGH TOGETHER

Co-created by



香港教育大學
The Education University
of Hong Kong



Massachusetts
Institute of
Technology



香港城市大學
City University of Hong Kong