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

CTE 2019

CoolThink@JC International Conference on Computational Thinking Education 2019

13-15 June 2019

CTE 2019
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
同心同步同進 RIDING HIGH TOGETHER

Co-created by

香港教育大學
The Education University of Hong Kong

Massachusetts Institute of Technology

香港城市大學
City University of Hong Kong

CoolThink@JC

Proceedings of International Conference on Computational Thinking Education 2019

13-15 June 2019

Hong Kong

Funded and Created by

The Hong Kong Jockey Club Charities Trust

Co-cerated by

The Education University of Hong Kong

Massachusetts Institute of Technology

City University of Hong Kong

Publication of The Education University of Hong Kong

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

ISBN 978-988-77034-6-4

ISSN 2664-035X (CD-ROM)

ISSN 2664-5661 (online)

CTE 2019

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

Heinz Ulrich HOPPE
University of Duisburg-Essen, Germany

Ting-chia HSU
National Taiwan Normal University, Taiwan

Rong-huai HUANG
Beijing Normal University, China

Bor-chen KUO
National Taichung University of Education, Taiwan

Kwok-yiu Robert 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

Ki-sang SONG
Korea National University of Education, South Korea

Jan VAHRENHOLD
University of Münster, Germany

Preface

International Conference on Computational Thinking Education 2019 (CTE2019) is the third 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 into problem-solving, creation and innovation. This 4-year initiative benefits over 18,500 upper primary students at 32 pilot schools on computational thinking through coding education. Through intensive professional teacher development, the Initiative develops teaching capacity of over 100 local teachers and help them master computational thinking pedagogy. Over time, the project team targets to make greater impact by sharing insights and curricular materials beyond the pilot schools.

CTE2019 is held at The Education University of Hong Kong on 13-15 June, 2019. Last year, the conference was held together with a Coding Fair to reach out to over 4500 parents and children. Riding on the success, the conference this year is organized along with the fair again to welcome enthusiastic family. Through a series of coding workshops and booth exhibition by pilot schools, participants will get a taste of computational thinking education. The parent seminars, with the theme “Code, Music and Sports”, include sharing from influencers who excel to incorporate coding in their expertise. We are excited to welcome participants to join us at the conference and the fair.

"Computational Thinking Education" is the main theme of CTE2019 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. CTE2019 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 16 sub-themes under CTE2019, namely:

Computational Thinking

Computational Thinking and Coding Education in K-12

Computational Thinking and Unplugged Activities in K-12

Computational Thinking and Subject Learning and Teaching in K-12

Computational Thinking and Teacher Development

Computational Thinking and IoT

Computational Thinking and STEM/STEAM Education

Computational Thinking and Data Science

Computational Thinking and Artificial Intelligence Education

Computational Thinking Development in Higher Education

Computational Thinking and Special Education Needs

Computational Thinking and Evaluation

Computational Thinking and Non-formal Learning

Computational Thinking and Psychological Studies

Computational Thinking in Educational Policy

General Submission to Computational Thinking Education

The conference received a total of 64 submissions (45 full papers, 12 short papers and 7 poster papers) by 137 authors from 17 countries/regions (see Table 1).

Table 1: Distribution of Paper Submissions for CTE2019

Country/Region	No. of Authors	Country/Region	No. of Authors
Taiwan	37	Israel	2
China	36	Malaysia	2
The United States	17	Sweden	2
Finland	8	Australia	1
Germany	6	Canada	1
Japan	5	Hong Kong	1
Singapore	5	Indonesia	1
South Korea	5	The Netherlands	4
The United Kingdom	4	**Total**	**137**

The International Programme Committee (IPC) is formed by 88 Members and 14 Co-chairs worldwide. Each paper with author identification anonymous was reviewed by at least three IPC Members. Related sub-theme Chairs then conducted meta-reviews and made recommendation on the acceptance of papers based on IPC Members' reviews. With the comprehensive review process, 49 accepted papers are presented (20 full papers, 19 short papers and 10 poster papers) (see Table 2) at the conference.

Table 2: Paper Presented at CTE2019

Sub-themes	Full Paper	Short Paper	Poster Paper	Total
CT	1	2	0	3
CT and Coding Education in K-12	3	2	3	8
CT and Unplugged Activities in K-12	1	2	1	4
CT and Subject Learning and Teaching in K-12	2	0	0	2
CT and Teacher Development	3	2	1	6
CT and STEM/STEAM Education	4	1	0	5
CT and Data Science	0	2	0	2
CT and Artificial Intelligence Education	0	1	1	2
CT Development in Higher Education	1	0	1	2
CT and Special Education Needs	0	0	1	1
CT and Evaluation	0	3	0	3
CT and Non-formal Learning	1	1	1	3
CT and Psychological Studies	1	0	0	1
CT in Educational Policy	1	0	0	1
General Submission to CT Education	2	3	1	6
Total	**20**	**19**	**10**	**49**

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

Keynote and Invited Speeches

There are four Keynote Speeches and one Invited Speech at CTE2019:

Keynote Speeches

1. “A Rigorous, Inclusive, and Sustainable Approach to CTforALL”

   by Dr. Leigh Ann DELYSER (CSforALL, The United States)
2. “Designing for Disciplinary-specific CT: How to Bring CT into Mathematics Classrooms?”

   by Prof. Chee-kit LOOI (Nanyang Technological University, Singapore)
3. “Evaluation and Assessment of Computational Thinking and ‘Unplugged’ Activities”

   by Prof. Jan VAHRENHOLD (University of Münster, Germany)
4. “Computational Thinking is Winning: What it is About?”

   by Prof. Valentina DAGIENĖ (Vilnius University, Lithuania)

Invited Speech

“Computational Thinking in the Interdisciplinary Robotic Game: the CHARM of STEAM”

by Prof. Ju-ling SHIH (National University of Tainan, Taiwan)

International Forum on Research, Practices and Policies on Computational Thinking Education in K-12

In this forum, there are presentations by speakers from different countries/regions on their sharing of research, practices and policies for promoting computational thinking education in their own countries/regions. Discussions focus on the directions related to the curriculum, teacher development plan, parent education campaign, and nation-/region-wide social consensus for CTE.

Panelists:

Dr. Leigh Ann DELYSER (CSforALL, The United States)

Prof. Rong-huai HUANG (Beijing Normal University, China)

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

Prof. Marcelo MILRAD (Linnaeus University, Sweden)

Prof. Ju-ling SHIH (National University of Tainan, Taiwan)

Moderator:

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

<u>Workshop "Exploring MIT App Inventor: Past, Present, and Future"</u>

This workshop introduces the history, architecture, and pedagogy of MIT App Inventor over the ten years since its inception. The new features and features under development are also demonstrated and discussed.

Speaker:

Dr. Evan PATTON (Massachusetts Institute of Technology, The United States)

<u>Workshop on Artificial Intelligence: How to Make It Easier for Students?</u>

Artificial Intelligence (AI) is a smash hit topic around the world. Gravity Link International Limited (Hong Kong) conducts a workshop on AI, in which participants are introduced with ways to bring AI education to schools.

Speaker:

Mr. Denny XIA, (Gravity Link International Limited (Hong Kong))

Doctoral Consortium

An occasion where outstanding doctoral students can present and discuss their research projects and ideas with other scholars, and thereby facilitating fruitful exchange and communication.

Moderators:

Prof. MÄKITALO, Kati (University of Oulu, Finland)

Prof. SHIH, Ju-ling (National University of Tainan, Taiwan)

Academic Paper and Poster Presentations

There are 14 sessions of academic paper presentation and an academic poster presentation session with 49 papers (20 full papers, 19 short papers and 10 poster papers) in the conference. Worldwide scholars present and exchange the latest research ideas and findings, which highlight the importance and pathways of computational thinking education covering K-12 education, artificial intelligence education, teacher development and STEM/STEAM education etc.

On behalf of the Conference Organizing Committee and CoolThink@JC, 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 CTE2019.

We sincerely hope everyone enjoy and get inspired from CTE2019.

Prof. Siu-cheung KONG

The Education University of Hong Kong, Hong Kong

Conference Chair of CTE2019 cum Coding Fair

Principal Tsz-wing CHU

St. Hilary's Primary School, Hong Kong

Conference Chair of CTE2019 cum Coding Fair

Table of Contents

COMPUTATIONAL THINKING

Full Paper

The System-analytic Approach for Gifted High School Students to Develop Computational Thinking
Nguyen-thinh LE, Niels PINKWART 2

Short Paper

Correlations among Figure Reasoning Intelligence, Computational Thinking, and Computer Programming Self-Efficacy in Scratch Program Problem Solving（圖形智能、運算思維、Scratch 程式問題解決及程式設計自我效能之關係初探）
You-Bang WU, Dai-Rung LI, Meng-Jung TSAI 8

The Study on the Factors Affecting Robotics Course Learning Intention based on Computational Thinking（基于计算思维培养的机器人课程学习意向的影响因素研究）
Jingwen SHAN........ 12

COMPUTATIONAL THINKING AND CODING EDUCATION IN K-12

Full Paper

Micro-Persistence in the Acquisition of Computational Thinking
Rotem ISRAEL-FISHELSON, Arnon HERSHKOVITZ 18

Constructing Expert Programming Thinking Process in the Field of Information Engineering, Promoting the Planning of Operational Thinking Teaching Activities（建構資訊工程領域專家程式設計思考程序及促進運算思維教學活動規劃）
Hsien-Sheng HSIAO, Yu-An LIN 24

The Effects of Gender Differences and Learning Styles on Scratch's Programming Performance and Computational Thinking Ability（性別與學習風格對程式設計學習成效與運算思維能力之影響）
Yun-jie JHOU, Jung-chuan YEN, Wei-chi LIAO........ 30

Short Paper

Supporting Representational Flexibility in Computational Thinking: Transitions between Reactive Rule-based and Block-based Programming
H. Ulrich HOPPE, Sven MANSKE, Sören WERNEBURG........ 37

A Study on the Current Situation of Visual Programming for Primary School Students and Its Influencing Factors（小学生可视化编程学习现状及其影响因素研究）
Min ZHANG, Yi ZHANG, Huan-huan LIU, Wei MO, Dan-dan WANG........ 41

Poster

The Design and Development of Coding Poker Cards
Sheng-yi WU........ 46

Promoting Computational Thinking Skills in the Context of Programming Club for K-12 Pupils with the Engaging Game Adventure in Minecraft
Jussi Koivisto, Jari Laru, Kati Mäkitalo........ 48

Investigating the Elementary School Students' Skills of Computational Thinking and Self-Efficacy through a Robot Programming Project（小學生專題式程式設計對運算思維和自我效能的影響）
Chien-Yuan SU, Song HAN, Yue HU 50

COMPUTATIONAL THINKING AND UNPLUGGED ACTIVITIES IN K-12

Full Paper

Effects of Plugged and Unplugged Advanced Strategy on Primary School Children's Outcomes in Scratch Learning（插電與不插電程式教學前導策略對國小程式設計學習成效之影響）
Wei-chi LIAO, Jung-chuan YEN 54

Short Paper

Exploring Evidence that Board Games can Support Computational Thinking
Ching-yu TSENG, Jenifer DOLL, Keisha VARMA.. 61

A Preliminary Study on Designing Learning Activity of Mathematics Path via Computational Thinking for the Elementary School Students with Learning Disability（結合運算思維在國小學習障礙學生的數學步道教學活動設計初探）
Ya-chi CHANG, Sung-chiang LIN 65

Poster

Designing Unplugged Activities for Learning Computational Thinking in the Context K-2 Pupils' Afterschool Coding Club
Eunice Eno Yaa Frimponmaa AGYEI, Jari LARU, Kati MÄKITALO 70

COMPUTATIONAL THINKING AND SUBJECT LEARNING AND TEACHING IN K-12

Full Paper

Research on Gamified Collaborative Learning in the Cultivation of Computational Thinking （游戏化协作学习在运算思维培养中的应用研究）
Yuyu LIN, Jiansheng LI... 73

Teaching Research on Cultivating Pupils' Computational Thinking in Scratch Course（在 Scratch 课程培养小学生计算思维的教学研究）
Zhi-lin LI, Hong YU, Yu-xiao XU... 73

COMPUTATIONAL THINKING AND TEACHER DEVELOPMENT

Full Paper

Employing Computational Thinking in General Teacher Education
Stefan SEEGERER, Ralf ROMEIKE 86

The Complexity of Teacher Knowledge, Skills and Beliefs about Software Education: Narratives of Korean Teachers
Da-hyeon RYOO, Hyo-jeong SO, Dongsim KIM 92

A Model for Readiness Analysis of Schools Conducting Computational Education（運算思維之學校準備度模型分析）
Yu-lan HUANG, Ting-chia HSU... 98

Short Paper

Computational Thinking in Finnish Pre-Service Teacher Education
Kati H. MÄKITALO, Matti TEDRE, Jari LARU, Teemu VALTONEN 105

Computational Thinking Education for In-Service Elementary Swedish Teachers: Their Perceptions and Implications for Competence Development
Dan KOHEN-VACS, Marcelo MILRAD 109

Poster

Technology Acceptance and Teacher Attitude for Computational Thinking in the Netherlands
Marcus SPECHT, Robert Jan JOOSSE 113

COMPUTATIONAL THINKING AND STEM/STEAM EDUCATION

Full Paper

An Empirical Study on STEM Learning Satisfaction and Tendency for Creativity of Chinese Secondary School Students（中国中学生 STEM 学习满意度与创新力倾向的实证研究）
Wangwei LI, Chun CHEN 116

Using a 6E Model Approach to Improve Students Learning Motivation and Performance about Computational Thinking（6E 學習模式結合機器人教育對學習動機與運算思維學習成效之影響）
Hsien-sheng Hsiao, Jyun-chen Chen, Yi-wei Lin, Hung-wei Tsai 122

A Robotic Course Designed with CT 3D Model （基于运算思维三维模型重新设计机器人课程）
Fengshen HE, Xiaoqing GU, Yuhe YI, Yong OU 128

Computational Thinking in STEM Task Design: Authentic, Useful, Experiential, and Visual（跨領域運算思維學習任務設計：真實、有用、體驗、視覺化）
Hao-min TIEN, Jung-chuan YEN 135

Short Paper

Research on STEM Curriculum Design for Computational Thinking: Framework Design and Case Analysis（面向计算思维的 STEM 课程设计研究：框架设计与案例分析）
Hui SHI, Feng LI.. 141

COMPUTATIONAL THINKING AND DATA SCIENCE

Short Paper

Block Affordances for GraphQL in MIT App Inventor
Lujing CEN, Evan W. PATTON.. 147

An Integration of Computational Thinking into Teaching Activity Design for Learning Data Analysis and its Application（探究融入運算思維於學習資料分析的教學設計與應用）
Yuan-yi HUANG, Sung-chiang LIN.. 151

COMPUTATIONAL THINKING AND ARTIFICIAL INTELLIGENCE EDUCATION

Short Paper

Classroom Activities for Teaching Artificial Intelligence to Primary School Students
Joshua W. K. HO, Matthew SCADDING 157

Poster

The Popstar, the Poet, and the Grinch: Relating Artificial Intelligence to the Computational Thinking Framework with Block-based Coding
Jessica Van BRUMMELEN, Judy Hanwen SHEN, Evan W. PATTON 160

COMPUTATIONAL THINKING DEVELOPMENT IN HIGHER EDUCATION

Full Paper

A Preliminary Study of Project-based Learning Teaching Activity for Programming based on Computational Thinking（基於運算思維之 PBL 程式設計教學活動成效初探）
Zi-yun LU, Sung-chiang LIN 163

Poster

Flipped Learning Approach for Coding Education in Higher Education
Hui-chun HUNG 169

COMPUTATIONAL THINKING AND SPECIAL EDUCATION NEEDS

Poster

Integrating Computational Thinking and Mathematics for Children with Learning Disabilities with Google Blockly（學習障礙學生運算思維與數學的 Google Blockly 教學設計）
Chen-huei LIAO, Bor-chen KUO, Kai-chih PAI, Pei-chen WU, Chih-wei YANG 172

COMPUTATIONAL THINKING AND EVALUATION

Short Paper

The Measurement of Computational Thinking Performance Using Multiple-choice Questions
Yerkhan MINDETBAY, Christian BOKHOVE, John WOOLLARD 176

Research on the Construction of App Inventor Program Evaluation Indicators based on Computational Thinking（基于计算思维的 App Inventor 程序评价指标构建研究）
Yue LIANG, Jinbao ZHANG 180

Development of a Computational Thinking Scale for Programming（程式設計之運算思維量表）
Yuan-kai CHU, Jyh-chong LIANG, Meng-jung TSAI 185

COMPUTATIONAL THINKING AND NON-FORMAL LEARNING

Full Paper

The Learning Effectiveness of Integrating Computational Thinking and English Oral Interaction（整合運算思維與英語互動的成效分析）
Yi-Sian LIANG, Ting-chia HSU 191

Short Paper

Establishing Equitable Computing Programs in Informal Spaces: Program Design, Implementation and Outcomes
Hui YANG, Chrystalla MOUZA, Lori POLLOCK 197

Poster

Implementing Computational Thinking through Non-formal Learning in after School Activities at Students Society Club
Poh-tin LEE, Xin-rui LEE, Chee-wah LOW, Athinamilagi KOKILA 201

COMPUTATIONAL THINKING AND PSYCHOLOGICAL STUDIES

Full Paper

What Underlies Computational Thinking: Exploring Its Cognitive Mechanism and Educational Implications
Rina Pak-Ying LAI 204

COMPUTATIONAL THINKING IN EDUCATIONAL POLICY

Full Paper

Implementing Computational Thinking in the Dutch Curriculum an Exploratory Group Concept Mapping Study
Marcus SPECHT, Marinka COENDERS, Slavi STOYANOV 210

GENERAL SUBMISSION TO COMPUTATIONAL THINKING EDUCATION

Full Paper

Exploring the Role of Algorithm in Elementary School Students' Computational Thinking Skills from a Robotic Game
Hsin-yin HUANG, Shu-hsien HUANG, Ju-ling SHIH, Meng-jung TSAI, Jyh-chong LIANG 217

Developing Computational Thinking Practices through Digital Fabrication Activities
Megumi IWATA, Kati PITKÄNEN, Jari LARU, Kati MÄKITALO... 223

Short Paper

A Goal Analysis of Computer Science Education: Setting Institutional Goals for CS Ed
Stephanie B. WORTEL-LONDON, Leigh Ann DELYSER, Lauren WRIGHT, Júlia Helena AGUIAR 229

Research on the Current Situation and Development Trend of Computational Thinking in K-12 Education in China ——Keywords Co-Word Analysis Based on Knowledge Map（我国 K-12 计算思维的现状审视与发展趋势研究——基于知识图谱的关键词共词分析）
Jue WANG, Yi ZHANG, Xing LI, Qiang REN, Lin MEI 233

Learning Effectiveness of Using Augmented Reality to Support Computational Thinking Learning Board Game（擴增實境運算思維教育桌遊之學習成效與認知負荷之分析）
Wei-chen KUO, Ting-chia HSU 238

Poster

Exploring Convergence and Divergence in Infinite Series
David ZEIGLER 243

Computational Thinking

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The System-analytic Approach for Gifted High School Students to Develop Computational Thinking

Nguyen-thinh LE[1*], Niels PINKWART[2]
[12] Humboldt Universität zu Berlin, Germany
nguyen-thinh.le@hu-berlin.de, niels.pinkwart@hu-berlin.de

ABSTRACT

In this paper, we report lessons learned from applying the system-analytic approach in developing computational thinking for high school students. We have been establishing a network of Society for Gifted School Students in Computer Science since two years. Every year, we offer a ten-weeks project for gifted students from schools around the city Berlin in Germany. In one study case in summer semester 2016, after ten weeks, students finished successfully their own projects ideas with a small software product. For evaluating the system-analytic approach, we used three measure instruments: 1) the subjective attitude of teacher students who supervised the school students, 2) products of the projects, and 3) the repeated participation of the school students. We could report the following results: Three teacher students showed positive experience with the system-analytic approach; Each student group could realize their own ideas and successfully developed apps; 70% of school students, who attended the project in summer semester 2016 applied for participation in the second project.

KEYWORDS

gifted students, system-analytic approach, computer science education

1. INTRODUCTION

We were faced by a question from the parents of a gifted high school student: “My son is able to write programs in five programming languages. Do you have a method to boost him?” To answer this question, first, we looked into the curricula of different federal states in Germany, and then international curricula (e.g., the CSTA K-12 Computer Science standards of ACM, 2011). Unfortunately, we could not really find a didactical principle or contents for gifted school students in Computer Science. A possible solution might be recommending such students to attend local courses held in communities around the globe such as CoderDojo (https://coderdojo.com), Hour of Code (https://hourofcode.com) or attending self-paced courses from online coding schools such as Code.org (http://code.org), CodaKid (https://codakid.com), Khan Academy (https://khanacademy.org). With those self-paced online courses and communities, they might develop their competency by themselves in programming. However, those courses rather support students in developing programming skills than computational thinking, which is a fundamental competence to be acquired.

Didactical approaches for developing computational thinking have mostly been developed and validated with average intellectual level students. For gifted school students, specific didactical approaches for developing computational thinking are rare. In a textbook, Schubert and Schwill (2011) proposed the system-analytic approach for novice Computer Science students, who have just begun learning Computer Science. They found a disadvantage of this approach that it would require high intellectual level of students. Exploiting this “disadvantage”, we hypothesize that this approach might be appropriate for gifted students, because they have higher ability level than others and have more curiosity.

In this paper, we investigate the research question: Can the system-analytic approach be adopted to gifted students? We briefly review approaches to teaching gifted students in Section 2 and didactical approaches for teaching computational thinking in Section 3. The implementation of the system-analytic approach for a group of gifted students is described in Section 4. We report on the success of our first implementation of the system-analytic approach to developing computational thinking for gifted students in Section 5.

2. APPROACHES TO TEACHING GIFTED STUDENTS

There are many diverse definitions for “giftedness”. While the definition for “giftedness” in most English literature relies on the Section 9101 of US Elementary and Secondary Education Act, “Students, children, or youth who give evidence of **high achievement capability in areas such as intellectual, creative, artistic, or leadership capacity, or in specific academic fields**, and who need services and activities not ordinarily provided by the school in order to fully develop those capabilities.” (US, 2019), the definition for “Giftedness” in German literature is rather based on a specific IQ (intelligence quotient) level. The Federal Ministry for Education and Research of Germany considers gifted students as the ones, who have IQ over 130 (BMBF, 2019). Johnsen (2004) summarized three common features among definitions for gifted students: 1) Students show high performance in different areas (e.g., intellectual, creative, artistic, leadership, academic); 2) The comparison with other groups (e.g., general education classrooms, of the same age, experience, or environment); 3) A need for development of the gift (e.g., capability or potential). Adopting these three features, in our following study case, we consider students, who participate in the Society for School Students in Computer Science, as potentially gifted, because they are recommended by their school teachers and are required to pass an exercise from the Computer Science Competition “Informatik Biber” (https://bwinf.de/biber).

Azzam (2016) suggested six strategies for challenging gifted students: 1) “Offer the Most Difficult First”, 2) “Pre-Test for Volunteers”, 3) “Prepare to Take It Up”, 4) “Speak to Student Interests”, 5) “Enable Gifted Students to Work Together”, and 6) “Plan for Tiered Learning”. The first strategy is to give all students (not only gifted students) most difficult tasks. If they can solve the most difficult tasks first, then they should be freed from additional homework assignments. However, Azzam did not discuss how to deal with the difficult tasks if some students cannot solve them. The second strategy is applied to sort out gifted students from non-gifted students. For those students, who pass most test items, would be recommended to solve advanced tasks and this decision is left to all students. This strategy avoids gifted students becoming bored. However, if a teacher already knows the ability of each student, such a pre-test would be not required. The third strategy aims at providing differentiated work materials to students of different ability levels and this can be referred to as performance-differentiated strategy. This strategy is usually adopted by teachers. However, this strategy requires much preparation by teachers. The fourth strategy is intended to help teachers develop learning materials adaptive to interests of his/her students. The fifth strategy promotes collaborative learning for enhancing their academic performance and benefits gained by other students. The sixth strategy suggests teachers to plan their lessons at different tiers of difficulty. The author argued that teachers have to develop their lesson plan, anyway. Thus, at the planning time, they can also develop deep and complex activities for gifted students and prepare work sheets at the entry, advanced, and extension levels. Similar to the third strategy, the plan for tiered learning aims at avoiding gifted students getting bored.

In the explorative study with 112 potentially gifted Master students in Serbia, Gojkov et al. (2015) suggested that didactical teaching approaches for gifted higher education students should “encourage curiosity, being well-informed, open-minded, flexible, confronting personal prejudices, carefully making decisions”, and thus stimulate critical thinking of gifted students.

3. DIDACTICAL APPROACHES TO COMPUTATIONAL THINKING

Didactical approaches for programming can be found in a huge body of literature, e.g. use peer instruction (Porter et al., 2011; Porter et al., 2013; Council, 2015), use live coding instead of showing slides (Barker et al., 2005; Rubin, 2013; Willingham, 2009), use worked examples and labelled subgoals (Margulieux et al., 2012; Morrison et al., 2015), use authentic tasks (Guzdial, 2013; Bouvier et al., 2016; Repenning, 2017). Brown & Wilson (2018) summarized 10 tips for teaching programming, which are based on research results. However, didactical approaches for computational thinking and their empirical validations are rare to be found.

Atmatzidou and Demetriadis (2014) proposed to deploy robotics activities to develop computational thinking skills. The authors focused on the following skills of computational thinking: abstraction, generalization, algorithm, modularity, decomposition and problem solving. The authors reported positive results that students became familiar with the concepts of computational thinking and could deploy them in the problem solving process.

Armoni et al. (2010) proposed to present computational elements and algorithm/program design in a “zipped” manner. That is, both theoretical and practical notions are “zipped” in a proposed order. The “algorithmic first, object second” approach suggests to teach fundamental algorithmic aspects first, followed by object oriented notions. In addition, the authors suggested deploying several didactical principles: utilizing motivating examples and demonstrating gradual design processes.

Caspersen and Nowack (2013) proposed the following five didactical principles to computational thinking education in Danish high schools: (1) A learning activity is not (necessarily) the same as a knowledge area; (2) Learning activities should be application-oriented; (3) Learning activities should facilitate and guide a consume-before-produce progression through the materials; (4) Learning activities should include several substantial worked examples; (5) Learning activities should illustrate stepwise improvement as a general approach to incremental development of artefacts.

Other didactical principles have been proposed such as game-based learning and narrative media-approaches (Andersen et al., 2003), activity-based approaches (Hazzan et al., 2011)

Either deploying robotics in educational activities (Atmatzidou & Demetriadis, 2014), or using motivating examples and gradual design process (Armoni et al., 2010), or game-based (Andersen et al., 2013), or activity-based approaches (Hazzan et al., 2011), those approaches underlie the constructivism theory, which is frequently promoted in general Computer Science education (Salanci, 2015; Hadjerrouit, 2009).

To our best knowledge, since a specific didactical approach for gifted students to computational thinking has not been proposed and validated, we attempted to investigate whether the system-analytic approach is applicable for gifted students.

The system-analytic approach (Schubert & Schwill, 2011) requires students to study a complex software system in a top-down manner through the following phases: (1) Looking on the system, (2) digging into the system, (3) modifying the system, and (4) constructing a system. In the first phase “Looking on the system”, the student’s activity starts with trying to use the system. The student tries to interact with the system through the system’s user interface. Through this activity, the student is expected to acquire the competency of using the computer system and evaluating a system. In the second phase “looking into the system”, the student is asked to identify the internal components of the system (maybe with a documentation) and to investigate the interplay between the internal components and the system’s interaction with the user. After the student has acquired an understanding about the internal “world” of the system, the third phase “modifying the system” requires the student to extend the system with a new functionality or to adjust the system according to a new requirement. In the last phase “constructing a system”, the student’s experience with system development will be applied and extended. The student can reuse existing components of the initial systems and construct a new system to solve a similar problem. Thus, the system-analytic approach can be considered a variant of the activity-based approach (Hazzan et al., 2011) and an implementation of the constructivism learning theory (Salanci, 2015; Hadjerrouit, 2009).

According to Schubert and Schwill (2011), the system-analytic approach has various advantages. First, this approach is authentic to software development in the industry, because it requires a usage and construction of information systems. Second, this approach may be suited in a project-oriented learning setting, which requires team work and that is the authentic working environment in IT companies. Third, this approach is subject-crossing and thus, a project-based learning setting could involve an application context outside of the learning subject computer science and various social competencies might be enhanced. Since this approach requires competencies in different areas (in addition to Computer Science), thus, students with less knowledge in Computer Science can contribute their knowledge in other subjects in the project as well. Schuber and Schwill (2011) suggested two disadvantages for this approach. First, this approach is highly intellectual-demanding. It requires the instructor to prepare an appropriate system (or program product) to be analyzed. Second, the approach does not solve the diversity problem of heterogeneous student groups. Due to the high demand of intellectuality, the system-analytic approach might be appropriate for gifted students, who usually have higher intellectuality than others.

4. METHOD

In the following, we present a study case, in which the system-analytic approach was used to teach a group of gifted students. The study case was the first project offered to gifted school students in Berlin. The school students between the 7^{th} and 10^{th} grade were recommended by Computer Science teachers in our partner schools around Berlin (Germany) to join the "Society of Computer Science for Gifted Students". That means, the student group is heterogeneous. The time capacity for each project was limited to ten weeks, each has two academic hours (90 minutes) and the project was required to take place after the regular school time. Given these constraints, we decided to adopt the system-analytic approach, because, first, it meets the intellectual level of gifted students. Second, they are creative and high demanding to create a system quickly, thus, the approach may meet their satisfaction of developing their own ideas after passing the first three phases. In these projects with gifted school students, we planned to deploy new technology (e.g., tablets, drones, robots, ect.). First, new technology serves as means to enhance motivation of students, because Ozcan and Bicen (2016) reported that gifted students indicated an important role of technology in their education. Second, we intended to implement the constructivism learning theory (see Section 3). The projects we conducted with gifted students were intended to develop the following competencies that were based on the recommendations of the Society for Computer Science in Germany for schools. The process-oriented skills include: 1) modeling and implementation, 2) reasoning and evaluating, 3) structuring and networking, 4) communication and cooperation, 5) presentation and interpretation. The content-oriented skills include: 1) information and data, 2) algorithms, 3) programming languages and automata, 4) informatics systems, 5) informatics, humans and society (GI, 2008).

The following study case was conducted in summer semester 2016 based on the constraints and conditions above. Due to time constraint of 10 weeks, we aimed at enhancing the computational thinking skills of school students by focusing on the following content-oriented skills: algorithms and informatics systems along the process-oriented competency dimension. The topic of our project was app development, because at that time apps were penetrating our daily life and students needed to know how an app can be developed, and thus, addressing the specified computational thinking skills (algorithms and informatics systems). After analyzing different Android development platforms, we decided for the MIT App Inventor (http://ai2.appinventor.mit.edu), because the other platforms such as Android Studio (https://developer.android.com/studio) requires an introduction into a high-level programming language, whereas MIT App Inventor provides visual programming language that is easier to acquire within short time period (10 weeks).

20 high school students were admitted to join our project, among which there were three female students. The 20 high school students were divided into ten groups, which were supervised by three teacher students for the computer science education. After the second week, three high school students dropped out. Seventeen remaining students continued to the end of the project period.

With the intention of adopting the system-analytic approach, we had to prepare materials for the first three phases "looking on the system", "looking into the system", "modifying the system". For these purposes, we collected existing apps (e.g. Photo Booth, TalkToMe, Quiz Me, No Text While Driving for AI2, Exploring with Location Sensor in AI2) provided on the tutorial page (http://appinventor.mit.edu/explore/ai2/tutorials). Each app was analyzed with respect to its difficulty level (easy, medium, and difficult) and its extension possibilities were suggested. The difficulty of each app served to recommend school students to choose an appropriate app corresponding to their level. The extension possibilities of each app were intended to give students as working exercises. Given the selected apps, adopting the system-analytic approach, in Phase 1, the students should choose an app and play with it. In Phase 2, the students analyze the functionality of the chosen app. In Phase 3, the students modify/extend the functionality or the design of the app. Finally, in Phase 4, the students are asked to design the concept for a new app and to implement the app using MIT App Inventor.

The teaching concept for the 10 weeks project looks as following. The first session aims at introducing the organization of the project and the App Inventor in general.

The second session aimed at carrying out Phase 1 and 2. First, we presented an example application and its code on App Inventor. This presentation was intended to help students be familiar with App Inventor. Then, the students were asked to choose an app from the collected list with flagged difficulty levels and to analyze it. In the last 15 minutes of the session, each group was required to present its results to the class.

The third and fourth sessions' objective was modifying an app. For this purpose, first, we presented an app and showed its code. Then, we asked the students for possible modifications on this app. From those suggested modifications, we illustrated some small modifications by changing/adding code of that app. During this step, we explained how the added code would change the app. After that, we asked the students to modify the design of a chosen app (i.e., the GUI components) and to add new functionalities to the existing app by copying and pasting existing code. Fifteen minutes before the session's end, we requested the students to present results of their modification tasks. Except one student in one group, other groups completed their task. One special gifted student finished the modification tasks, left his group and worked on developing his own app. Each group was requested to present results of their modification tasks.

From now on, the students were supported to develop their own projects. We adopted the project method of Frey (2010): finding a project idea, drawing a project concept, concretizing the project plan, carrying out the project, reflecting the project plan, meta interaction (discussion about the progress of the project). Adopting this project method suggested by Frey (2010), the fifth session was devoted to helping students develop ideas for new apps. Each group was asked to develop own idea and concept for an app. Before the session was ended, the groups were requested to exchange their ideas. The presentations of the groups' ideas showed that students had difficulty. Some of the groups did not have concrete ideas whereas some others had ideas that were not realistic to be realized within the given time constraint (4 weeks left). Thus, we encouraged the students to think about ideas for their apps as homework. In addition, for special gifted students, we encouraged them to look at the Android studio platform, if they find MIT App Inventor would not satisfy the requirements of their app project.

As a support for students in developing ideas for a new app, in the beginning of the sixth session, we gave each group a structure, which includes the following questions: 1) How is our app? 2) What kind of functionalities can our app provide? 3) How should our app look like (how should the user interface be designed)? 4) How is the mile stone plan for the project to be carried out in the next 4 weeks? The concept (Questions 2 and 3) and the project plan (Question 4). During this session, we supported the groups to concretize their ideas and discussed with them about the realistic components of their apps. After their ideas have been agreed by us, they started to design the user interface for their apps.

The next three sessions were planned for the implementation of the apps' concepts. In the beginning of this construction phase, we gave the students some hints regarding project management (e.g., milestones specification and phases of a software development cycle). We were available for the groups on demand. Through this construction phase, the students applied their multifaceted interdisciplinary knowledge and their competency in using MIT App Inventor that they acquired in the first three phases to realize their own app ideas. At the end of the ninth session, the groups could realize their apps' ideas, but not completely. As homework, we encouraged them to optimize their apps and informed them about the presentation on the last session with the presence of their parents.

The last session was reserved for preparing the presentation of each project group. All the groups could demonstrate their project results. As lessons learned, the most difficult part of the 10-weeks project was the task to develop a project idea with the students. Some students did have great ideas and desired to realize them. However, those too big project ideas can not be realized within the limited project time. Instead, we encouraged them to limit the realization possibilities for their ideas. For example, some students would like to develop an additional server platform and that is unrealistic for the project period, if the students did not have experience with server-based software development. Of course, we could show them the possibility to connect web services with App Inventor and let them decide by themselves, if their project could be finished in the given time frame.

5. RESULTS

Since the aim of our paper is to investigate the research question "How can the system-analytic approach be applied appropriately to gifted students?", it is required to evaluate the results of the project that implemented the system-analytic approach. For evaluating the success of the project, we used three instruments: 1) The attitude of the student teachers, who developed learning materials and supervised the gifted school students; 2) Products of the project, i.e., developed apps; 3) The motivation of the gifted students.

Results based on the first measure were collected from the reports of the teacher students. The first student reported as follows: "*through the possibility of developing and realizing the own ideas, on the opposite to the common didactical approach in the school, gifted students could make a lot of new experience. Thus, they can benefit a lot from their different experiences independent from their teacher*", "*the system-analytic approach helped the students discover and understand apps quickly. Thus, it brought the students necessary experience to realize their own apps after solving the modification tasks successfully*". The second student reported that "*this is my first time I could test the system-analytic approach. I must say that this approach is appropriate for our conditions (i.e., 10 weeks project, gifted school students) and the top-down manner of the approach could help students to find additional understanding for a complex system*". The third teacher student found the project "*very positive that we could connect between theories and practice. I think that App inventor enhances the intrinsic motivation of the students because they could develop their apps easily and share them with other users on the same platform*".

Considering the number of developed apps as a measure for the success of the project, seven apps of seven groups were developed and demonstrated on the last session with the presence of the students' parents. They were proud to present and explain their apps. Figure 1 illustrates one of the apps developed by the students. This game requires the player to avoid the bricks representing moving meteoroids. The left part of the figure shows the media that were required to build the app and the right part is the emulator of MIT App Inventor.

Figure 1. A project entitled "meteoroids".

In order to measure the motivation of the gifted students about their project, we checked the number of students who sent us application for the second project in the winter semester 2016/2017. We could find that 70.1% of them (12 old school students) wanted to join our second project.

6. CONCLUSIONS

While some approaches to developing computational thinking in schools have been researched and developed, for the special group of gifted students, didactical approaches are rarely found in literature. In this paper, we have suggested to adopt the system-analytic approach for gifted students. We learned the following lessons. First, this approach helps students work through and be familiar with a new technology very quickly. We did not need to present new Computer Science concepts related to the technology in the bottom-up manner. Second, in order to apply this approach, a careful choose of existing applications is required in order to develop modification tasks. The applications should meet the students' interests. Thus, not only one single application, but several applications are required in order to satisfy different students. With different applications, various types of underlying concepts can be learned. Therefore, preparing appropriate applications for students is the most important task adopting the system-analytic approach. Since this approach does not address the performance heterogeneity of students, additional strategy, e.g., collaborative learning in a project setting, should be embedded.

7. REFERENCES

ACM. (2011). *CSTA K-12 CS Standards, 2011 Edition*. Retrieved April 5, 2019, from https://www.csteachers.org/page/standards

Atmatzidou, S., & Demetriadis, S. (2014). How to Support Students' Computational Thinking Skills in Educational Robotics Activities. *Proceedings of 4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, 43-50.

Andersen, P. B., Bennedsen, J., Brandorff, S., Caspersen, M.E., & Mosegaard, J. (2003). Teaching Programming to Liberal Arts Students – A Narrative Media Approach. *Proc. of the Conference on Innovation and Technology in Computer Science Education*. ACM Press.

Armoni, M., Benaya, T., Ginat, D., & Zur, E. (2010). Didactics of Introduction to Computer Science in High School. *In Proceedings of the 4th International Conference on Informatics in Secondary Schools - Evolution and Perspectives*. Springer Verlag, 36-48.

Azzam, A. (2016). Six Strategies for Challenging Gifted Learners. *ASCD Education Update, 58*(4). Retrieved April 5, 2019, from http://www.ascd.org/publications/newsletters/education-update/apr16/vol58/num04/Six-Strategies-for-Challenging-Gifted-Learners.aspx

Barker, L. J., Garvin-Doxas, K., Roberts, E. (2005). What Can Computer Science Learn from a Fine Arts Approach to Teaching? *In: Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. ACM, 421-425.

BMBF. (2019). *Bundesministerium für Forschung und Bildung*. Retrieved April 5, 2019, from https://www.bmbf.de/upload_filestore/pub/Begabte_Kinder_finden_und_foerdern.pdf

Bouvier D, Lovellette E, Matta J, Alshaigy B, Becker BA, Craig M, et al. (2016). Novice Programmers and the Problem Description Effect. *In: Proceedings of the 2016 ITiCSE Working Group Reports*. ACM, 103-118.

Brown, N. & Wilson, G. (2018). Ten Quick Tips for Teaching Programming. *PLoS Computational Biology, 14*(4), e1006023. DOI:10.1371/journal.pcbi.1006023

Caspersen, M. & Nowack, P. (2013). Computational Thinking and Practice — A Generic Approach to Computing in Danish High Schools. *Proceedings of the 15th Australasian Computing Education Conference*, 137-143.

Council, N. R. (2015). *Reaching Students: What Research Says about Effective Instruction in Undergraduate Science and Engineering*. Washington, DC: The National Academies Press.

Frey, K. (2010). *Die Projektmethode*. Der Weg Zum Bildenden Tun. *Beltz Verlag*. ISBN: 978-3-407-25688-1

GI. (2008). *Bildungsstandards Informatik für die Sekundarstufe I*. Retrieved April 5, 2019, from https://gi.de/fileadmin/GI/Hauptseite/Aktuelles/Meldungen/2016/Bildungsstandards_2008.pdf

Gojkov, G., Stojanović, A., Gojkov-Rajić, A. (2015). Didactic Strategies and Competencies of Gifted Students in the Digital Era. *CEPS Journal, 5*(2), 57-72.

Guzdial M. (2013). Exploring Hypotheses about Media Computation. *In: Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ACM, 19-26.

Hadjerrouit, S. (2009). Didactics of ICT in Secondary Education: Conceptual Issues and Practical Perspectives. *Issues in Informing Science and Information Technology, Vol. 6*. Retrieved April 5, 2019, from https://pdfs.semanticscholar.org/800b/bc1b5fb9bf60a3c3faa01045c2aa043433bd.pdf

Hazzan, O., Lapidot, T., & Ragonis, N. (2011). *Guide to Teaching Computer Science: An Activity-based Approach*. Springer-Verlag. ISBN 978-0-85729-443-2

Johnsen, S. K. (2004). Definitions, Models, and Characteristics of Gifted Students. *In Johnsen, S. K. (ed.): Identifying Gifted Students: A Practical Guide*. Prufrock Press, 1-22.

Margulieux, L. E., Guzdial, M., & Catrambone, R. (2012). Subgoal-labeled Instructional Material Improves Performance and Transfer in Learning to Develop Mobile Applications. *In: Proceedings of the 9th Annual International Conference on International Computing Education Research*, 71-78.

Morrison, B. B., Margulieux, L. E., & Guzdial, M. (2015). Subgoals, Context, and Worked Examples in Learning Computing Problem Solving. *In: Proceedings of the 11th Annual International Conference on International Computing Education Research*. ACM, 21–29.

Ozcana, D., & Bicen, H. (2016). Giftedness and Technology. *Procedia Computer Science, 102*, 630-634.

Porter, L., Bailey Lee, C., Simon, B., Cutts, Q., & Zingaro, D. (2011). Experience Report: A Multi-classroom Report on the Value of Peer Instruction. *In: Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 138-142.

Porter, L., Guzdial, M., McDowell, C., & Simon, B. (2013). Success in Introductory Programming: What Works? *Communications of the ACM, 56*(8), 34-36.

Ratuporo, J., Poentinen, S., & Kukkonen, J. (2006). Towards the Information Society – The Case of Finnish Teacher Education. *Informatics in Education, 5*(2), 285-300.

Repenning A. (2017). Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets. *Journal of Visual Languages and Sentient Systems, 3*, 68-91 DOI: 10.18293/VLSS2017-010.

Rubin, M. J. (2013) The Effectiveness of Live-coding to Teach Introductory Programming. *In: Proceeding of the 44th ACM Technical Symposium on Computer Science Education. SIGCSE'13*. ACM, 651-656.

Salanci, L. (2015). Didactics of Programming. *ICTE Journal, 4*(3), 32-39.

Schubert, S. & Schwill, A. (2011). *Didaktik der Informatik. Springer Spektrum*, ISBN 978-3-8274-2653-6, 287-302.

US. (2019). *Elementary & Seconary Education*. Retrieved April 5, 2019, from https://www2.ed.gov/policy/elsec/leg/esea02/pg107.html

Willingham, D. T. (2009). *Why Don't Students Like School? A Cognitive Scientist Answers Questions about How the Mind Works and What It Means for the Classroom*. John Wiley & Sons.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Correlations among Figure Reasoning Intelligence, Computational Thinking, and Computer Programming Self-Efficacy in Scratch Program Problem Solving

You-bang WU[1], Dai-rung LI[2], Meng-jung TSAI[3*]
[12] National Taiwan University of Science and Technology, Taiwan
[3] National Taiwan Normal University, Taiwan
elvis0226@gmail.com, mos398@gmail.com, mjtsai99@ntnu.edu.tw

ABSTRACT
Computational thinking plays a critical role in learning computer programming. However, the relationship between the development of computational thinking skills and learner's intelligence is still not clear in past studies. This study investigated the correlations among learner's figure reasoning intelligence, computational thinking, Scratch program problem solving and computer programming self-efficacy. A total of 44 university students from north Taiwan participated in this study in which 6 Scratch loop programs were used for problem solving. A Pearson correlation analysis was conducted and the coefficients among the Figure Reasoning Intelligent test scores, the Bebras test scores, the Scratch program problem solving performance and the computer programming self-efficacy scores were positively significant. This study suggested future studies to further explore the roles of figure reasoning skills and computational thinking in learning computer programming and possible applications for individualized learning and instruction.

KEYWORDS
computational thinking, graphical thinking, scratch computer programming, problem solving, self-efficacy

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

圖形智能、運算思維、Scratch 程式問題解決及程式設計自我效能之關係初探

吳侑邦[1]，李岱榕[2]，蔡孟蓉[3*]
[12]國立臺灣科技大學，臺灣
[3]國立臺灣師範大學，臺灣
elvis0226@gmail.com，mos398@gmail.com，mjtsai99@ntnu.edu.tw

摘要

運算思維在程式設計學習中扮演重要角色，然而，運算思維能力發展與學習者本身的智能發展之間的關係在研究文獻上仍不十分清楚。本研究探討學習者的圖形智能和運算思維、Scratch 程式問題解決和程式設計自我效能之間的關係。研究對象為 44 位台灣北部的大專生，實驗素材為六題 Scratch 迴圈問題解決。初步研究結果顯示，皮爾森相關係數在圖形思考智能測驗與運算思維測驗、Scratch 程式問題解決表現以及程式設計自我效能之間皆呈現顯著正相關。建議未來繼續深入探討圖形智能發展和運算思維能力在程式設計學習中所扮演的角色，並探討其在個別化教學實務上的應用。

關鍵字

運算思維；圖形思考；Scratch 程式設計；問題解決；自我效能

1. 研究背景

運算思維能力是未來重要的能力之一，如何有效的提升運算思維能力更是近年來的熱門研究。Wing（2006）將運算思維這個名詞活絡了起來，也倡議將運算思維當成 K-12 學生的學習基礎，而程式設計就是讓學生體現運算思維的方式。多項研究指出利用圖像化的程式設計語言 scratch 開始廣泛的被使用在學生的運算思維訓練上（Lye & Koh, 2014），視覺化的程式語言工具可以減輕學生的認知負荷，讓學生專注於程式邏輯與程式結構上，無需去擔心寫程式的機制（Kelleher & Pausch, 2005, p. 131）。這讓我們產生了一個研究問題，圖形智商是否與運算思維能力及圖像化程式設計能力有關。另外是否可以尋找一個可以快速又便利施測的圖形智能量表來快速篩選具有較佳運算思維潛力的生員，提早給予適性化教學，讓其有較佳的發展。然而，圖形化介面的程式設計（如 Scratch）是否有助於所有學習者對於程式語言的閱讀理解? 學習者的圖形智能是否在圖形介面程式問題解決中扮演關鍵角色?目前這部分的研究文獻還沒有很清楚的輪廓，因此本研究之目的在於對此關係進行初探。

2. 文獻探討

根據研究指出（Ambrosio, Xavier, & Georges, 2014），運算思維與 Cattell-Horn-Carroll（CHC）智力結構理論（McGrew, 2009）中的三個能力因子有關係。分別是流體推理（Gf）、視覺處理（Gv）及短期記憶（Gsm）。根據 CHC 最新的定義（Schneider & McGrew, 2018）來看，流體推理必須刻意使用心智上的運作來解決嶄新或是現場突發的問題，而這些問題無法用既往的思維來或是習慣來自動解決。視覺處理指的是一種能夠運用心智圖像來解決問題的能力，又稱為心眼。短期記憶最新版名稱被修改為工作記憶（Gwm），定義為一種積極關注在維持與操弄訊息的能力。可以發現這些能力都是用來解決問題，與運算思維的核心目的一致。在 Román-González, Pérez-González, & Jiménez-Fernández（2017）的研究中也得到了一致的結果。因此，研究將著重挑選一個運用視覺、推理元素及記憶元素並能檢測智能的施測工具。

3. 研究方法

研究募集有程式設計課程經驗一年以上的大專學生共 44 人進行研究，其中 22 人為理學院背景，另外 22 人為非理學院背景。研究工具中採用 Bebras2016 國際運算思維測驗題（bebras.org）10 題來衡量運算思維能力。智能測驗則採用圖形思考智能測驗（朱錦鳳，2005）來進行測量。以 6 題 Scratch 程式設計問題來測量程式設計的表現，最後再以程式設計自我效能量表（Tsai, Wang, & Hsu, 2018）檢測其程式設計自我效能。研究參與者依序施測各類測驗後，將全部的結果數據進行皮爾森積差相關統計分析。

圖形思考智能測驗主要的目的是在評量一個人的智能程度。主要包含推理分析的認知能力，以及觀察敏銳程度。訴求是利用多元及非語文的方式、短時間且有效的評量人類多元的智能。測驗中共包含三個分測驗，分別為點線描繪、形狀組合及方格分解。

點線描繪測驗類似藏圖測驗（Witkin, 1971）及外描測驗的綜合。主要應用在測量場地獨立的認知型態及手眼協調精確速度能力。

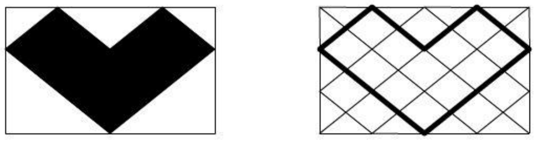


圖 1 點線描繪測驗範例

形狀組合測驗是一個有顏色及形狀的測驗題型。主要在測量知覺能力，與觀察敏銳度有關。

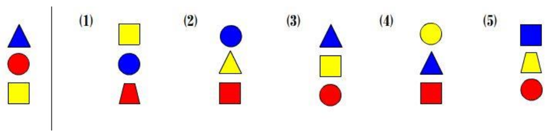


圖 2 形狀組合測驗範例

方格分解測驗類似有些抽象推理方面的測驗題型，但是方格分解涉及更多的圖形分解及組合能力，主要在測量抽象及空間推理分析的能力。

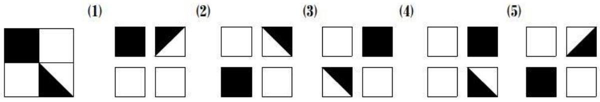


圖 3 方格分解測驗範例

再者，本研究設計六題 Scratch 程式問題，範例如圖四，用以檢核 Scratch 圖形化程式問題解決表現，難度從簡單到困難，問題中所包含的元素羅列如表一。

表 1 Scratch 程式設計問題解決題型分析表

難度	題號	迴圈使用	特性
簡易	一	單一迴圈	迴圈次數固定
	二	單一迴圈	迴圈次數固定 單變數數值變動
中等	三	單一迴圈	迴圈次數變動 單變數數值變動
	四	單一迴圈	條件迴圈 雙變數數值變動
困難	五	巢狀迴圈	條件迴圈 內迴圈次數固定 雙變數數值變動
	六	巢狀迴圈	條件回圈 內迴圈次數變動 雙變數數值變動

圖 4 Scratch 程式設計問題範例

最後，本研究利用程式設計自我效能量表（Computer Programming Self-Efficacy Scale, CPSES）（Tsai, et al., 2018） 檢測學生的程式設計自我效能，CPSES 為含蓋五個面向的五等第量表，檢測學習者對於自己的程式設計能力高低的看法，總信度 Cronbach Alpha 為 0.91.

4. 研究結果與討論

根據各項施測結果，將各測驗之間的皮爾森相關係數整理如下表二所示。

表 2 皮爾森相關係數分析結果摘要表

	程式設計自我效能	Bebras 運算思維測驗	圖形思考智能測驗
Scratch 程式問題解決	.432**	.532**	.327*
程式設計自我效能	-	.415**	.313*
運算思維測驗	-	-	.502**

*：p<0.05 **：p<0.01

從表二可以發現，圖形思考智能測驗與 Scratch 程式設計問題、程式設計自我效能及 Bebras 運算思維測驗都有顯著正相關（r=.313 到 r=.532）。也就說，當一個人圖形智能越高時，越容易在運算思維的表現上較佳，在圖形化程式設計上也較容易有好的表現，同時對自己的程式設計能力也比較有信心。

表 3 圖形思考測驗細項相關分析

	圖形思考智能測驗	點線描繪	形狀組合	方格分解
Scratch 程式問題解決	.327*	.267	.301 （註）	.221
程式設計自我效能	.313*	.236	.224	.362*
Bebras 運算思維測驗	.502**	.205	.479**	.429**

*：p<0.05 **：p<0.01

本研究進一步檢驗圖形思考智能測驗的三個子分項測驗與程式設計表現及 Bebras 運算思維測驗之間的相關係數。結果發現，程式設計自我效能與圖形思考智能測驗有顯著正相關（r=.313*），並在方格分解這一項測驗有著顯著的正相關（r=.362*），在空間抽象及空間推理分析的能力上較佳的人，在對於程式設計的我自效能也就較佳。

Scratch 程式設計表現，則是與整體的圖形智能測驗有著正相關（r=.327*），雖然與各分項測驗上並無統計上的顯著相關，但是在形狀組合的分測驗上，有著較

高的相關係數，並且 p 值落在.05，多少可作為參考依據。Scratch 程式表現與知覺能力、觀察敏銳度有關。

最後 Bebras 運算思維測驗也與圖形智能測驗有正相關（r=.502**），並且與其兩項分測驗：形狀組合及方格分解顯著相關（r=479**, r=429**），這表示 Bebres 運算思維測驗與知覺能力、觀察敏銳度、空間抽象及空間推理能力這幾項能力有關。

另外，有趣的是，在各種測驗中，都沒有與圖形思考智能測驗中的點線描繪有相關。點線描繪主要在測量場地獨立的認知型態及手眼協調精確速度能力，而在以往的研究中，發現場獨立特質的人對於分析及抽象的學科表現較好，男生較傾向場獨立，女生偏向場依賴。（Deress & Futch, 1971）。在目前的研究中似乎並不是那麼有關係，是個可以後續討論的議題。

5. 結論

本研究發現運算思維能力確實與圖形智能有顯著關係，在教學實務上，可以藉由簡易的圖形智能測驗初始評判一個個體是否在運算思維能力上具有潛力，提早給這些有運算思維優異的學生給予適當的教學內容將可以讓其快速發展，節省 1 到 2 年的學習時間（Román-González, Pérez-González, Moreno-León, & Robles, 2018）。另外數據上也顯示人類智能因子中測量抽象及空間推理分析的能力與運算思維有著密不可分的關係，在往後的研究中，可以朝這方向繼續深入追蹤，也能夠運用視覺追蹤技術來探索整個視覺認知的歷程。

6. 致謝

本研究感謝以下科技部計畫編號之研究經費補助：MOST 106-2511-S-003-065-MY3 和 MOST 106-2511-S-003-064-MY3。

7. 參考文獻

朱錦鳳（2005）。**圖形思考智能測驗**。台北：心理出版社。

Ambrósio, A. P., Xavier, C., & Georges, F. (2014). Digital Ink for Cognitive Assessment of Computational Thinking. *Frontiers in Education Conference (FIE)*, 1-7. *IEEE.*

Bebras International Challenge on Informatics and Computational Thinking (2016). Retrieved December 26, 2016, from http://www.bebras.org

DeRussy, E. A., & Futch, E. (1971). Field Dependence-independence as Related to College Curricula. *Perceptual and Motor Skills*, *33*(3_suppl), 1235-1237.

Kelleher, C., & Pausch, R. (2005). Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Computing Surveys (CSUR)*, *37*(2), 83-137.

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.

McGrew, K. S. (2009). CHC Theory and the Human Cognitive Abilities Project: Standing on the Shoulders of the Giants of Psychometric Intelligence Research. *Intelligence*, *37*(1), 1-10.

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.

Román-González, M., Pérez-González, J. C., Moreno-León, J., & Robles, G. (2018). Can Computational Talent be Detected? Predictive Validity of the Computational Thinking Test. *International Journal of Child-Computer Interaction*, *18*, 47-58.

Schneider, W. J., & McGrew, K. S. (2018). The Cattell-Horn-Carroll Theory of Cognitive Abilities. *Contemporary Intellectual Assessment: Theories, Tests, and Issues*, 73-163. US: New York: Guilford Press.

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

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

Witkin, H. A. (1971). *A Manual for the Embedded Figures Tests*. Califonia: Consulting Psychologists Press.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Study on the Factors Affecting Robotics Course Learning Intention based on Computational Thinking

Jing-wen SHAN
East China Normal University, China
Sallyecnu@163.com

ABSTRACT

In order to explore the influencing factors of students' learning intentions in the robotics course based on computational thinking, this study used the technology acceptance model as the theoretical basis, and took 153 primary and middle school students in Shanghai as the research subject, and constructed the learning intention model of primary and middle school students in the robotics course. By analyzing the relationship between variables, it is concluded that students can improve their perceived usefulness to robotics courses by enhancing subjective norms and entertainment perceptions; and improving perceived ease of use by enhancing self-efficacy. The robotics course designers who aim to cultivate computational thinking can optimize in terms of entertainment and interactivity, while paying attention to the gradual progress of programming teaching, analyzing the characteristics of learners, and improving the quality of the course to cultivate students' computational thinking.

KEYWORDS

learning behavioral intentions; TAM; computational thinking; robotics courses

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

基于计算思维培养的机器人课程学习意向的影响因素研究

单靖雯
华东师范大学，中国
Sallyecnu@163.com

摘要

为探究基于计算思维培养的机器人课程中，学生学习意向的影响因素，本研究以技术接受模型为理论基础，以上海市 153 名中小学生作为研究对象，构建了中小学生对于机器人课程的学习行为意向影响因素模型。通过分析各变量间因果关系得出结论：可通过增强主观规范、娱乐感知性来提高学生对于机器人课程的感知有用性；通过增强自我效能感来提高感知易用性。旨在培养计算思维的机器人课程设计者可以在娱乐性，交互性等方面进行优化，同时注重编程教学的循序渐进、进行学习者特征分析，提高课程质量更好地培养学生的计算思维。

关键词

学习行为意向；TAM；计算思维；机器人课程

1. 研究背景

随着信息科技和学习行为的深度融合，中小学机器人课程开办得如火如荼，部分省市已将机器人教学内容纳入到中小学信息技术、综合实践和科学课程中，这代表着机器人教学逐渐进入基础教育领域。中小学机器人教育涵盖多方面内容，项目式学习强调手脑并用，机器人课程成为培养中小学生科学素养和创新能力的重要载体。国内已有数百所学校开设机器人课程，主流教学形式主要集中在编程教学和机器人竞赛两种，旨在培养学生的信息素养和计算思维。

卡耐基梅隆大学（CMU）计算机科学系主任周以真教授，在美国计算机权威刊物《Communications of the ACM》上发表论文，首次提出“计算思维”的概念，他将计算思维定义为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广度的一系列思维活动。（周以真，2006）计算思维代表着一种普遍的认识和一类普适的技能，习得计算思维能让我们“像计算机科学家一样思考”。因此在基础教育阶段更加应该重视计算思维的培养，从编程教育和机器人教学入手，展开基于计算思维培养的机器人课程推广和普及，但是在培养过程中，课程开展过程中，需要思考哪些因素影响课堂的质量和学生的学习行为意向，学习的持续度才是计算思维培养的基本要求。

2. 研究综述

2.1 技术接受模型

美国学者戴维斯（Davis, 1986）提出了学术界著名的技术接受模型（Technology acceptance model，TAM），为了解释影响计算机广泛接受的决定性因素。此模型中，感知的有用性（Perceived usefulness）和感知的易用性（Perceived ease of use）是核心组成部分，TAM 认为感知易用性和感知有用性可以影响用户的行为意向（Behavioral intention）。感知有用性指个体认为使用某特定系统对绩效水平的提高程度，感知易用性指个体认为使用某一特定系统的难易程度。PU 和 PEU 可以一起决定行为意向，PEU 也可以通过改变 PU 从而间接影响个体的行为意向。

文斯卡特和芭拉（Venkatesh & Bala, 2010）将 TAM2 模型和易用性感知影响因素模型合并，提出了 TAM3 模型。TAM3 模型强调技术使用的有用性和易用性，具有比较强的可靠性和有效性。在 TAM3 模型中，影响有用性感知的变量包括主观规范、地位、工作相关性、工作绩效和结果展示性，影响易用性感知的变量包括计算机自我效能感、外部支持、计算机焦虑、计算机爱好、娱乐性感知和客观使用。本文研究中小学生机器人课程学习行为意向受哪些关键因素影响，因此，本量表在 TAM3 模型基础上进行调整，形成了如图 1 所示的“中小学生机器人课程学习行为意向的影响因素模型”。由图可以看出，感知有用性和感知易用性都对行为意向有正向积极的影响，自我效能感，主观规范，娱乐感知性

等因素通过影响感知有用性和感知易用性这两个中介变量来影响行为意向，并且外部变量之间都在相互影响，这些路径是根据 TAM3.0 和课堂观察等多方面共同确定的假设路径，此文在探索影响因素的过程中会对此假设图进行补充和验证。

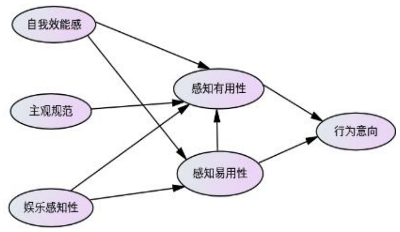


图 1 中小学生机器人课程学习意向的影响因素模型

2.2 计算思维与机器人教学

计算思维是把一个看起来困难的问题重新阐述成一个我们知道怎样解决的问题，如通过约简、嵌入、转化和仿真的方法，计算思维是一种递推思维，它把代码译成数据，又把数据译成代码，计算思维采用抽象和分解迎战浩大复杂的任务或设计复杂的系统，计算思维的本质是抽象和程序化。（陆平，2016）这种抽象和程序化在机器人课程中体现的淋漓尽致，第一阶段，学生不仅需要设计一个机器人的运动轨迹和机械臂的运动方案，还要考虑机器人的运动是否可以用已学会的算法和编程技能来实现。第二阶段，课堂中需求分析环节是对机器人行为的描述，也反映了学生在归纳、抽象方面的积极思维以及明确表达的能力依据需求分析编程，在这一环节，学生不仅可以巩固已学的编程知识，还可以探究学习新技能。第三阶段，进入到机

器人课堂学习的评价环节，学生通过对程序作品的调试与展示，加深理解作品的设计。基于计算思维培养的机器人课程具有开放性和探究性，所以学生具备主动尝试、主动探究、主动表达、主动评价的时间和空间，再通过模仿代码、向他人提问、尝试操作等方法掌握编程技巧。课堂中，教师可以通过一些形成性评价量表引起学生思考、讨论和交流，从而渗透计算思维的教育，提高学生解决问题的能力，真正做到把计算思维融入到机器人课堂中，机器人教学与计算思维的内在联系如何？究竟机器人课程的哪些因素会影响计算思维的培养是本文探讨的核心。

3. 研究过程

3.1. 研究对象

本研究以上海市嘉定区某学校 153 名小学生和初中生为研究对象，采用封闭式结构性问卷，问题设置分七个维度，采用 5 点李氏积分量表，重点探究学生机器人课程学习行为受哪些因素影响，及潜在变量的因果路径。

3.2. 研究方法

文献研究法：本研究查阅中外计算思维培养和机器人课程的文献，参考 TAM3.0，设置自我效能感，主观规范，娱乐感知性，三个外部潜在变量，构建中小学生机器人课程学习行为意向的影响因素模型。

问卷调查法：参考美国 TAM 初始问卷（Venkatesh, 2010），编制职中小学生机器人课程学习行为意向的量表，经过小规模前测，并根据专家建议修订问卷，采用李克特 5 点量表，1 表示完全不同意，5 表示完全同意，量表部分由感知有用性、感知易用性、自我效能感、娱乐感知性、主观规范、行为意向六个维度构成，人口统计学部分 4 个题目，共计 25 题。根据 Cronbach’s Alpha 值检验问卷的信度，通过因子分析得到 KMO=0.805，$p=0.000<0.01$，显著性很高，经过三次修正，形成最终问卷，在线发放。

统计分析法：本研究同时使用社会科学统计软件 SPSS23.0 和结构方程模型分析软件 Amos17.0 对职前教师的有效数据进行描述性统计，信度和效度检验，路径分析，主成分分析等，逐个验证研究假设是否成立，找出显著影响中学学生机器人课程学习意向的因素。

3.3. 研究变量和假设

本研究通过课堂观察结合 TAM3.0 量表发现了几个关键影响学习意向的因素，他们共同影响着学生是否愿意参加后续的机器人课程，他们分别是：2 个中介变量，感知有用性（PU）和感知易用性（PEU）。3 个外部变量，自我效能感（SE）即学生对自己是否可以胜任机器人学习课程的自我认知、娱乐感知性（ENJ）即机器人课程的趣味性、主观规范（SN）即老师和同伴等重要他人是否会推荐学生参加机器人课程。结果变量，行为意向（BI）。每个变量之间的假设路径关系如表 1 所示。

表 1 变量与假设路径

编号	假设的描述	自变量	因变量
H1	娱乐感知性对感知易用性正向影响	ENJ	PEOU
H2	自我效能感对感知易用性正向影响	SE	PEOU
H3	自我效能感对感知有用性正向影响	SE	PU
H4	主观规范对感知有用性有正向影响	SN	PU
H5	感知易用性对感知有用性正向影响	PEU	PU
H6	感知有用性对行为意向有正向影响	PU	BI
H7	感知易用性对行为意向有正向影响	PEU	BI
H8	娱乐感知性对感知有用性正向影响	ENJ	PU

4. 数据分析

4.1. 描述性统计分析

本研究发放问卷 162 份，本研究运用 IBMSPSS 软件进行描述性统计分析，从 162 份样本中反复抽样剔除明显不符合研究的样本，经筛选得到有效问卷 153 份，回收率 94.4%，使用 SPSS23.0 软件针对量表中结构性问题进行分析得到，女性占 23.5%，男性占 76.5%，在机械臂机器人操控，编程方面明显男生比女生有更多的兴趣，男女比例大于 3：1，但经过课堂考察，也不乏具有对于编程机器人感兴趣的女生仍然持续在机器人竞赛中取得好成绩。初中占 28.8%，小学占 71.2%。小学生参加机器人课程的比重较大，初中学习压力大，难度高，参加机器人课程时间相对少一些。从学生参加机器人课程的总体时间来看，半年以内的占 79.7%，半年到一年占 7.8%，一年以上占 12.4%，其中 83.7%的学生只能在课堂中使用机器人，家里没有可以操控的机器人及其硬件套件，人口学变量的描述性统计结果如表 2 所示。

表 2 描述性统计

题目	选项	频率	百分比（%）	平均数	方差
性别	男	117	76.5	1.24	.181
	女	36	23.5		
年级	小学	109	71.2	1.29	.206
	初中	44	28.8		
学习机器人课程的时间（年）	< 0.5	122	79.7	1.33	.471
	0.5-1	12	7.8		
	>1	19	12.4		
家中是否有可供操作的机器人	是	25	16.3	1.84	.138
	否	128	83.7		

4.2. 信度分析

利用 SPSS 中的信度检验（Reliability Analysis），测量模型的内在信度，依据统计学规定：“若 α 大于等于 0.7，则说明测量模型信度很好”。调查问卷中各测量项目的 α 值如表 3 所示，总量表的 Cronbach’s α 系数如表 4 所示，均符合要求，因此本研究的量表信度合格。

表 3 分量表 Cronbach’s α

潜在变量	测量变量	Cronbach’s α
PU	4	0.850
PEOU	4	0.863
SE	4	0.890
SN	3	0.898
ENJ	3	0.847
BI	3	0.904

表 4 总量表 Cronbach’s α

克隆巴赫 Alpha	项数
.928	21

4.3. 相关性分析

在 SPSS 中，使用斯皮尔曼相关分析（Spearman），由于测量的变量属于顺序变量，所以本研究中没有采用皮尔逊相关，而是采用了斯皮尔曼相关分析。本文中，感知有用性和感知易用性是外部变量影响行为意向的中间纽带，与感知有用性存在显著的线性相关关系的

决定性因素为主观规范、自我效能感，娱乐感知性和感知易用性。与感知易用性存在显著的线性相关关系的决定性因素是自我效能感、娱乐感知性，主观规范。具体相关参数如表 5 所示。

表 5 斯皮尔曼相关分析

	PU	PEOU	SE	SN	ENJ	BI
PU	1.000	.457**	.477**	.452**	.506**	.513**
PEOU	.457**	1.000	.639**	.344**	.367**	.453**
SE	.477**	.639**	1.000	.408**	.506**	.602**
SN	.452**	.344**	.408**	1.000	.409**	.421**
ENJ	.506**	.367**	.506**	.409**	1.000	.659**
BI	.513**	.453**	.602**	.421**	.659**	1.000

4.4. 聚合效度检验

为了判断本文中建构的机器人课程行为意向分析模型是否合适，在结构模型分析之前先进行测量模型分析，以验证研究模型的信度和效度。在上一部分已经进行信度的检验，此部分着重针对效度进行检验。所谓效度是指测量工具确实是在测量其所要探讨的观念，而不是其他 观念。本测量变量的 KMO = 0.815，sig = 0.000 如表 6 所示，适合做因子分析;因此可以使用 AMOS 17.0 对中小学机器人课程学习行为意向影响因素的模型进行验证性因子分析，检验 21 个观察变量和 6 个潜在变量的相关性，聚合效度是指同一潜在变量的测量指标会落在同一个因子层面上，且各测量指标之间呈现中高度相关。聚合效度可以用观察变量的因子负荷量、组合信度（CR）和平均方差提取值（AVE）三个值进行评估（吴明隆，2010），三个指标的检验结果如表 7。

表 6 KMO 和巴特利特检验

KMO 取样适切性量数		.815
巴特利特球形度检验	近似卡方	358.043
	自由度	15
	显著性	.000

表 7 聚合效度检验结果

潜在变量	测量变量	因子载荷	AVE	CR
PU	PU1	.739		
	PU2	.746	0.5806	0.8468
	PU3	.818		
	PU4	.742		
PEU	PEU1	.804		
	PEU2	.719		
	PEU3	.836	0.6134	0.8635
	PEU4	.769		
SE	SE1	.829		
	SE2	.854	0.6795	0.8944
	SE3	.839		
	SE4	.773		
SN	SN1	.873		
	SN2	.878	0.7473	0.8987
	SN3	.842		
ENJ	ENJ1	.873		
	ENJ2	.745	0.6626	0.8544
	ENJ3	.819		
BI	BI1	.913		
	BI2	.975	0.7802	0.9132
	BI3	.746		

第一，因子载荷量：因子结构中，因子载荷量值越大，代表观察变量与潜在变量的相关性越高，通常要求围绕每个潜在变量的几个观察变量的因子载荷量要大于 0.5。在本文构建的中小学生机器人课程学习行为意向的量表中，经数据分析 21 个观察变量的因子负荷量如表 7 所示，得到各测量变量的因子载荷值均大于 0.5，说明本论文模型中的观察变量和潜在变量呈高度相关。

第二，组合信度（Composite Reliability）：组合信度的值越大，代表围绕每个潜在变量下的几个观察变量之间的稳定性和内部一致性越高。在本量表中，表 7 汇总了 6 个潜在变量的 CR 值。如果各观察变量代表的潜在变量的组合信度均大于 0.7，则内部稳定性较高，说明此模型组合信度很好。

第三，平均方差抽取值（Average Variance Extracted）：AVE 值大于 0.5 说明聚合效度良好（荣泰生，2009）。在本次中小学机器人学习行为意向的研究中使用 AMOS-CR and AVE 插件计算 AVE 值，如表 7 所示，除了 PU 之外，剩余 5 个潜在变量的 AVE 值均在 0.613~0.78 之间波动，说明本模型科学性较高。

4.5. 区分效度检验

区分效度（Discriminant Validity）:测量模型中每个 AVE 值的平方根如果大于各个潜在变量的相关系数，则说明模型的区分效度良好。如表 8 所示，标注出的对角线上数值（AVE 平方根）均大于同行和列的非对角线数值，即本研究模型区分效度很好。

表 8 区分效度检验结果

变量	PU	PEOU	SE	SN	ENJ	BI
PU	0.762					
PEOU	0.457	0.783				
SE	0.477	0.639	0.824			
SN	0.452	0.344	0.408	0.864		
ENJ	0.506	0.367	0.506	0.409	0.814	
BI	0.513	0.453	0.602	0.421	0.659	0.883

4.6. 结构方程模型

本研究利用 AMOS17.0 软件构建了“中小学生机器人课程学习行为意向的影响因素模型”，其中包括 3 个外部变量，2 个中介变量，1 个结果变量，此步骤主要检验假设模型与样本数据之间的契合程度，对结构模型的适配度进行评估，将 Amos 评估报告中不理想的参数予以调试。在 Amos 17.0 中，采用极大似然法（吴明隆，2010）（Maximum Likelihood Method）获得模型的拟合指标，删除不符合拟合指标的路径，并且通过 Amos17.0 绘制出标准化回归系数路径图，如图 2 所示。通常路径系数分为两种，第一种实际项目的载荷量，即测量模型中潜在变量和观测变量之间系数；第二种是回归系数，即各个潜在变量之间的系数，本文的模型图中为标准化估计系数。

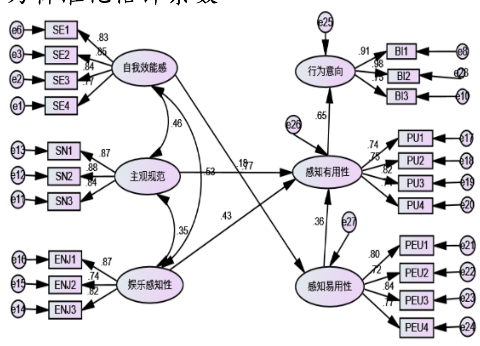


图 2 标准化估计的模型图

4.7. 模型拟合检验和假设检验结果

在 Amos17.0 报表中可以得到模型拟合度的各参数，其中绝对拟合指数有 GFI，RMSEA，RMR 等，其中相对拟合指数有NFI，TLI，CFI等。使用结构方程模型进行分析过程中，模型的检验方法为最大似然法， 使用六个拟合参数检验模型的整体拟合度情况。 六个指数分别是：CMIN/DF、 NFI、RMR、RMESA 近似误差均方根、 GFI 拟合优度、AGFI 调整拟合优度。在模型修正过程中，对于参数的增减，限定是否合适，变量之间的路径和关系进行实当的调整，也是需要参考这些拟合评价指标。GFI 大于 0.9 比较好，RMR 小于 0.05，越小越好，RMSEA 小于 0.1，越小越好，NFI 大于 0.9，越接近 1 越好，其中最终的拟合报表如表 9 所示；假设检验结果如表 10 所示。

表 9 拟合指标

指标	测量值	理想值
CMIN/DF	2.056	<2
RMR	0.052	<0.05
RMSEA	0.083	<0.10
GFI	0.815	>0.90
AGFI	0.764	>0.90
NFI	0.846	>0.90

表 10 假设检验结果

编号	假设的描述	是否成立
H1	娱乐感知性对感知易用性正向影响	否
H2	自我效能感对感知易用性正向影响	是
H3	自我效能感对感知有用性正向影响	否
H4	主观规范对感知有用性有正向影响	是
H5	感知易用性对感知有用性正向影响	是
H6	感知有用性对行为意向有正向影响	是
H7	感知易用性对行为意向有正向影响	否
H8	娱乐感知性对感知有用性正向影响	是
H9	主观规范和娱乐感知性存在强相关	新增

5. *研究结论*

主观规范对感知有用性有积极正向影响，当中小学生身边的同学，老师等重要他人要求或者推荐他们参加机器人课程时，他们会觉得学习机器人比较有用，中小学生的判断很容易受到来自外界的影响，善于利用主观规范这一因素，其机器人课程学习行为意向会显著提升。感知易用性对行为意向有积极正向影响，中小学生如果觉得机器人操作比较容易对行为意向有积极影响，会提高学习效率，如果机器人编程课程和位移操作难度很高，这将降低其对机器人的感知有用性，所以机器人课程开发者需要使机器人操作更加便捷，化繁为简。

自我效能感对学习意向有正向影响，感知到自己操控机器人和编程学习的能力越强，自我效能感越正向。因此在基于计算思维的机器人课程中，编程课程内容难度应该循序渐进，充分考虑到自我效能感对于行为意向的影响。

娱乐感知性对行为意向，感知有用性，感知易用性都有显著正向的影响。机器人课程中要重视娱乐感知性，趣味性的机器人课程越来越得到中小学生的青睐，机器人课程资源应该将趣味性、交互性和可操作性融为一体。

综上所述，在基于计算思维培养的机器人课程中，不仅需要思考课程内容的深度和广度，更加需要注重自我效能感这一重要因素，提升学生的内在学习动机，从而增加其机器人课程的行为意向，同时也需要注重主观规范这一外部影响因素，因为初中小学阶段的学生没有很强的自主学习意识，其行为意向会受到家长，教师和同学等重要他人的影响。感知有用性和感知易用性是决定机器人课程行为意向的中介变量，也应该纳入课程资源整合和机器人教学过程中，同时娱乐感知性对于课程的采纳度，未来课程持续学习的行为意向都有积极正向的影响。

6. 研究局限和展望

第一，样本的局限性，本研究样本过于集中，未来研究可以通过扩大样本来增强研究的普遍性，将信息化发展程度不同的地区进行横向比较研究。第二，平台的局限性，调查样本集中于同一种机器人型号和课程，有其自身局限性。未来样本的选取应该尽量更加多元化。尽管有这些限制，本研究将技术接受模型和中小学生机器人课程学习行为进行联系，而采用数据建模的方式来探究问题的 6 个维度的因素对其学习行为意向的影响程度，为日后研究打下基础。机器人教育是国家对创新型人才培养重要途径，教师在进行机器人课程教学时，不仅仅要对机器人编程知识讲授，更要重视学生计算思维和信息素养的提升。在培养计算思维的同时，更要注重多维度的设计和思考，提升课程质量，让学习效率最大化。

7. 参考文獻

周以真、徐韵文和王飞跃译（2007）。计算思维。**中国计算机学会通讯**，**49**（**3**）， 33-35。

陆平（2016）。计算思维：编程教育的价值追求。 **中小学信息技术教育**，**10**， 19-21。

荣泰生（2009）。 **AMOS 与研究方法**。重庆：重庆大学出版社。

吴明隆（2010）。**结构方程模型：AMOS 的操作与应用**。 重庆：重庆大学出版社。

Davis, F. D. (1986). *A Technology Acceptance Model of Empirically Testing New End－User Information Systems: Theory and Result*. (Doctoral Dissertation, Massachusetts Institute of Technology).

Venkatesh, V., & Bala, H. (2010). Technology Acceptance Model 3 and a Research Agenda on Interventions. *Decision Sciences*, *39*(2), 273-315.

Venkatesh, V. (2000). Determinants of Perceived Ease of Use: Integrating Control, Intrinsic Motivation, and Emotion into the Technology Acceptance Model. *Information Systems Research*, *11*(4), 342-365.

Computational Thinking and Coding Education in K-12

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Micro-Persistence in the Acquisition of Computational Thinking

Rotem ISRAEL-FISHELSON[1], Arnon HERSHKOVITZ[2*]
[12]School of Education, Tel Aviv University, Israel
rotemisrael@tauex.tau.ac.il, arnonhe@tauex.tau.ac.il

ABSTRACT

Taking a Learning Analytics approach, we study the micro-persistence of students in acquiring computational thinking. Micro-persistence is the behavior characterized by being persistent in completing a task with the best possible solution. We do so by analyzing data of 1st-6th-grade children (n=119) who used an online, game-based learning platform (CodeMonkey™). Overall, we find that micro-persistence is associated with task difficulty, and that contextual variables may explain persistence better than personal attributes.

KEYWORDS

persistence, computational thinking, game-based learning, learning analytics, state-or-trait.

1. INTRODUCTION

Computational Thinking (CT), which is a way to solve human problems based on mental tools and computing processes, is considered today an imperative skill for the 21st century (Wing, 2010). Persistence—that is, a learner's will to complete a learning process and to achieve her or his learning goals —is considered as an essential dimension of CT (Barr, Harrison, & Conery, 2011).

In recent years, a wide variety of online game-based challenge-based learning platforms have been developed to support the acquisition of CT concepts (Kim & Ko, 2017). Such platforms take advantage of the Game-Based Learning approach in order to increase motivation (Ibanez, Di-Serio, & Delgado-Kloos, 2014; Kazimoglu, Kiernan, Bacon, & MacKinnon, 2011, 2012), which is closely related to persistence (Moreira, Dias, Vaz, & Vaz, 2013; Vollmeyer & Rheinberg, 2000). In such platforms, which inherently encourage progressing in the game, persistence may serve as an obstacle, as it may come at the expense of investing in each of the game's tasks. Therefore, examining **persistence on the macro level** (i.e. persistence in the learning process) may not reveal the whole picture of knowledge acquirement. This is why it is important to focus on persistence in each component of the learning process, which we defined as **micro-persistence**.

Indeed, it was recently shown that being actively engaged with learning tasks while using an interactive learning platform distinguishes learners who demonstrate a productive persistence from those who just spend time without achieving mastery (Kai, Almeda, Baker, Heffernan, & Heffernan, 2018). Hence, the importance of studying micro-persistence.

A plethora of factors—related to learners' characteristics, programs' structure, technology in use, and the context in which learning occurs—are associated with persistence in online learning platforms (Dalipi, Imran, & Kastrati, 2018; Gazza & Hunker, 2014; Lee & Choi, 2011; Naito, Bezerra, Márcia, & Silva, 2016). In this context, gamification and interactivity—attributes shared by most of the online learning platforms for CT—were proposed as central features that increase persistence and reduce dropout in online learning (Croxton, 2014; Sümer & Aydın, 2018).

Therefore, the main purpose of the current study is to examine the associations between students' micro-persistence and task difficulty, as they expressed while acquiring CT in a game-based learning platform. Moreover, the study examines whether micro-persistence is better explained by contextual variables (State) or alternatively by personal attributes (Trait) – hence is it state-or-trait dependent?

2. METHODOLOGY

2.1. The Learning Platform: CodeMonkey™

CodeMonkey (http://www.playcodemonkey.com) is a game-based challenge-based learning platform for developing CT, aimed mainly at K-12 students. CodeMonkey is unique in that students are required to enter a code from the very first stage (in contrast to the most common, block-based programming approach), however, no previous knowledge in coding is required.

In each level of the game, the learner needs to help the main character, a monkey, catch bananas while overcoming various obstacles. Here, we analyze data drawn from the first four Worlds of the game, teaching basic commands to control the game’s characters' movement (Worlds 1-2), times-loops (World 3), and the concept of variables (World 4). Each of the game's Worlds is built of a few Challenges, and moving forward from one Challenge to another, and from one World to another, is only possible upon completing the former.

Upon submitting a solution to a Challenge, the user gets immediate feedback. A correct solution can award the user with one, two or three Stars: one Star for successfully accomplishing the task (i.e., the monkey collected all the needed bananas), two Stars for a correct solution that also demonstrated the newly-presented concepts, and three Stars for a 2-Star solution which is also the most efficient solution. See Figure for a screenshot of one Challenge, along with 1-, 2-, and 3-Star example solutions. Upon submitting a solution, hints are given in order for the user to improve the code and achieve a higher-Star solution. It is when users attempt to improve their Star-rating—that is, re-trying to solve a Challenge after already solving it correctly—that we identify as micro-persistence.

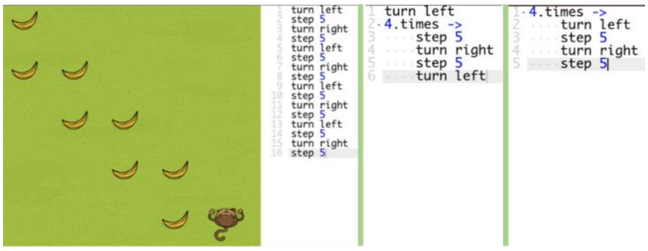


Figure 1. Demonstrating 1-, 2-, and 3-Star solutions to the same Challenge (#25, in the Times-Loops World).

2.2. Population and Dataset

For this study, we analyzed actions of 119 elementary school students from all over Israel, who played the game between March-July 2017 and completed all the Challenges in Worlds 1-3 and at least 10 Challenges in World 4 (only the first 10 Challenges were considered). Note that due to a natural dropout, population size is decreasing as the game progresses. Therefore, we referred only to students who continuously carried out the above-mentioned worlds. All students were connected to the game using their school-provided user accounts, however, we have no information on whether they used it in a formal school context, or on a voluntary basis.

The dataset we analyzed included only correct solution attempts of these users (failed attempts were not fully documented); however, we believe that the number of correct attempts to resolve a task which was already solved, is a good proxy for micro-persistence.

2.3. Variables

2.3.1. Task Difficulty

We have two different measures for task difficulty, referring to both success and effort. These measures are first calculated at the Challenge level, and then they are averaged for each World across its Challenges.

Success. *Maximum Stars Achieved* (across all student's attempts) is first calculated for each student in each Challenge and then averaged for the Challenge across all students. Finally, an average for a World is calculated across the World's Challenges.

Effort. The number of attempts to achieve 2- or 3-Star solutions is another proxy for difficulty. Again, this is first calculated for each student at the Challenge-level, then aggregated to the World-level by taking an average across that World's Challenges. (Note that in this case, not all students had submitted 2- or 3-Star solutions in every Challenge.) So, we get two variables: *2-Stars Attempts* and *3-Star Attempts*.

2.3.2. Student Micro-Persistence

We have two different measures for micro-persistence, indicating an improvement of a correct solution that got either 1- or 2 Stars to a 3-Star solution. Note that this improvement can span over more than a single additional attempt and that we count the actual improvement and not the number of attempts. Each of these measures is first calculated at the Challenge level and then aggregated to the World level (across the World's Challenges) in two ways, as described below.

1- to 3-Star Improvement. We consider cases where the first correct solution got 1 Star and the next better solution got 3 Stars (i.e., no 2-Star solutions were submitted in between). For each student in each Challenge, we set a value of 1 if this student got 1 Star for his first correct solution in that Challenge and their next better attempt was a 3-Star solution; otherwise, we set a value of 0. Then, for each World, we aggregate these values over the World's Challenges, for each student, by either averaging or taking the maximum, which gives us two variables: *1-to-3-Star Improvement Average* and *1-to-3-Star Improvement Binary*, accordingly. Note that for the latter, a value of 0 means that no improvements were done in any of this World's Challenges, while a value of 1 means that improvement was done in at least one Challenge in this World.

2- to 3-Star Improvement. We consider cases where the first correct solution got 2 Stars and the next better solution got 3 Stars (additional 1-Star solutions in between are counted) and calculate this measurement done similarly to the previous one, resulting with two additional variables: *2-to-3-Star Improvement Average* and *2-to-3-Star Improvement Binary*.

(The other two forms of micro-persistence—i.e., 1-to-2-Star improvement, and 1-to-2-to-3-Star improvement, were rarely observed in the data and therefore omitted from the data analysis).

3. FINDINGS

3.1. Descriptive Statistics of the Research Variables

3.1.1. Task Difficulty

Overall, as evident from *Maximum Stars Achieved*, task difficulty is linearly decreasing as the game progresses. This variable's values are rather high, with low variability, and even in World 4 it takes a value of 2.78 (SD=0.32); these findings are summarized in Table 1. This means that generally, students achieve the highest number of Stars. Indeed, in 4361 of 4760 student-Challenge cases (92%), students achieved a 3-Star solution.

Additionally, we can look at the two other variables measuring the number of attempts to achieve a 2- or 3-Star solution, namely, *2-Star Attempts* and *3-Star Attempts*, accordingly. (Note that contrary to *Maximum Star Achieved*, these variables are positively associated with difficulty.) Here, we see that the number of attempts is not linearly increasing along Worlds, but rather that World 2 is more difficult than World 3 (See Table 1).

Table 1. Task difficulty descriptive statistics (n=119).

Difficulty Variable	World Average (SD)			
	1	2	3	4
Max. Stars Achieved	2.97 (0.08)	2.90 (0.15)	2.86 (0.22)	2.78 (0.32)
2-Star Attempts	0.03 (0.05)	0.15 (0.10)	0.12 (0.12)	0.16 (0.14)
3-Star Attempts	1.01 (0.10)	1.12 (0.42)	1.01 (0.28)	1.05 (0.34)

3.1.2. *Students' Persistence*

Recall that we have four micro-persistence variables, measuring persistence in improving from 1-Star to 3-Star solutions and from 2-Star to 3-Star solutions, each having Binary and Average calculations. For all these variables, we observe an increase from World 1 to World 2, and from World 3 to World 4, as well as an increase in the standard deviation. This means that persistence is increasing between these Worlds (albeit with increased variance). However, there are differences in the variables' trend from World 2 to World 3.

When examining improvement from 2-Star to 3-Star solutions, both variables decrease from World 2 to World 3. An improvement from 1-Star to 3-Star solutions behave differently: Its Binary variable—which indicates the very existence of micro-persistence anywhere along the World's Challenges—increases from World 2 to World 3; Its Average variable—which indicates the cumulative effect of micro-persistence along the World's Challenges—is about the same in these two Worlds. Findings are summarized in Table 2.

This irregularity of the micro-persistence trend is associated with the above-mentioned irregularity of task difficulty in World 2. That is, we saw that World 2 yields more attempts than World 3 for 2- and 3-Star solutions; nevertheless, we see that students are more eager in World 2, compared to World 3, to achieve the best solution once they started with a 2-Star solution, but this is not evident for students who first achieved a 1-Star solution.

Table 2. Micro-persistence descriptive statistics (n=119).

Persistence Variable	World Average (SD)			
	1	2	3	4
1-to-3 Average	0.01 (0.03)	0.02 (0.04)	0.02 (0.04)	0.04 (0.06)
1-to-3 Binary	0.08 (0.28)	0.16 (0.37)	0.18 (0.39)	0.35 (0.48)
2-to-3 Average	0.01 (0.03)	0.07 (0.09)	0.05 (0.07)	0.06 (0.08)
2-to-3 Binary	0.09 (0.29)	0.5 (0.5)	0.34 (0.47)	0.48 (0.5)

3.2. *State- and Trait-Models for Persistence*

To understand whether micro-persistence is more related to the World's characteristics (state) or to the student's characteristics (trait), we have constructed two linear regression models (state and trait) for each of the four research variables. Each of the eight models is built on the full dataset of 476 rows.

A State Model tries to predict persistence by Worlds. It uses four variables that denote the game Worlds and is set as follows: for each row in the data, the variable that corresponds to the World documented in this row is set to 1, the others are set to 0. Similarly, a Trait Model tries to predict persistence by the student; this model uses 119 variables, each denotes a student. Note that by using this approach, we refer to the students themselves, or to the Worlds themselves, as the trait/state variables, accordingly, since each of them is a good proxy to the sum of all their characteristics (Baker, 2007).

The models were built using Rapid Miner Studio Version 9.1 and their quality was measured using r^2 (squared correlation), using 10-fold cross-validation.

3.2.1. *Understanding the State Models*

Regarding the 1-to-3-Star improvement, in both State models (Average and Binary), World 1 has a negative coefficient (β=-0.011 at p<0.05 and β=-0.101 at p<0.01, respectively), while World 4 has a positive coefficient (β =0.019 and β=0.168, respectively, both at p<0.001). This means that World 1 is associated with lower persistence compared to the other Worlds, while World 4 is associated with higher persistence compared to the other Worlds. This may be explained by the difference in difficulty between these Worlds, as detailed above (*Section 3.1.1*), with World 4 is more difficult than World 1.

Regarding 2-to-3 Star improvement, we find a similar trend. Here, again, in both State models (Average and Binary), World 1 has negative coefficients (β=-0.036 and β=-0.244, respectively, both at p<0.001), while World 4 has positive coefficients (β=0.019, at p<0.05, and β=0.143 at p<0.01, respectively). This, again, may be explained by the difference in difficulty between these two Worlds. Additionally, World 2 has positive coefficients in both models (β=0.029 and β=0.168, respectively, both at p<0.001); that is, World 2 is associated with higher persistence compared to the other Worlds. This may be related to our previous findings, according to which students in World 2 are more persistent in achieving the best solution once they started with a 2-Star solution, as was detailed above (*Section 3.1.2*).

3.2.2. *Understanding the Trait Models*

When looking at the *Trait models*, we find that for all four variables, there are a few students who came up with significant positive coefficients (no student came up with a significant negative coefficient); these numbers range between 8 (*1-to-3 Average*) to 42 (*2-to-3 Average*). For a student to come up significantly positive in a trait-model means that this student demonstrated higher persistence along the game than other students.

We should highlight that all students who came up with significant coefficients in the *1-to-3 Average* model are also significant in the *1-to-3 Binary* model; this is obvious, based on the definition and construction of the related variables (i.e. those who are, on average, more persistent – are more persistent in essence). Interestingly, we observe an opposite logic relation for the 2-to-3 improvement: All students who came up with significant coefficients in the *2-to-3 Binary* model are also significant in the *2-to-3 Average* model; i.e., those students who were, in principle, persistent throughout the game – were also highly persistent by their action. Of course, this relation is not a necessity.

Additionally, nine students came up with significant coefficients in both the *1-to-3 Binary* model and the *2-to-3 Binary* model. Same goes for seven students who came up significant in both the *1-to-3 Average* and the *2-to-3 Average* models. This means that there are a few students who are more persistent than others in improving their result, no matter what is the initial solution.

3.2.3. The State-or-Trait Question

Overall, we find that **regarding the four research variables, the State models were significant, while the Trait models were not**, indicating a possible state, rather than a trait explanation for persistence. However, the state models had low prediction power, with r^2 ranges between 0.072-0.11. Results are summarized in Table 3 and Table 4 (for the Trait models, we only note how many student-variables were found significant, without mentioning the specific students nor the coefficients; this will be discussed below).

Comparing *1-3 and 2-3 models*, we see that the *2-3 models* have higher r^2 values, but less significant coefficients. This may be a result of the 1-3 data having more 0 values, hence the prediction model can be simpler (predicting 0), but it's more difficult to predict non-0 values.

Table 3. The 1-to-3-Star State and Trait models.

		State	**Trait**
Avg.	r^2	0.072	0.021
	Significant Coefficients	World1 (-0.011*) World4 (0.019***)	8 students
Binary	r^2	0.066	0.014
	Significant Coefficients	World1 (-0.101**) World4 (0.168***)	21 students

* $p<0.05$, ** $p<0.01$, *** $p<0.001$

Table 4. The 2-to-3-Star State and Trait models

		State	**Trait**
Avg.	r^2	0.104	0.046
	Significant Coefficients	World1 (-0.036***) World2 (0.029***) World4 (0.019**)	42 students
Binary	r^2	0.11	0.047
	Significant Coefficients	World1 (-0.244***) World2 (0.168***) World4 (0.143**)	25 students

* $p<0.05$, ** $p<0.01$, *** $p<0.001$

4. DISCUSSION

In this study, we investigated students' persistence while acquiring CT in an online game-based learning platform. Rather than referring to persistence on the macro-level, as commonly done—that is, as the opposite of disengaging from the *learning process*—we explored micro-persistence, which reflects the behavior of keeping students engaged with a *learning task*. This level of persistence has only been little studied (Dumdumaya et al., 2018; Fang et al., 2017). Analyzing persistence at that level allows us to examine the nuanced relationship between persistence and difficulty at the task level. As we measured both students' persistence in a task and the task difficulty using different mechanisms, we were able to demonstrate the complex relationship between these two constructs.

Overall, we observed positive associations between task difficulty and student's persistence. This is in line with recent studies of game-based learning, which found positive links between difficulty and proxies of persistence, like engagement or flow (Hamari et al., 2016; Hung, Sun, & Yu, 2015).

However, our nuanced examination of persistence enabled us to identify a specific set of learning tasks (World 2) in which students demonstrated an interesting behavior: while they were relatively highly persistent in achieving the best solution once they started with a 2-Star solution, this persistence was not evident for students who first achieved a 1-Star solution. Recall that it is the 2-Star solution in which students apply the new knowledge taught. That is, according to our findings, students who have already demonstrated a certain ability to learn new material are the ones who are motivated to achieve the best solution. It may be that those students are intrinsically motivated, as mastery-oriented learners—i.e., those who wish to increase their competence and abilities while mastering new tasks—are characterized by higher persistence, even when facing difficulties, than those who seek a positive judgment of their abilities and performance (performance-orientation). In other words, motivation, mainly intrinsic, has a positive effect on persistence (Dweck, 1986; Garris, Ahlers, & Driskell, 2002). Of course, the alternative explanation may also apply, that is, that playing the game persistently assisted in increasing learners' motivation, as was argued by Hamari et al. (2016) in the context of challenging games in which skills are promoted. Therefore, one important research direction is to further study the causal dynamics of the persistence-difficulty association.

Importantly, the unique behavior described above happened in World 2, which is somehow an extension of World 1; Worlds 3 and 4 teach new concepts. That is, when aiming at extending the learner's knowledge, we observe a situation where those who are already capable of solving the tasks – keep trying until achieving the best solution. While this behavior may seem desirable, it may also increase the knowledge gap between learners, and may eventually harm those who need help the most. Interactive learning platforms (like the one studied here) often have help mechanisms that may assist the struggling students, but, paradoxically, it was found that these mechanisms mostly to promote the medium-achievers (probably represented in our case by those who initially got a 2-Star solution) (Roll, Baker, Aleven, & Koedinger, 2014). Therefore, it is advisable to keep studying the ways in which the knowledge gap may be reduced while using interactive learning platforms.

Examining the state-or-trait question—that is, whether personal or contextual attributes better explain micro-persistence behavior—we overall demonstrate that the former has a stronger predictive value than the latter; this is in accordance with previously mentioned findings regarding the persistence-difficulty association. However, both types of predictions are not necessarily strong. Indeed, the literature indicates that persistence is related to both contextual and personal characteristics. Persistence may be influenced by contextual variables—such as task difficulty, or even a teacher's encouragement—but also by personal attributes, e.g., self-perception of abilities (Schunk, 1996). While intrinsic motivation may be more pronounced, extrinsic factors also have a substantial role (Garris et al., 2002). It may be that some characteristics of the learning platforms—e.g. gaming and interactivity—promote students' engagement; specifically, reward systems (in our case, the Stars) are often mentioned as having positive motivational or metacognitive effect on learning, in a way that increases engagement (Buckley & Doyle, 2016; Mekler, Brühlmann, Opwis, & Tuch, 2013; O'Rourke, Peach, Dweck, & Popović, 2016; Richter, Raban, & Rafaeli, 2015). In future research, we suggest to further explore how external factors such as the gameplay, the rewards system or the challenges' structure affect students' persistence to acquire CT in similar platforms.

Carefully examining the Trait-models sheds some important light on the personal tendency for persistence. We found a subset of the students (as large as 35% of the research population) who are prominently more persistent than the rest of the population. More than that, some students appear to be consistently persistent in attempting to achieve the best solution, no matter what was their starting point. Such a group of highly-motivated students may serve as the basis for understanding the differences in learners' demonstration of persistence. In that light; this may be studied qualitatively.

This study has some practical implications as well. First, educational content developers who wish to keep at a high level of micro-persistence should monitor the difficulty of the learning processes in which learners are involved (Luckin, 2001); optimally, learners should find their flow state, in which challenge and ability to overcome the challenge are matched perfectly (Peterson, Verenikina, & Herrington, 2008). Supporting learners' motivation to solve challenges will subsequently result in improving the acquirement of new knowledge; in this case, CT skills. Second, teachers who wish to use game-based learning in their instruction, should motivate learners to the task and not solely rely on extrinsic motivation to be ignited by the rewarding mechanisms of the game (Peterson et al., 2008; Pucher, Mense, & Wahl, 2002).

5. REFERENCES

Baker, R. S. J. d. (2007). Is Gaming the System State-or-trait? Educational Data Mining through the Multi-contextual Application of a Validated Behavioral Model. *Proceedings of Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling*. Boston, MA: User Modeling Inc., 76-80.

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Buckley, P., & Doyle, E. (2016). Gamification and Student Motivation. *Interactive Learning Environments*, *24*(6), 1162–1175.

Croxton, R. A. (2014). The Role of Interactivity in Student Satisfaction and Persistence in Online Learning. *MERLOT Journal of Online Learning and Teaching*, *10*(2), 314–325.

Dalipi, F., Imran, A. S., & Kastrati, Z. (2018). MOOC Dropout Prediction Using Machine Learning Techniques: Review and Research Challenges. *Proceedings of 2018 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1007-1014.

Dumdumaya, C. E., Banawan, M. P., Mercedes, M., Rodrigo, T., Dumdumaya, C. E., Banawan, M. P., & Rodrigo, M. M. T. (2018). Identifying Students' Persistence Profiles in Problem Solving Task. *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. ACM, 281–286.

Dweck, C. S. (1986). Motivational Processes Affecting Learning. *American Psychologist*, *41*(10), 1040.

Fang, Y., Xu, Y. J., Nye, B., Graesser, A., Pavlik, P., & Hu, X. (2017). Online Learning Persistence and Academic Achievement. *Proceedings of the 10th International Conference on Educational Data Mining*, 312–317.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, Motivation, and Learning: A Research and Practice Model. *Simulation and Gaming*, *33*(4), 441-467.

Gazza, E. A., & Hunker, D. F. (2014). Facilitating Student Retention in Online Graduate Nursing Education Programs: A Review of the Literature. *Nurse Education Today*, *34*(7), 1125-1129.

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

Hung, C.Y., Sun, J. C.Y., & Yu, P.T. (2015). The Benefits of a Challenge: Student Motivation and Flow Experience in Tablet-PC-game-based Learning. *Interactive Learning Environments*, *23*(2), 172-190.

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

Kai, S., Almeda, M. V., Baker, R. S., Heffernan, C., & Heffernan, N. (2018). Decision Tree Modeling of Wheel-spinning and Productive Persistence in Skill Builders. *Journal of Educational Data Mining*, *10*(1), 36-71.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2011). Understanding Computational Thinking Before Programming. *International Journal of Game-Based Learning*, *1*(3), 30-52.

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

Kim, A. S., & Ko, A. J. (2017). A Pedagogical Analysis of Online Coding Tutorials. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science*

Education - SIGCSE '17. ACM, 321-326.

Lee, Y., & Choi, J. (2011). A Review of Online Course Dropout Research: Implications for Practice and Future Research. *Educational Technology Research and Development*, *59*(5), 593-618.

Luckin, R. (2001). Designing Children's Software to Ensure Productive Interactivity through Collaboration in the Zone of Proximal Development (ZPD). *Information Technology in Childhood Education Annual*, *2001*(1), 57-85.

Mekler, E. D., Brühlmann, F., Opwis, K., & Tuch, A. N. (2013). Do Points, Levels and Leaderboards Harm Intrinsic Motivation? *Proceedings of the First International Conference on Gameful Design, Research, and Applications - Gamification*. New York: ACM Press, 66-73.

Moreira, P. A. S., Dias, P., Vaz, F. M., & Vaz, J. M. (2013). Predictors of Academic Performance and School Engagement — Integrating Persistence, Motivation and Study Skills Perspectives Using Person-centered and Variable-centered Approaches. *Learning and Individual Differences*, *24*, 117-125.

Bezerra, L. N. M., & da Silva, M. T. (2016). A Review of Literature on the Reasons that Cause the High Dropout Rates in the MOOCS. *Revista Espacios*, *38*(5).

O'Rourke, E., Peach, E., Dweck, C. S., & Popović, Z. (2016). Brain Points: A Deeper Look at a Growth Mindset Incentive Structure for an Educational Game. *Proceedings of The Third Annual ACM Conference on Learning at Scale*. ACM, 41-50.

Peterson, R., Verenikina, I., & Herrington, J. (2008). Standards for Educational, Edutainment, and Developmentally Beneficial Computer Games. *Digital Media*, *June*, 1307-1316.

Pucher, R., Mense, A., & Wahl, H. (2002). How to Motivate Students in Project Based Learning. *Proceedings of IEEE 6th Africon Conference in Africa*, 443-446.

Richter, G., Raban, D. R., & Rafaeli, S. (2015). Studying Gamification: The Effect of Rewards and Incentives on Motivation. In *Gamification in Education and Business*, 21-46. Cham: Springer.

Roll, I., Baker, R. S. J. d., Aleven, V., & Koedinger, K. R. (2014). On the Benefits of Seeking (and Avoiding) Help in Online Problem-solving Environments. *Journal of the Learning Sciences*, *23*(4), 537-560.

Schunk, D. H. (1996). Self-efficacy for Learning and Performance. *Proceedings of the Annual Conference of the American Educational Research Association*. New York: American Educational Research Association, 1-25.

Sümer, M., & Aydın, C. H. (2018). Gamification in Open and Distance Learning: A Systematic Review. In M. Sümer & C. H. Aydın (Eds.), *Learning, Design, and Technology: An International Compendium of Theory, Research, Practice, and Policy*, 1-16. Switzerland: Springer.

Vollmeyer, R., & Rheinberg, F. (2000). Does Motivation Affect Performance via Persistence? *Learning and Instruction*, *10*(4), 293-309.

Wing, J. M. (2010). Computational Thinking: What and Why? *The Link Magazine*.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Constructing Expert Programming Thinking Process in the Field of Information Engineering, Promoting the Planning of Operational Thinking Teaching Activities

Hsien-sheng HSIAO[1], Yu-an LIN[2*]
[12] Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taiwan
[1] Chinese Language and Technology Center, National Taiwan Normal University, Taipei, Taiwan
[1] Institule for Research Excellence in Learning Sciences, National Taiwan Normal University, Taipei, Taiwan
etlab.paper@gmail.com, lin904287@gmail.com

ABSTRACT
At present, the important issue of engineering technology education is how to solve the problem of engineering talent training (Han, Capraro, Capraro, 2015), how to carry out engineering design teaching, etc., is a very important and urgent problem research problem. Today, technical artificial intelligence, the Internet of Things and other technology industries require a large number of talent. Therefore, it is necessary to cultivate good information engineering talents. The correct procedure should be used to teach students. This study will summarize in the semantic flow chart analysis. Expert course design courses in the field of information engineering can provide students with the best teaching content, train their information engineering talents, and improve their thinking skills.

KEYWORDS
programming thinking program, a flow-map, computational thinking

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

建構資訊工程領域專家程式設計思考程序及促進運算思維教學活動規劃

蕭顯勝[1]，林育安[2*]
[12]科技應用與人力資源發展學系，臺灣師範大學，臺灣
[1]學習科學跨國頂尖研究中心，臺灣師範大學，臺灣
[1]華語文與科技研究中心，臺灣師範大學，臺灣
etlab.paper@gmail.com，lin904287@gmail.com

摘要

目前工程與科技教育重要課題方面為如何解決工程人才培育的缺問題、如何進行工程設計教學的問題等，都是相當重要且迫切需要解決的研究問題，並且現今科技人工智慧、物聯網等科技產業人才需求量大，因此要培養出一位好的資訊工程的人才，該以正確的程序教導於學子們，並且開發一套有關資訊工程教育的課程，本研究將以語意流程圖析法整理歸納出我國資訊工程領域專家程式設計程序並且開發相關課程以利將來教導資訊工程的人才並且提升運算思維能力時，能給予學子們最好的教學內容。

關鍵字

程式設計思考程序；語意流程圖析法；運算思維

1. 前言

在現階段的工程與科技教育重要課題方面，依據近年國際科技與工程教育相關學術期刊的研究趨勢，以及觀察連兩年參與世界工程教育論壇（World Engineering Education Forum, WEEF）與國際科技與工程教師學會研討會（International Technology and Engineering Educators Association Conference, ITEEA Conference）的觀察，如何解決工程人才培育的缺問題（Han, Capraro, & Capraro, 2015）。

課綱當中資訊科技課程「運算思維」（Computational thinking）為重要理念，透過電腦科學相關知能的學習，培養邏輯思考、系統化思考等運算思維，並藉由資訊科技之設計與實作，增進運算思維的應用能，也因此培育有運算思維能力的資訊工程人才是一大重點。

本研究藉由語意流程圖析法訪談，語意流程圖析法是研究者以不具引導性的問題進行訪談，研究者再將受訪者在訪談時所呈現內容依照其順序轉錄成流程圖，以分析受訪者的認知結構，透過此方法，可以對受訪者的陳述進行內容分析，以瞭解受訪者的訊息處理模式。本次對三位資訊工程領域中的專家進行訪談，並且分析歸納探討程式設計工程程序以便往後在資訊工程教育上能有更多的貢獻。

本研究以語意流程圖析法訪談方式對專家進行訪談，了解正確的程式設計工程程序並且運用此程序來設計一套相關課程，進而培育有運算思維能力的相關人才。研究問題如下：

（1） 專家的程式設計工程程序為何？

（2） 藉由專家的程式設計工程程序設計教學內容為何？

2. 文獻探討

2.1. 程式設計思考程序

Tritrakan、Kidrakarn 與 Asanok（2017） 所提出在電腦程式設計課程中所應用的工程設計程序包含以下幾個步驟：（1）清楚定義問題與期望的解決方法；（2）分析有關輸入與輸出變項的相關問題；（3）發展可能解決方法；（4）選擇最佳解決構想；（5）應用電腦程式語言以進行建模；（6）測試與評估模型並找出可能的問題；（7）與其它團隊溝通與討論演算法；（8）改良程式。

而本研究想藉由語意流程圖析法將國內資訊工程專家的程序歸納出一個有助於對於國內資訊工程教育有幫助的程式設計思考程序，並且藉由此程序來開發課程，透過課程培育出有運算思維能力的資訊工程人才。

2.2. 語意流程圖析法

本研究語意流程圖析法所採用的訪談較為單純，以資訊程式設計程序的知識結構方面，主要訪談內容如下:（1）請問當您若要撰寫一支程式時，你所採用的程式設計思考程序為何?包含哪幾個重要的步驟?（2）除了剛剛所提到的這幾個重要步驟之外，請問一下您是否還有需要補充的重要步驟?（3）請問剛剛所提及的這些重要步驟中，彼此之間是否有連結性，若有的話，可否請您說明哪些步驟之間是有相互關聯的呢?（4）請問針對您剛剛所提到的這幾個程式設計思考程序的步驟中，每一個步驟所應該展現的能力為何?請問這些能力之間是否有連結性呢?（針對專家所提的每一個步驟進行詢問）（5）請問針對您剛剛所提的內容，是否還是需要補充說明的呢？

語意流程圖析法是研究者以不具引導性的問題進行訪談，研究者再將受訪者在訪談時所呈現內容依照其順序轉錄成流程圖，以分析受訪者的認知結構，透過此方法，可以對受訪者的陳述進行內容分析，以瞭解受訪者的訊息處理模式，並經過訪談完後將專家敘說的內容打成逐字稿，並且加以歸納內容整理出一個資訊程式設計的工程程序。

2.3. 運算思維

美國國際教育科技協會（The International Society for Technology in Education [ISTE], 2011）認為運算思維是問題解決的過程，它包括：架構問題、邏輯化、抽象化、自動化、效率化、及一般化等特性。

Google for Education 網站則指出運算思維適用於任何一門學科，因其為一系列問題解決之技巧及科技，網站中並列出運算思維包括：心智模式問題拆解、樣式識

別、抽象化、演算法設計、自動化、資料分析表達及模式一般化等。

目前廣為大家接受的運算思維內涵包括：模式一般化與抽象化（包括建模及模擬）、系統化處理資訊、符號系統及表示方法、流程控制的演算法概念、結構化分解問題（模組化）、條件邏輯、效率及執行限制、與除錯及系統性錯誤偵測等。運算思維不等同於程式設計，但學習程式設計為培養運算思維的重要途徑，透過程式撰寫，能實作運算思維中的抽象化、流程控制、模式化、遞迴、重覆、除錯等基本能力。當學生擁有基本程式設計能力後，也同時提升邏輯和運算思維能力（Brennan & Resnick, 2012），吳正已、林育慈（2016）說儘管運算思維之重要性已在各國極力推展的各項政策與活動中不言而喻，大家對運算思維仍有許多疑義，尤其是如何將運算思維的培養落實於資訊科技課程之教學，因此，實有必要釐清運算思維的定義並提供教學方法（林育慈和吳正已，2016），然而此研究就發展出一套運算思維教學活動。

3. *研究方法*

本研究欲探討資訊工程專家的程式設計工程程序及藉由此程序開發一套相關課程並且將來運用於我國高中資訊課程當中，使高中生能更加瞭解資訊工程這塊領域，並且提升運算思維能力，進而萌芽他們對於資訊工程的興趣。

3.1. 實驗設計與實施

本研究採用質性研究，研究對象為國內資訊工程專家分別於高中及大學任職，選取共 3 人進行訪談，研究主要以錄音訪談方式配合「後設重聽法」收集資料，再以「語意流程圖析法」將訪談內容繪製成語意流程圖，最後整理歸納出資訊工程專家的程式設計工程程序。

3.2. 訪談流程

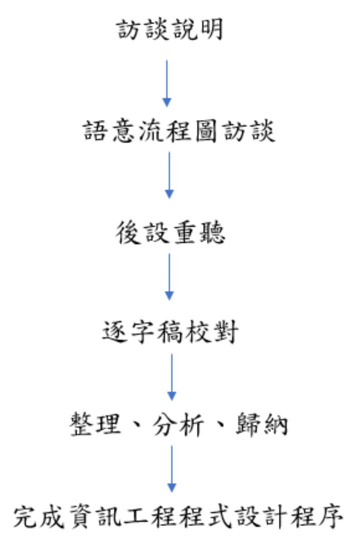


圖 1 訪談流程圖

3.3. 研究工具－語意流程圖析法

為了瞭解資訊工程領域專家對於程式設計程序步驟，將對資訊工程領域專家進行語意流圖，並將結果做分析。本研究將採用 Tsai（1998）、吳穎沺與蔡今中（2005）指出之兩次訪談進行語意流程圖析法，進行的步驟如下：

（一） 第一次訪談：的過程中會進行錄音，訪談題目參考 採 Tsai（1998）、吳穎沺與蔡今中（2005）編製，將與相關領域專家進行討論修改編成。

（二）第二次訪談：訪談的過程中會進行錄音，當完成第一次訪談後，研究者參考 Tsai（1998）後設重聽法，將所錄的音新播放給學習者聽，讓透過錄音紀來確認自己是否所敘述的正確且完整的，若過程中有需要做增加或修改則立刻停止第一次訪談錄音的播放，讓學習者進行補充修改。第二次必須訪談錄音的播放，讓學習者聽完第一次的訪談 ，並成所有增加或修改確認其學習者聽完第一次的訪談，並成所有增加或修改確認其訪談的內容都是正確且完整。

3.4. 訪談結果

本研究將資訊工程領域專家對於程式設計程序步驟進行整理歸納如圖 2 並且將次程序發展出一套資訊工程課程，以下為訪談完三位專家而歸納出的程序步驟圖：

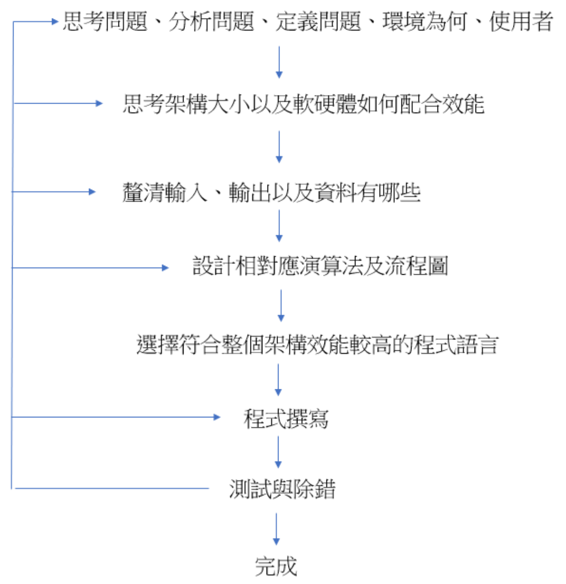


圖 2 專家程式設計程序

訪談三位資訊工程領域專家發現第一步驟皆是定義問題、分析問題及使用者情境為何？再來思考架構以及在哪些情境下需要用哪些硬體設施或是感測器，接著釐清輸入、輸出籍資料，再將剛剛所思考的種種繪製出流程圖，藉由流程圖可以去評估選擇哪個程式語言對整個架構及效能是最有益處的，進而使用此程式語言去撰寫，最終會進行一連串的測試及除錯，然而這階段會與前面階段互相關聯，每一階段都有它的重要含意。本研究將運用此專家程式設計程序來設計課程。

3.5. 實驗研究設計與規劃

3.5.1. 實驗設計與實施

本研究預計以實驗研究進行前、後測之實驗設計，研究對象為台北市某高三學生，六個班級，選取共 240 人，進行為期 8 堂課，每堂課 50 分鐘的教學實驗。 本次實驗活動設計，將學習過程分為三個學習階段，活動將由三個學習階段貫穿整個教學活動如圖 3，進行課程學習階段一：學習目標是熟悉 Micro:bit 介面操作和程式基礎，包含循序結構、條件結構、音效、迴圈結構和變數。學習階段二：學習目標是認識和熟悉 Micro:bit 擴充板的電子元件以及如何應用電子元件的程式基礎，包含 LED 燈、按鈕、伺服馬達控制並且製作自動澆花器如圖 4。學習階段三：學生必須完成廢材機器人運用 Micro:bit 與電子元件的作品，學生必須應用前面所學，透過創意發想、撰寫程式和組裝硬體、測試與修改設計出：（1）Micro:bit 程式（2）電子元件的應用（3）外觀造型，最後產出廢材機器人，本課程廢材機器人知識建構如圖 5，主要分為四大學習方面，首先為摩擦力，學生會了解到摩擦力如何生成的物理知識，以及摩擦力的應用，然而藉由摩擦力的應用學生需要去構思廢材機器人如何走動並且繪製設計圖，完成後學生需要考量軟硬體如何相互結合，並且撰寫 Micro:bit 拼圖程式來驅使馬達轉動進而帶動廢材機器人。

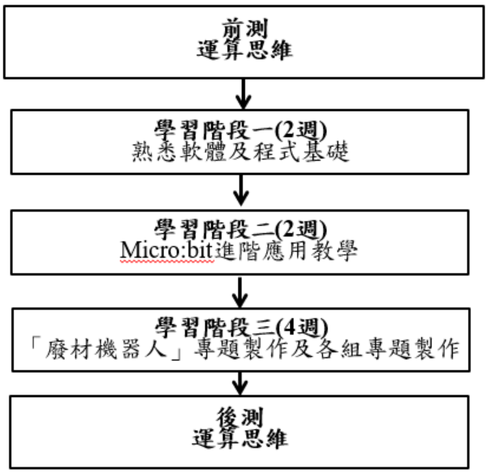


圖 3 實驗流程圖

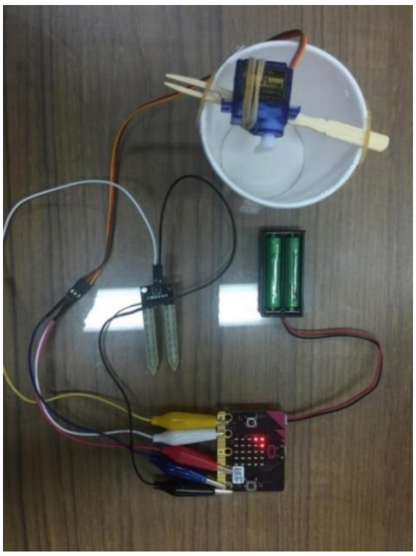


圖 4 自動澆花器

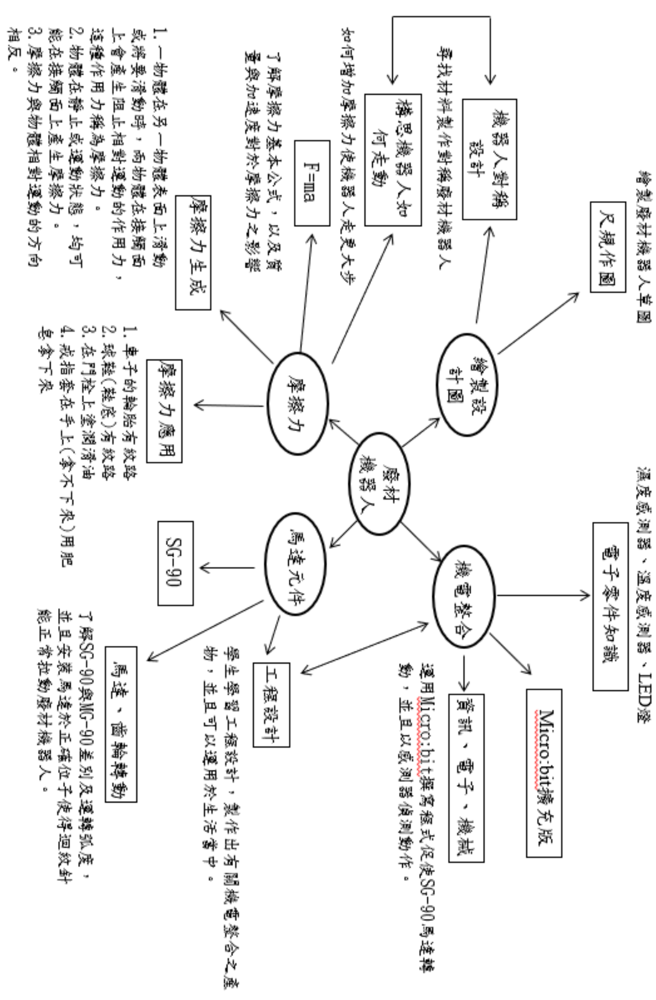


圖 5 知識建構圖

3.6. 研究工具

本研究使用的工具包括： Micro:bit、運算思維量表以下分項敘述之。

3.6.1. Micro:bit

本研究將使用 Micro:bit（如圖 6 所示）為教學工具，Micro:bit 的程式開發，使用 Blockly、JavaScript 或 Python 語言來編輯程式，同時支援藍芽或 Micro USB 連接，使用者能透過電腦、平板或手機，在瀏覽器上即時撰寫程式碼，撰寫完畢後，下載 HEX 檔案至 micro:bit 板子中，即完成程式的上傳燒錄動作。另外，micro:bit 有一項非常便利的特點，無須安裝特殊軟體即可驅動 micro:bit 板子，隨時可以享受編輯程式的樂趣，如同將資料上傳至雲端一樣地簡單快速。這款口袋微型電腦 Micro:bit，可幫助學生學習基本程式編寫，學習如何在螢幕上輸出，學習控制程式的流程，進一步到檔案的控制。Micro:bit 的大小僅 4×5 公分，內嵌 25 顆紅色 LED 做為顯示，有兩個可編程的按鈕，另外還內建可偵測動態的加速計、磁力計、USB 插孔、藍牙及 5 個附鱷魚夾的 I/O 環。為完成廢材機器人成品，本研究教導程式內容包含循序、迴圈、條件結構及語法定義等

圖 6 Micro:bit 實圖

3.6.2. 運算思維測驗量表

國際運算思維能力測驗（International Bebras Contest）幫助了解 8 至 18（三年級至十二年級）學生的運算思維（computational thinking）能力。測驗是為了激起學生對於資訊科學之興趣，同時了解學生是否具備學習資訊科學之性向。本測驗利用淺顯易懂的方式呈現題目，各題皆為情境式任務，讓學習者利用自己既有的知識進行解題。本研究採用 2013 年國際運算思維測驗題目，參考 Brennan 與 Resnick（2012）評量運算思維程式概念的框架。

3.6.3. 實作評量

實作作品是指學生在學習階段三的學習活動必須產出一個廢材機器人作品，如圖 7，作為程式設計與實作能力的展現。因此在評量標準方面，本研究針對程式設計和廢材機器人成品規劃出作品評分的標準。

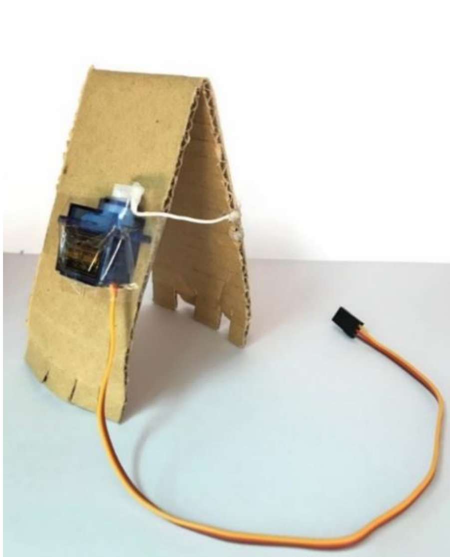


圖 7 廢材機器人

依據實作能力相關文獻探討評分標準，本研究採用 Besemer 與 Trefiger（1981）發展的創意作品評分矩陣（Creative Product Assessment Matrix, CPAM）為實作能力標準，CPAM 由三個向度構成，如圖 8，包含創新性（Novelty）、解決方案（Resolution）、製作與統合（Elaboration & Synthesis）。此項評量指標已被引用多次，遍及於不同領域的實作作品評分。

向度	指標	評分標準
1.創新性（Novelity）	1.1 原創性（Original）	作品是否是自行構思創造，而不是經由複製、改編、模仿衍生作品
	1.2 驚奇性（Surprise）	作品呈現出意想不到的資訊或效果
2.解決方案（Resolution）	2.1 價值性（Valuable）	作品設計構想是否具有意義價值?
	2.2 邏輯性（Logical）	作品設計構想是否符合邏輯?程式設計邏輯是否正常?
	2.3 有用性（Useful）	作品設計構想是否有運用到城市概念?使用是否正確?
	2.4 可理解性（Understandable）	作品設計構想是否易於理解?是否有運用生活相關內容?程式設計是否易於理解?
3.製作與統合（Elaboration & Synthesis）	3.1 基本品質（Organic）	作品是否能正常運作?
	3.2 精緻程度（Elegance）	作品程式指令是否精簡且正確?
	3.3 良好手藝（Well-crafted）	作品外觀或圖示是否美觀?

圖 8 CPAM

4. 資料分析方法

本研究經過教學實驗後，將蒐集之實驗資料以 SPSS 22 統計軟體分別針對「實作能力」與「運算思維」進行分析，統計分析之顯著水準皆為.05，本研究實作能力及運算思維採共變數分析（ANCOVA），以教學模式作為自變項，實作能力後測及學習運算思維後測之資料為依變項，實驗活動進行實作能力、運算思維前測為共變量進行分析。

5. 結語

本研究預期學生藉由以專家程式設計程序去設計的課程來提升運算思維能力，進而培育資訊工程這塊領域的人才並且萌芽他們對於資訊工程的興趣。使學生在學習過程中透過情境以及運用 Micro:bit 去了解到資訊工程，學生學習的面向包括硬體的考量、感測器的使用、程式撰寫、問題分析、運算思維、實作能力等能力。

6. 參考文獻

林育慈和吳正已（2016）。運算思維與中小學資訊科技課程。**教育脈動**，**6**，5-20。

吳穎沺和蔡今中（2005）。建構主義式的科學學習活動對國小高年級學生認知結構之影響-以"電與磁"單元為例。**科學教育學刊**，**13**（**4**），387-411。

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.

Besemer, S. P., & O'Quin, K. (1999). Confirming the Three-factor Creative Product Analysis Matrix Model in an American Sample. *Creativity Research Journal, 12*(4), 287-296.

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, 1, 25.

Han, S., Capraro, R., & Capraro, M. M. (2015). How Science, Technology, Engineering, and Mathematics (STEM) Project-based Learning (PBL) Affects High, Middle, and Low Achievers Differently: The Impact of Student Factors on Achievement. *International Journal of Science and Mathematics Education, 13*(5), 1089-1113.

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning. Proceedings of 2010 IEEE Symposium on Visual Languages and Human-Centric Computing. IEEE, 59-66

Tritrakan, K., Kidrakarn, P., & Asanok, M. (2017). Development and Study the Usage of Blended Learning Environment Model Using Engineering Design Concept Learning Activities to Computer Programming Courses for Undergraduate Students of Rajabhat Universities. *Journal of Education, 11*(1), 46-59.

Tsai, C. C. (1998). Science Learning and Constructivism. *Curriculum and Teaching, 13*, 31-52.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Effects of Gender Differences and Learning Styles on Scratch's Programming Performance and Computational Thinking Ability

Yun-jie JHOU[1], Jung-chuan YEN[2*], Wei-chi LIAO[3]
[13] Graduate School of Mathematics and Information Education in National Taipei University of Education, Taiwan
[2] Department of Mathematics and Information Education in National Taipei University of Education, Taiwan
yjjhou.ntue@gmail.com, jcyen.ntue@gmail.com, heart1543@gmail.com

ABSTRACT

The purpose of this study was to explorer the effects of gender differences and learning styles on learners' scratch programming achievement, motivation and computational thinking ability. The object consisted of 39 sixth-grade students in two classes, including 21 males and 18 females. A quasi-experimental design was adopted and conducted a six-hour teaching experiment for four weeks. The results show that: The game-based learning project approach of this study can effectively promote learning. Gender and learning style have no interaction in scratch learning achievement and computational thinking ability. The effects of gender and learning style on programming learning achievements are not significantly different. In computing thinking, female learners outperform men, but male learners have greater progress. In learning motivation, the accommodator and assimilator style learners are significantly more attention than the converger learners.

KEYWORDS

gender difference, learning style, programming education, computational thinking

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

性別與學習風格對程式設計學習成效與運算思維能力之影響

周昀潔[1]，顏榮泉[2*]，廖韋綺[3]
[13]國立臺北教育大學數學暨資訊教育研究所，臺灣
[2]國立臺北教育大學數學暨資訊教育學系，臺灣
yjjhou.ntue@gmail.com，jcyen.ntue@gmail.com，heart1543@gmail.com

摘要

本研究旨在探討性別與學習風格對學習者 Scratch 遊戲製作的程式設計學習動機、學習成就與運算思維能力之影響。研究對象為國小六年級兩班共 39 位學童（男 21 位、女 18 位），以實證研究進行為期四週共六小時的教學實驗。結果顯示：本研究遊戲式專題程式設計教學能有效促進學習；性別與學習風格在程式設計學習成就及運算思維能力皆無交互作用且無影響；運算思維能力方面，女性學習者表現優於男性，但男性學習者之進步幅度較大；學習動機方面，行動型與理論型學習者在 Scratch 學習動機中的注意力顯著高於應用型學習者。

關鍵字

性別差異；學習風格；程式設計教育；運算思維

1. 前言

性別差異（gender difference）一直是教育研究領域相當重要的議題。如何讓不同性別的學習者有均等的學習機會，促進不同性別的學生在科學學習的興趣、信心、意願與成就，是從事性別與科學教育研究者多年來的期望與目標（Severiens & Geert, 1997；余曉清，1998；余民寧和趙珮晴，2010）。然而，許多學者認為兩性在學習上仍存在許多差異，例如 Bain 和 Rice（2006）即歸納性別因素對科技輔助學習的影響：（1）男性學習者在使用新科技方面，比女性展現較高之動機與興趣；（2）男性學習者傾向將資訊科技視為遊戲與娛樂的載台，而女性學習者則較傾向視為輔助學習的工具；（3）使用電腦的先備經驗，顯著影響兩性學習者接受新科技的態度。

在資訊科學領域的學習情境中，性別因素對學習動機與學習成效的影響，在不同的領域知識及教學情境下呈現不同的結果。Busch（1995）的研究發現：當學習任務較具體且容易時，兩性學習者的自我效能並無顯著差異；然當學習任務較抽象且難度較高時，男性學習者則會顯著比女性具備較高之學習動機與自我效能。相反的，亦有研究結果顯示性別因素對 Scratch 程式設計與運算思維之學習成效並無影響（林欣璇，2018）。與性別因素相比，學習風格是另一個可能影響程式設計學習成效的重要因素。學習風格通常是指學習者個人的學習性向（aptitude）、學習偏好或慣用的學習方式（Jonassen & Grabowski, 1993）。許多學者的研究指出：不同學習風格的學習者在程式設計的學習動機與學習成效上確實存在差異（黃斐曼，2009；蔡孟憲，2010；高榮志，2012；黃樹群，2018）。

過去，資訊科學中的程式設計教學旨在讓學生思考如何運用程式語言的演算法則與設計技巧來解決特定的問題（Fernaeus, Kindborg, & Scholz, 2006）。然而，當資訊科技已無所不在的融入我們日常生活中的環境與事物，也逐漸改變人類的生活方式與運作模式，未來不瞭解資訊科技的演算法則與運用方法的人，其生活作息將處處受限（Wing, 2006）。因此，各國教育目標開始重視在基礎教育層級的程式設計教學，將學習目標設定為培養學生邏輯與抽象思考的能力，希望藉由程式設計教學幫助學童發展在不同領域問題解決的認知基模，於是運算思維（Computational Thinking）的理念於焉而生（Wing, 2008；Google, 2015）。歸納各家定義與主張，運算思維可視為是一種思考歷程，是規劃問題與解決方案的心智活動，而這些問題解決方案能由人、電腦或兩者的結合來實施（Wing, 2011）。

隨著運算思維教育逐漸受到重視，國小階段透過機器人、玩具、遊戲、甚至是自造（Maker）課程的程式設計教學，已成為基礎教育中不可或缺的學習內涵。然而，在此教學趨勢下，性別因素與學習風格是否仍對學習者之學習動機、學習成就、乃至於對運算思維能力產生若干影響，是研究者相當感興趣的議題。

因此，本研究探討不同性別與學習風格之國小學童，在 Scratch 程式設計教學的遊戲製作專題中，對程式設計之學習動機、學習成就與運算思維能力之影響。研究問題如下：

(1)不同性別與學習風格的學習者，其 Scratch 程式設計學習成就是否達顯著差異？

(2)不同性別與學習風格的學習者，在 Bebras 國際運算思維能力測驗之得分上是否達顯著差異？

(3)不同性別與學習風格的學習者，其 Scratch 程式設計學習動機是否達顯著差異？

2. 文獻探討

2.1. 運算思維與程式設計教學

運算思維（Computational Thinking）一詞最早是由 Carnegie Mellon 大學的電腦科學學者 Jeannette Wing 所提出。她最初將運算思維視為是一種運用電腦科學的概念來解決問題、設計系統、以及瞭解人類行為的認知處理能力（Wing, 2006）。Wing 認為資訊科技已逐漸改變人類的生活方式與運作模式，因此未具備此種運算思維素養的人，其生活作息將處處受限。以此觀點而言，運算思維是人類適應未來科技生活所必須具備的電腦科學知識、與運用此種知識進行問題解決的技能。Google 在 Exploring Computational Thinking 專案

計畫中將運算思維定義為是一種包含許多技能、態度、與心智處理的問題解決過程，例如拆解、識別、抽象化及形成演算法則的技能，有自信處理複雜問題與容忍及歧異的態度，及邏輯思考、資料蒐集與分析等心智處理過程（Google, 2015）。林育慈、吳正己（2016）統整各國資訊科技教育的發展趨勢與歸納分析主流的研究，總結運算思維的定義為能有效應用運算方法與工具解決問題之思維能力。

然而，運算思維究竟該如何教？教什麼？學者認為應該從基礎教育層級的程式設計課程教起（劉明洲，2017）。教師透過設計一些解決日常生活問題的專題式程式設計學習任務，或是透過能實體操作的程式遊戲及模擬動畫，均適合用來培養學童的運算思維能力（Hsu, Chang, & Hung, 2018；謝宗翔和顏國雄，2017）。此外，隨著資訊科技的不斷演進，程式語言的典範與類型也日趨多元，學者專家亦建議程式設計課程的內涵，不應再著重程式語言的語法、結構與設計技巧，而是應強調從具體動手的實作經驗中學習問題拆解（decomposition）、模式識別（pattern recognition）、抽象化（abstraction）及演算法則（algorithms）的邏輯思考與問題解決能力。

2.2. 性別差異與程式設計學習

有關性別的研究將觀察變項聚焦於兩性在學習成就、態度、動機、興趣、焦慮、自信與學習行為表現上之不同，多數研究認為兩性確實存在許多學習上的差異（余曉清，1998；余民寧和趙珮晴，2010；Severiens & Geert, 1997；王敏娟，2007）。Fan 和 Li（2005）的研究指出：台灣女學生在資訊科學相關課程中，對電腦的興趣與自信普遍低於男學生，而男學生在創新科技使用之自我效能與接受度上顯著比女學生為高。然而，有研究指出有趣且值得探討的現象，那就是女學生的自我效能與學習成效間之相關比男生來得高。換句話說，女學生的自我效能比男學生更能準確的預測其學習成效，而男學生雖對資訊相關課程展現高度的自信，然其學習表現並不見得能與其自信相符（Anjum, 2006）。

Denner（2011）認為社會刻板印象及基礎教育時期缺乏資訊科學相關的學習經驗，是造成兩性學習者在程式設計之學習動機與成效上產生差異的主因。Rubio、Romero-Zaliz、Mañoso 和 Angel（2011）對大一基礎程式設計課程的調查研究顯示：女性學習者對程式設計課程感興趣且願意投入學習的比例顯著低於男性學習者。Çakır、Gass 、Foster 和 Lee（2017）的研究發現：透過遊戲設計的學習任務，能增進女性學習者學習程式設計的態度與自信，並進而改善原本兩性在學習成效上的差異。綜上所述，性別差異在資訊科學乃至於程式設計的學習是普遍存在的現象，然藉由探討形成差異的成因為何，實務上仍能透過教學設計與課堂中的教學策略，改善性別因素對學習動機與成就的影響。

2.3. 學習風格與程式設計學習

吳百薰（1998）歸納有關學習風格的理論與主張，依據各家定義將之區分為學習情境、行為模式、策略、情意、與多元取向等五大類別。其中，學術研究中以行為模式研究取向的 Klob（1976）體驗式學習，及被歸類為認知和訊息處理研究取向的 Felder-Silverma 學習風格理念，最常為教育實務研究者所採納。Klob 認為學習是一種連續的循環過程，學習風格是學生在具體經驗、觀察後反應、形成抽象概念及行動後獲得新經驗等四個循環當中的行為表現（如圖 1）。本研究教學設計以 Klob 的體驗式學習為理念，因此採用其學習風格量表進行研究對象之學習風格分類。

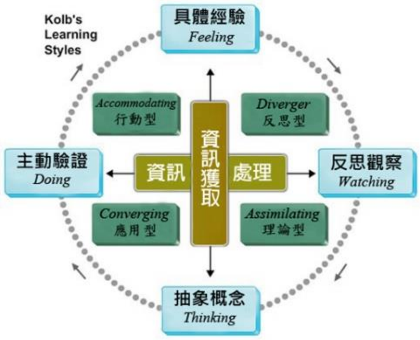


圖 1 Kolb 以體驗式學習為架構之學習風格分類

有關學習風格對程式設計學習動機與成就之影響，許多研究認為存在差異。蔡孟憲（2010）以 Klob 學習風格量表，對國小學童 Scratch 程式設計之教學發現：行動型的學習者在邏輯推理能力上明顯優於理論型；而行動型、反思型及應用型的學習者，在學習成就上皆顯著優於理論型。高榮志（2012）以 Klob 學習風格，探討學習風格對遊戲式學習之學習成效的影響，結果顯示理論型學習之學習表現顯著優於反思型。黃樹群（2018）採用 Felder-Silverman 的學習風格問卷，發現高中生程式設計教學中：感受型學習者在程式設計成就測驗的表現，顯著優於直覺型學習者；而視覺型學習者在程式概念的理解上則表現優於文字型學習者。

3. *研究方法*

3.1. 研究對象與環境

本研究之研究對象因受限於班級人數的影響，因此以原班組成之立意抽樣方式，隨機選擇臺北市某國小六年級兩班共 44 位學習者參與，扣除資料不全之樣本後，實際參與學習者為男生 21 位（53.8%）、女 18 位（46.2%）（含 1 名學習風格無法分類者）。研究對象年齡層介於 11-12 歲之間，在皮亞傑認知發展理論中，處於具體運思期後期和形式運思期初期之間，已能夠進行具體事物的思考以及基本的邏輯運算。

3.2. 程式設計教學活動設計

本研究之 Scratch 教學活動以製作 Scratch 遊戲為課程主題，將學習者學習到的基本程式概念，應用於專案作品中，課程說明如下：

3.2.1. Scratch 程式設計教學

介紹 Scratch 程式介面、指令積木區的八大類別、舞台操作及角色設置等，並提供實際操作與體驗的機會。接著讓學習者實際體驗教師所提供的「Scratch 遊戲設計專題範例」之專案檔，透過觀察分析專案檔案中的

遊戲角色與執行效果，學習者能從**玩遊戲**與**設計遊戲**的過程，瞭解遊戲內容的程式設計概念。

3.2.2. Scratch 遊戲設計專題作品

學生透過設計遊戲專題之障礙物和角色移動的程式指令學習序列和迴圈概念。在學習者實際操作演練後，研究者再進行統一講解與概念澄清，讓學習者瞭解程式指令的差異性。學習者經反覆的模擬與操作後，再從角色的路徑偵測及終點的程式設定學到多重條件判斷的概念，若有疑問再進一步與教師或同儕討論。

3.3. 研究工具

本研究之 Scratch 學習成就測驗為根據實際授課內容自行編製而成，題目包含程式概念的基本知識、Scratch 指令積木的意義與用法、依照題意選出適當的指令積木、或是配合題意進行除錯使程式能正確執行等。測驗共 10 題，每題 10 分，滿分為 100 分。此外，運算思維能力測驗乃依據 2016 年及 2017 年 Bebras 國際運算思維能力測驗公告之題目，考量其題目隱含的資訊科學概念和難易度，篩選出適合實驗對象的題目編製而成。測驗依難易度分別以簡易 2 題、中等 2 題、稍難 1 題共 5 題，編製成運算思維能力前後測驗卷。

本研究之程式設計學習動機問卷採孫琇瑩（2000）改編自 Keller 未出版的 IMMS（Instructional Materials Motivational Scale）。問卷內容涵蓋 Keller 的 ARCS 動機模式四大要素：Attention（引起注意）、Relevance（切身相關）、Confidence（建立信心）以及 Satisfaction（獲得滿足）。本問卷共36題，信度方面整體問卷 Cronbach’s α=0.86，具良好信度。

4. 結果與討論

4.1. 成對樣本t檢定顯示學習者之學習成就表現與運算思維能力均有顯著進步

本研究首先檢驗程式設計教學實驗後，全部參與學生在 Scratch 學習成就測驗與 Bebras 運算思維能力測驗之前、後測分數是否達顯著差異。表 1 與表 2 分別為學習成就測驗與運算思維能力測驗之成對樣本 *t* 檢定（Paired-Samples t test）結果。

表 1 Scratch 學習成就前、後測之成對樣本 *t* 檢定結果

學習成就	個數	平均數	標準差	*t* 值	*df*	顯著性
前測	39	52.82	15.72	-2.439	38	.020*
後測	39	61.03	15.69			

表 2 Bebras 運算思維前、後測之成對樣本 *t* 檢定結果

運算思維	個數	平均數	標準差	*t* 值	*df*	顯著性
前測	39	51.67	31.82	-4.342	38	.000**
後測	39	66.92	25.67			

從表 1 及表 2 之成對樣本 t 檢定結果可知：本研究實施之 Scratch 程式設計教學活動，學習者之學習成就與運算思維能力後測成績均高於前測成績，且雙尾檢定之顯著性分別達顯著與非常顯著之水準，由此可知本研究之教學成效相當良好。

4.2. 性別與學習風格在學習成就之二因子共變數分析中無交互作用且主效果均未達顯著差異

其次，本研究以獨立樣本二因子共變數分析（two-way ANCOVA）探討性別因素與學習風格對 Scratch 學習成就測驗表現上是否存在交互作用。首先，表 3 為不同性別與學習風格之學習者，在 Scratch 學習成就測驗前後測之描述性統計摘要。

表 3 Scratch 學習成就前、後測之描述性摘要

變項	組別	人數	前測 平均數	前測 標準差	後測 平均數	後測 標準差
性別	男生	21	52.86	18.75	58.57	18.24
	女生	18	52.78	11.79	63.89	11.95
學習風格	行動型	9	50.00	21.21	62.22	13.02
	反思型	8	46.25	15.98	58.75	19.59
	應用型	11	60.00	11.83	61.82	18.34
	理論型	10	53.00	13.37	62.00	13.98

接著以 Scratch 學習成就測驗之前測成績為共變數，檢驗性別與學習風格對成就測驗後測成績之交互作用是否達顯著差異，經檢驗符合迴歸係數同質性檢定之條件後，獲得如表 4 之共變數分析結果。

表 4 性別與學習風格對學習成就之二因子共變數分析

變異來源	型 III 平方和	自由度	均方	*F*	*p*
成就測驗前測	209.494	1	209.494	.744	.396
性別	213.934	1	213.934	.759	.391
學習風格	25.116	3	8.372	.030	.993
性別×學習風格	701.938	3	233.979	.830	.448
誤差	8170.982	29	281.758		

註：R 平方= .115（調整後 R 平方= -.129）

從表 4 結果可知：性別與學習風格在 Scratch 學習成就測驗上之交互作用 F（1, 29）= .830，p = .448，未達顯著水準。表示在排除成就測驗前測的影響後，性別變項對 Scratch 學習成就測驗後測的影響，不會因學習風格之不同而有所不同，反之亦然。此外，除了交互作用不顯著外，性別與學習風格兩因子個別之主效果亦未達顯著差異。由此可知，本研究之教學實驗雖有不錯的學習成效，然男性與女性、以及四種不同學習風格之學習者，在 Scratch 程式設計學習表現上並無差異。

4.3. 性別與學習風格在運算思維能力之二因子共變數分析中無交互作用但性別因素呈現有趣現象

其次，本研究以獨立樣本二因子共變數分析（two-way ANCOVA）探討性別因素與學習風格對 Bebras 運算思維能力測驗表現上是否存在交互作用。表 5 為性別與學習風格，在運算思維前、後測之描述性統計摘要。

表 5 Brbras 運算思維能力前、後測之描述性摘要

變項	組別	人數	前測		後測	
			平均數	標準差	平均數	標準差
性別	男生	21	37.86	31.33	57.86	28.22
	女生	18	67.78	24.45	77.50	17.76
學習風格	行動型	9	47.22	34.20	60.00	28.72
	反思型	8	52.50	27.12	72.50	17.32
	應用型	11	52.73	35.52	71.82	28.22
	理論型	10	50.50	33.37	61.50	27.49

接著以 Bebras 運算思維能力之前測成績為共變數，檢驗性別與學習風格對運算思維能力後測之交互作用是否達顯著差異，經檢驗符合迴歸係數同質性檢定之條件後，獲得如表 6 之共變數分析結果。

表 6 性別與學習風格對運算思維之二因子共變數分析

變異來源	型 III 平方和	自由度	均方	F	p
運算思維前測	6845.129	1	6845.129	19.086	.000*
性別	285.141	1	285.141	.795	.380
學習風格	1168.796	3	389.599	1.086	.370
性別×學習風格	285.676	3	95.255	.266	.850
誤差	10400.942	29	358.653		

註：R 平方= .579（調整後 R 平方= .463）

從表 6 結果得知：性別與學習風格在 Bebras 運算思維能力測驗之交互作用 F（1, 29）= .266，p = .850，未達顯著水準。表示在排除運算思維能力前測的影響後，性別變項對運算思維能力後測的影響，不會因學習風格之不同而有所不同，反之亦然。此外，除了交互作用不顯著外，性別與學習風格個別主效果亦無顯著差異。然而，值得注意的是作為共變數之運算思維前測成績是達顯著差異的（F = 19.086, p < .01）。我們以 t 檢定及變異數分析分別進行性別與學習風格對運算思維前測成績之分析發現：性別在運算思維前測成績有顯著差異存在，且女生成績（平均數 67.78）優於男生（平均數 37.86），而學習風格則無顯著差異存在。

這是個有趣的現象：女性學習者在運算思維前測成績較優，但在共變數分析之結果卻無顯著差異，代表男性學習者在此教學實驗過程中可能進步的幅度較大，以致於造成性別因素在共變數分析中，對後測成績產生無顯著差異的結果。

表 7 性別因素對學習成就測驗後測-前測之 t 檢定結果

性別	個數	後測-前測 平均數	後測-前測 標準差	t 值	df	顯著性
男生	21	20.00	22.14	1.481	37	.147
女生	18	9.72	20.97			

為驗證此假定，本研究改以運算思維後測成績-前測成績（進步幅度）為依變項，進行性別因素之 t 檢定分析，結果發現性別因素在運算思維能力進步幅度之 t 檢定雖未達顯著（t = 1.481, 顯著性為.147），然男生平均進步 20.00 分、女生則平均僅進步 9.72 分（表 7），顯然男性學習者在教學實驗中有較大的進步幅度。

4.4. 性別與學習風格在程式設計學習動機之二因子多變量分析中無交互作用但學習風格主效果顯著

本研究以獨立樣本二因子多變量變異數分析（two-way MANOVA）探討性別因素與學習風格對 Scratch 程式設計學習動機之影響，表 8 為描述性統計摘要。

表 8 Scratch 程式設計學習動機之描述性統計摘要

依變項	性別	學習風格	人數	平均數	標準差
注意力	男生	行動型	5	3.40	.56
		反思型	5	3.47	.93
		應用型	7	3.20	.24
		理論型	3	3.78	.25
		總計	20	3.40	.56
	女生	行動型	4	3.98	.39
		反思型	3	3.42	.42
		應用型	4	2.88	.42
		理論型	7	3.82	.57
		總計	18	3.58	.61
相關性	男生	行動型	5	3.22	.38
		反思型	5	2.91	.72
		應用型	7	3.19	.55
		理論型	3	3.81	.63
		總計	20	3.22	.63
	女生	行動型	4	3.36	1.04
		反思型	3	3.48	.46
		應用型	4	3.08	.43
		理論型	7	3.48	.77
		總計	18	3.36	.70
信心感	男生	行動型	5	3.20	.57
		反思型	5	2.91	.72
		應用型	7	3.19	.55
		理論型	3	3.81	.63
		總計	20	3.22	.63
	女生	行動型	4	3.47	.53
		反思型	3	3.30	.89
		應用型	4	2.64	.78
		理論型	7	3.41	.70
		總計	18	3.23	.73
滿足感	男生	行動型	5	3.67	.86
		反思型	5	3.23	1.12
		應用型	7	3.29	.44
		理論型	3	3.89	.79
		總計	20	3.46	.78
	女生	行動型	4	3.75	1.02
		反思型	3	3.35	.94
		應用型	4	3.17	.48
		理論型	7	3.75	1.03
		總計	18	3.49	.85

接著進行性別與學習風格共變異數矩陣等式的 Box's M 檢定，同質性檢定結果未達顯著水準（F = 1.410，顯著性為.072），故繼續進行多變量變異數分析，獲得受試者間效應項檢定分析結果如表 10 所示。

表 9 Scratch 程式設計學習動機之變異數同質性檢定

Box's M	F	df1	df2	顯著性
71.672	1.410	30	842.425	.072

*p<.05

表 10 Scratch 程式設計學習動機之受試者間效應檢定

來源	依變項	型III平方和	*df*	平均平方和	*F* 值	顯著性
性別	注意力	.032	1	.032	.111	.741
	相關性	.021	1	.021	.056	.815
	信心感	.047	1	.047	.108	.745
	滿足感	.008	1	.008	.010	.921
學習風格	注意力	3.234	3	1.078	3.805	.020*
	相關性	.541	3	.180	.487	.694
	信心感	2.469	3	.823	1.870	.156
	滿足感	2.625	3	.875	1.111	.360
性別×學習風格	注意力	.995	3	.332	1.171	.337
	相關性	.111	3	.037	.100	.959
	信心感	1.460	3	.487	1.106	.362
	滿足感	.541	3	.180	.980	.417
誤差	注意力	8.498	30	.283		
	相關性	11.117	30	.371		
	信心感	13.201	30	.440		
	滿足感	23.627	30	.788		

$^*p$<.05

從表 10 結果得知：性別與學習風格在學習動機的四個分向度之交互作用均未達顯著差異，且性別因素之主效果亦無顯著差異，只有學習風格在學習動機分向度之注意力上達顯著差異（F = 3.805，p = .020），本研究經 LSD 法進行事後分析後獲得如表 11 之結果。

表 11 學習風格在學習動機注意力之成對事後分析

依變項	學習風格	學習風格	平均差異	標準誤	顯著性
動機之注意力向度	行動型	反思型	.246	.264	.359
		應用型	.649	.244	.013*
		理論型	-.112	.256	.665
	反思型	行動型	-.246	.264	.359
		應用型	.403	.256	.126
		理論型	-.358	.267	.191
	應用型	行動型	-.649	.244	.013*
		反思型	-.403	.256	.126
		理論型	-.761	.248	.005*
	理論型	行動型	.112	.256	.665
		反思型	.358	.267	.191
		應用型	.761	.248	.005*

$^*p$<.05

表 11 顯示：在學習動機之注意力向度中，行動型學習者顯著高於應用型學習者（平均差異為.649，顯著性為.013）；此外，理論型學習者亦顯著高於應用型學習者（平均差異為.761，顯著性為 p = .005）。

5. 結論及建議

本研究旨在探討性別與學習風格對學習者 Scratch 遊戲製作的程式設計學習動機、學習成就與運算思維能力之影響。經二因子之共變數與多變量變異數分析後，獲得如下幾點結論：

5.1. 遊戲式專題程式設計教學能有效促進學習，而性別與學習風格對學習成就似無影響

經前述實驗數據之分析結果歸納，本研究的 Scratch 遊戲式專題程式設計教學活動，能有效促進學習者在學習成就測驗與 Bebras 運算思維能力測驗之學習表現。而實驗結果支持性別與學習風格兩項變因，在程式設計學習成就上並無交互作用，且不同性別與不同學習風格之學習者，其學習成就並無差異。

5.2. 性別與學習風格對運算思維能力雖無交互作用及顯著影響，但女生表現較好、男生進步幅度較大

本研究實驗數據結果顯示：性別與學習風格在運算思維能力上並無交互作用，且不同性別與不同學習風格之學習者，其運算思維能力並無差異。研究者原本假定 Scratch 的程式設計教學，能協助學習者發展問題解決之思維與方法，提升學習者在運算思維能力測驗上的得分。然本研究之運算思維能力測驗篩選自 Bebras 國際運算思維能力挑戰賽的題目，需要更高階的思辨能力與解題方法，本研究整體教學時程較短，學習者可能尚未發展精熟的問題解決能力，致成效不顯著。

此外，從教學現場的觀察紀錄發現，Bebras 運算思維挑戰賽題目之敘述字數比一般學生熟悉的測驗題目繁複，作答者需耐心閱讀並理解題目內容後才能作答。男性學習者普遍對於題目的冗長感到不耐煩，且有較高比例採用略讀的方式作答；反觀女性學習者則比較能耐心閱讀題意後才作答，這或許是女性學習者在運算思維前、後測的平均得分皆較高的原因。相反的，由於男性學習者前測成績較差，在經過遊戲式專題教學活動後，對程式設計產生較高的學習動機，故在運算思維的後測表現上有進步幅度較大的發展空間。

5.3. 行動型學習者與理論型學習者在程式設計學習動機中的注意力皆優於應用型學習者

本研究實驗數據結果顯示：不同學習風格之學習者在程式設計學習成就及運算思維能力上均無顯著差異；而學習動機方面，行動型學習者與理論型學習者均在注意力向度上顯著優於應用型學習者。

依據 Kolb 及 Kolb（2005）對學習風格的描述：行動型（accommodator）學習者學習時傾向使用具體經驗和主動實驗，喜歡冒險與嘗試，習慣以直覺和嘗試錯誤的方法實際動手去做；理論型（assimilator）學習者傾向使用抽象概念和省思觀察來學習，具有較強的歸納統整以及架構理論的能力，喜歡邏輯性的概念學習；應用型（converger）學習者則傾向使用抽象概念和主動實驗，善於假設驗證的演繹推理，來尋求問題解決的方法。本研究教學活動內涵，除了學習使用 Scratch 程式積木將抽象的問題解決演算法則轉化成具體可執行的遊戲角色動作外，專題目標為無標準答案的開放式遊戲設計任務，專題製作過程需經常與同儕討論及合作，且專題程式設計過程中，必須反覆進行除錯來修正錯誤的程式執行結果。因此，從學習任務觀之，行動型與應用型學習者的動手做特質，似乎較適合程式撰寫與除錯的學習任務，而理論型與應用型的抽象思考特質，似乎較適合問題解決的演算法則設計。

以文獻回顧檢視實驗結果，研究者合理推論開放式的專題遊戲設計，且同時需兼顧邏輯思考與動手實作的程式設計任務，似乎比較容易吸引行動型與理論型學習者的專注與投入，較不利於應用型學習者偏好單一解答的學習風格。

新世紀的教育思潮是以學習者為中心的理念架構，教師應運用不同的教學策略或是活動安排來促進不同性別與學習風格的學習者，提升其學習動機與學習成效。未來研究建議增加實驗教學之參與人數，以提高研究結論之推論與適用性。其次，性別與學習風格對運算思維學習成效方面，建議考慮將先備知識納入第三因子探討，藉以修正原研究架構中女性學習者前測成績達顯著差異對整體模式之影響。

6. 致謝

本研究承蒙科技部 MOST 106-2511-S-152-001 與 MOST 107-2511-H-152-009 專題研究計畫之經費補助，謹此致謝。

7. 參考文獻

王敏娟（2007）。在線學習中的性別差異、對話風格和平等參與。**中國遠程教育**，**2**，25-29。

佘曉清（1998）。科學教育與性別差異的省思。**兩性平等教育季刊**，**2**，51-57。

余民寧和趙珮晴（2010）。選擇科學職業意圖的性別差異分析-以 TIMSS 2003 臺灣八年級學生為例。**諮商輔導學報**，**22**，1-29。

林育慈和吳正己（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**，**6**，5-20。

林欣璇（2018）。**應用同儕教導法在程式教學對於國中生運算思維及學習成效之影響**（未出版碩士論文）。臺北市：臺北市立教育大學。

吳百薰（1998）。學習風格理論探究。**國教輔導**，**37（5）**，47-53。

高榮志（2012）。**認真遊戲教學對不同學習風格及電腦自我效能七年級學生學習成效影響之研究**（未出版碩士論文）。彰化縣：國立彰化師範大學。

黃樹群（2018）。**不同學習風格高中生以任務導向策略學習程式設計的學習成效**（未出版碩士論文）。彰化縣：國立彰化師範大學。

蔡孟憲（2010）。**Scratch 程式設計對國小五年級學生幾何概念及邏輯推理能力的影響**（未出版碩士論文）。臺北市：臺北市立教育大學。

劉明洲（2017）。創客教育、運算思維、程式設計～幾個從「想」到「做」的課程與教學設計觀念。**臺灣教育評論月刊**，**6（1）**，138-140。

謝宗翔和顏國雄（2017）。**偷插電的資訊科學-教師手冊**。台北：中華民國軟體自由協會。

Anjum, R. (2006). The Impact of Self-efficacy on Mathematics Achievement of Primary School Children. *Pakistan Journal of Psychological Research, 21*(3), 61-78.

Bain, C. D., & Rice, M. L. (2006). The Influence of Gender on Attitudes, Perceptions, and Uses of Technology. *Journal of Research on Technology in Education, 39*(2), 119-132.

Busch, T. (1995). Gender Differences in Self-efficacy and Attitudes toward Computers. *Journal of Educational Computing Research, 12*(2), 147-158.

Çakır, N. A., Gass, A., Foster, A., & Lee, F. J. (2017). Development of a Game-design Workshop to Promote Young Girls' Interest towards Computing through Identity Exploration. *Computers & Education*, *108*, 115-130.

Denner, J. (2011). What Predicts Middle School Girls' Interest in Computing?. *International Journal of Gender, Science and Technology, 3*(1), 54-69.

Fan, T. S., & Li, Y. C. (2005). Gender Issues and Computers: College Computer Science Education in Taiwan. *Computers & Education, 44*(3), 285-300.

Fernaeus, Y., Kindborg, M., & Scholz, R. (2006). Programming and Tools: Rethinking Children's Programming with Contextual Signs. *Proceedings of Conference on Interaction Design and Children IDC '06*. New York: ACM Press, 121-128.

Google (2015). Exploring Computational Thinking. Retrieved January 14, 2019, from https://www.google.com/edu/resources/programs/exploring-computational-thinking/

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

Jonassen, D., & Grabowski, B. (1993). *Handbook of Individual Differences, Learning, and Instruction*. Hillsdale, NJ: Erlbaum.

Kolb, D. A. (1976). *Learning Style Inventory Technical Manual*. Boston, MA: McBer.

Kolb A. Y., & Kolb D. A. (2005). *The Kolb Learning Style Inventory 3.1: Technical Specifications*. Boston, MA: Hay Resources Direct.

Rubio, M. A., Romero-Zaliz, R., Mañoso, C., & Angel, P. (2015). Closing the Gender Gap in an Introductory Programming Course. *Computers & Education, 82*, 409-420.

Severiens, S., & Geert, T.D. (1997). Gender and Gender Identity Differences in Learning Styles. *Educational Psychology*, *17*(1&2), 79-93.

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

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

Wing, J. M. (2011). Research Notebook: Computational Thinking - What and Why?. *The Link Magazine*, 20-23.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Supporting Representational Flexibility in Computational Thinking: Transitions between Reactive Rule-based and Block-based Programming

H. Ulrich HOPPE[1*], Sven MANSKE[2], Sören WERNEBURG[3]
[123] COLLIDE Research Group, University of Duisburg-Essen, Germany
hoppe@collide.info, manske@collide.info

ABSTRACT
Although the general principles of Computational Thinking are manifold and not bound to one specific programming paradigm current practice is less varied, dominated by visual block-based approaches following the imperative paradigm. For this study, an environment with an agent in a maze has been made accessible to programming in two different modalities: visual block-based programming (the current standard) and reactive rule-based programming. The latter approach allows for defining the agent behavior in bottom-up style in reaction to current local conditions. We have tested the transfer between the two approaches in both directions, i.e. starting with reactive rule-based programming followed by visual block-based programming and vice versa. It turns out starting with the reactive rule-based approach is superior in terms of achievement (level gain) and problem understanding.

KEYWORDS
representational flexibility, algorithmic thinking, reactive rule-based programming, block-based programming

1. INTRODUCTION
According to Hoppe and Werneburg (2019), the "essence of Computational Thinking (CT) lies in the creation of 'logical artifacts' that externalize and reify human ideas in a form that can be interpreted and 'run' on computers". These logical artifacts are often the results of programming activities, which links computational thinking to programming as a medium. Although the term "Computational Thinking" has gained popularity more recently, especially through Wing (2006), already Papert (1996) described the idea and used the term in conjunction with the development of the LOGO language as a medium for learning mathematics. In her recent characterization of CT, Wing (2017) emphasizes the importance of abstraction: "The most important and high-level thought process in computational thinking is the abstraction process. Abstraction is used in defining patterns, generalizing from specific instances, and parameterization".

The executable artefacts that are created in CT activities are examples of "computational models". Such models can be generated by learners from scratch, they can be modified, or they can be used for experimentation as is often the case with interactive simulations supporting scientific inquiry learning. Systematically using or modifying simulations can also involve CT skills (cf. Sengupta et al., 2013). However, in such environments the basic computational "ingredients", namely underlying data structures and a basic processing model, are usually predefined and fixed. However, from a computer science point of view it is desirable that learners should be able to create computational models with a certain freedom of choice regarding the use of different representations (e.g. data structures) and processing mechanisms. Even beyond simple variation in formulating a specific programming solution, we would like to enable learners to actively experience different computational approaches and paradigms. Aho (2012) uses the term "models of computation" to address this aspect. Variations on this level are found between different classes of programming languages (e.g., imperative versus declarative languages) but also comprise "abstract machines" such as automata or grammars. We use the term "representational flexibility" to denote the characteristic of a CT environment that supports different models of computation.

The study reported on in this paper combines two different computational approaches and investigates sequencing and transfer effects between two different ensuing experiences. This is enabled through the provision of different computational approaches that are applied to the same problem, namely steering a programmable agent to escape from a maze. In this context, successful problem solving requires understanding and skills on two levels: (1) programming and (2) maze strategies. Related to (2), we would speak of "learning through programming" as compared to "learning to program" (1). The co-existence of these two orientations is frequently found in programming-based microworlds (cf. DiSessa, 2000). The maze problem domain is closely related to turtle geometry. Labyrinth algorithms are discussed from this perspective by Abelson and DiSessa (1981).

The aim of our study is to investigate the influence of "representational variation" in terms of multiple models of computation on both problem understanding and the development of programming skills.

2. THE ENVIRONMENT: TMAZESTUDIO
The *ctMazeStudio* system facilitates the definition of agent behavior in a maze environment with different difficulty levels. At all levels, the goal is to define a strategy that lets the agent find a way out of any maze of the given level. In the overall learning process, the learners will formulate strategies of more and more general nature, ending up with a correct implementation of "wall following" (i.e. navigating through maze keeping the walls always on one hand side).

ctMazeStudio offers two different computational approaches to formulate agent strategies as solutions to the given problems:

The reactive rule-based approach facilitates the formulation of strategies in a bottom-up and "situated" fashion: In a given situation (i.e. with the agent in a certain position in a certain maze), the learner is provided with a localized rule that reflects the concrete situation in the neighborhood of the agent in its pre-instantiated conditions (IF-part) whereas the action part (THEN-part) of the rule is still empty. Now the learner has to fill in a corresponding action or action sequence made up of 90° turns ("Left" / "Right") or stepwise movements forward. Figure 1 shows an example of such a situated rule construction: The agent (called "Hero") facing towards the right has walls to the left and right ("blocked") and a free space in the viewing direction ("front free"). Here, the choice is clearly "Go Forward".

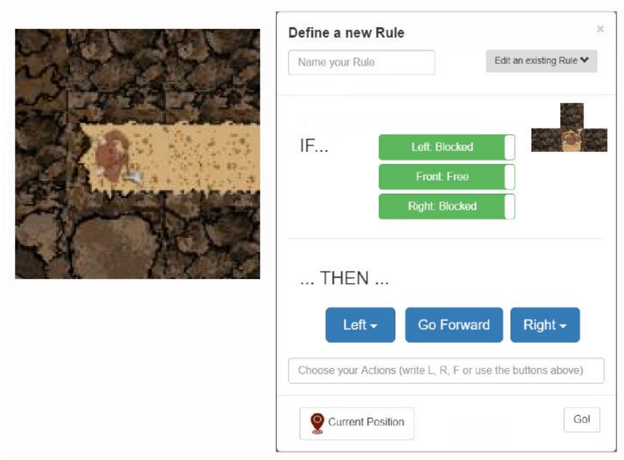


Figure 1. ctMazeStudio's "situated" Rule Editor.

The rule editor shown above is always invoked when a new situation is encountered. For the given conditions, the learner selects the desired actions and thus defines a situated rule of "reactive" behavior (triggered by the given conditions). The user can also delete conditions, which implies that the corresponding rule will be applied in situations more general than the given one, disregarding one of the premises (as a generalization mechanism). The rules will be "memorized" by the agent and will be re-applied under the same conditions. This approach was inspired by the kind of visual agent programming introduced in "KidSim" (Smith, Cypher, & Spohrer, 1994).

In addition to the rule editor, *ctMazeStudio* contains two more components: the behavior stage and a rule library (Figure 2).

Figure 2. ctMazeStudio with Rule Library.

The architecture of the rule-based variant of *ctMazeStudio* is shown in Figure 3. In the graphical user interface, the user can create a new rule or modify an existing rule in the rule editor. Each created rule is listed in the rule library, initially in the order of creation. The ordering of the rules determines the order of the matching and ensuing execution. The rule manager combines the rule library with the interpretation of rules and renders the result in the behavior stage using a game engine.

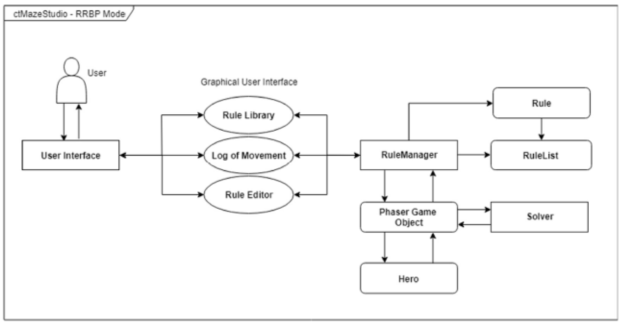


Figure 3. Architecture of the rule-based system.

The rule library (shown in Figure 2) allows for managing the collection of all previously defined rules. The learners can edit or delete already defined rules, directly enter new rules and change the order (and thus priority) of the rules to be checked. Depending on the entries in the rule library, the corresponding actions are executed and the specific entry in the rule library is highlighted. The execution will stop if now applicable is found or the goal is reached.

On the higher levels, learners have to apply different strategies to improve their programming code (i.e. the rule set). When they investigate and test their rule sets in consecutive situations, they may revise formerly defined rule sets through generalization (dropping of conditions) or reordering. The challenge is to create a maximally powerful rule set with a minimum number of rules. This requires a level of understanding that allows for predicting global behavior based on the locally specified rules. In the maze example, a small set of rules (minimally three) will be created to implement a wall-following strategy. A correct algorithmic solution has to ensure that the wall is always kept either on the right or on the left hand. This strategy works with any kind of maze that has no cycles or "islands".

Figure 4. Block-structured programming interface.

In addition to the reactive rule-based mode, the *ctMaze-Studio* environment can also be programmed through a block-structured interface (Figure 4), similar to Scratch (Resnick et al., 2009). This corresponds to top-down imperative programming approach with conditionals and loops as control structures. In combination of both approaches, *ctMazeStudio* affords a specific form of "representational flexibility" the gives the learner two different programming interfaces for the same task.

3. HYPOTHESES AND STUDY DESIGN

Providing the learners with choices between different computational approaches and representations when teaching CT is a postulate that resonates with conveying the power and richness of computer science constructs to the learners. This is very much what Aho (2012) advocates when he introduces the notion of "models of computation". CT learning environments based on one specific computational approach will particularly support a learning progression within this approach. In *ctMazeStudio* we can examine the impact of and the interaction between different computational representations and approaches. The target is, in first place, the development of problem understanding in the given task domain conditioned by the one or the other computational approach. The computational approaches provide different versions of agent programming, whereas the task domain is the same (namely labyrinth algorithms leading to "discovering" the wall-following strategy).

Based on these premises, we have studied the effect of sequencing the usage of reactive rule based programming approach (RRBP) and of visual block-based programming (VBBP). Our central hypothesis was:

(H1) The understanding and active mastery of wall-following will be better supported by RRBP.

Our two experimental conditions were RRBP first, followed by VBBP second for group A and vice versa for group B. Following H1, we would expect the learning gain (related to the maze strategy) to be higher for group A than for group B after the first trial. We would expect group B to "catch up" after the second round. Additional observations were made regarding the problem-specific and general coding abilities in the VBBP approach. Specifically, we would expect:

(H2) Prior experience with RRBP will lead to better solutions in the VBBP modality in terms of finding and implementing correct strategies.

Figure 5 represents the overall experimental procedure:

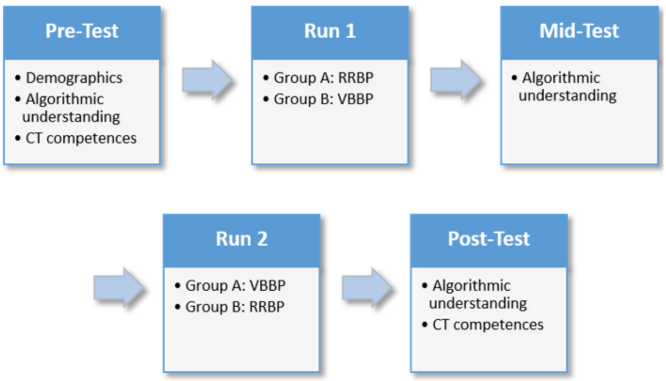


Figure 5. Experimental procedure.

The tests of algorithmic understanding were related to the maze problem and operationalized through specific questions, involving paper and pencil solutions with given labyrinths. CT competencies were tested through questions inspired by Gonzalez (2015) and Grover and Basu (2017).

The study was conducted in a public German high school ("Gymnasium") with a group of 31 students of grade nine participating in a computer science course (elective), 2 were female and 29 male, and all between 14 and 16 years old (M = 14.87). The average self-assessment of programming skills was 2.77 on a 5-point Likert scale. Group had 15, group B 16 participants. The duration of the test was 90 minutes.

4. RESULTS

Table 1 captures the distribution of successful completions of level 8 (corresponding to wall following) for all groups and conditions. For both groups, the rate of success increased from trial 1 to trial 2 (A: 6 to 8; B: 2 to 5). The overall success was higher in group A.

Table 1. Success (completion of Level 8) per group and programming modality.

	RRBP	VBBP
Group A	6	8
Group B	5	2

The outcome for group A indicates that a better problem understanding was developed in the RRBP condition and, if this was the first experience, it could be transferred to the second phase allowing for a re-coding in the other modality (VBBP). In contrast, starting with VBBP did not facilitate initial understanding and success in solving the problem.

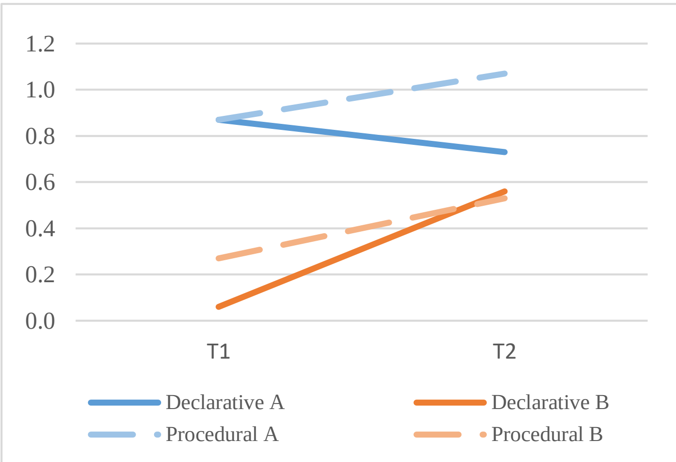


Figure 6. Algorithmic understanding: declarative and procedural knowledge of groups A and B, measured at times $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$.

Figure 6 shows the quantified results of the "algorithmic understanding" test applied after Run 1 (T1) and after Run 2 (T2). The questions were designed in such a way as to distinguish procedural and declarative knowledge related to this problem, and the diagram shows the results with this distinction. First, we compared the different measurements for each of the groups (A and B) separately using the non-parametric Wilcoxon signed-rank test. For both groups, the difference (in both cases an increase) in procedural knowledge was not significant. However, group B showed a significant increase in declarative knowledge between T1 and T2 ($Z=19.5$, $p=0.026$). The corresponding difference (slight decrease) of declarative knowledge in group A was not significant.

Secondly, we used the Mann-Whitney U test to compare the declarative understanding between groups A and B. We found a significant difference for the measurements at time point T1 ($U=68.5$, $p=0.20$), but no significant difference at T2 ($U=105.5$, $p=0.286$).

This corroborates our central hypothesis (H1): The RRBP experience is essential for a better (declarative) under-standing of the maze strategy in both groups. The essential knowledge gain comes from the exposition to RRBP.

The second hypotheses (H2) is plausibly backed by the comparison of success figures in Table 1: Success in the VBBP condition is four times higher if this modality is preceded by RRBP. However, an analysis of "productivity" in terms of number of trials did not show a difference between groups A and B in the VBBP condition.

Evidently, the empirical basis for our findings is limited. The quantitative dominance of male participants in our experimental group introduces a gender imbalance but it is typical for elective (choice-based) computer science courses in German high schools and was inevitable in a study conducted in the field.

In spite of the overall positive result for RRBP, there was a specific issue that often created a learning obstacle: The Rule Editor allowed for entering an unlimited sequence of actions so that a specific solution for the given maze could be specified at a single blow. However, such solutions would not be transferable to other mazes (not even of the same level). To avoid this problem, the number of actions in one rule can be limited in the current version of *ctMazeStudio*. The maximally necessary number of actions would be two, which allows for combining one forward step with a turn.

5. DISCUSSION

This study aimed at investigating the differences in CT "induced" by different computational approaches or para-digms used in then maze problem solving task. The differences were reflected and measured in terms of the understanding of the problem-related strategies as an effect of "learning through programming".

The RRBP approach favors a bottom-up and "situated" type of reasoning and is certainly closer to the problem than VBBP. Accordingly, it provides an easier start. However, RRBP comes with the challenge of inferring the global behavior of the agent from a collection of such locally defined rules. A correct implementation of the wall-following strategy requires the generalization of rules and typically also a reduction of the accumulated rule set, which requires more than a local understanding of the individual rules. We could demonstrate that the prior experience with RRBP supports the declarative understanding of the problem better than VBBP and also leads to higher success rates in the following VBBP condition, which in turn is characterized by a top-down and imperative model of computation. In this sense, RRBP is able to "feed into" VBBP in terms of a transfer of learning.

Our findings suggest that sticking to one computational approach alone may not be adequate. Different models of computation do not only increment the learners' knowledge base by juxtaposition, they can also positively interact with each other. Accordingly, we should further explore "representational flexibility" in teaching CT.

6. ACKNOWLEDGMENTS

We dedicate this publication to the memory of Sören Werneburg who designed and developed the *ctMazeStudio* environment and planned this study. We thank our former student Anna Taphorn for orchestrating and conducting the empirical investigation and the statistical analysis.

7. REFERENCES

Abelson, H., & DiSessa, A. (1981). *Turtle Geometry*. Cambridge (USA): MIT Press.

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

DiSessa, A. A. (2000). *Changing Minds: Computers, Learning, and Literacy*. Cambridge (USA): MIT Press.

González, M. R. (2015). Computational Thinking Test: Design Guidelines and Content Validation. *Proceedings of the EDULEARN15 Conference*. IATED Academy, 2436-2444.

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 SIGCSE Technical Symposium on Computer Science Education*. New York: ACM, 267-272.

Hoppe, H. U., & Werneburg, S. (2019). Computational Thinking - More than a Variant of Scientific Inquiry! In Kong, S. C. & Abelson, H. (eds.), *Computational Thinking Education*. Singapore: Springer.

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

Papert, S. (1996). An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

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.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating Computational Thinking with K-12 Science Education using Agent-based Computation: A Theoretical Framework. *Education and Information Technologies*, *18*(2), 351-380.

Smith, D. C., Cypher, A. & Spohrer, J. (1994). KidSim: Programming Agents without a Programming Language. *Communications of the ACM*, *37*(7), 54-67.

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

Wing, J.M. (2017). Computational Thinking's Influence on Research and Education for All. *Italian Journal of Educational Technology*, *25*(2), 7-14.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Study on the Current Situation of Visual Programming for Primary School Students and Its Influencing Factors

Min ZHANG[1], Yi ZHANG[2*], Huan-huan LIU[3], Wei MO [4], Dan-dan WANG[5]
[12345] School of Educational Technology, Central China Normal University, China
[4] School of Education Science, Hunan Institute of Science and Technology, China
1187516892@qq.com, zhangyi@mail.ccnu.edu.cn, 1329891230@qq.com, mowei0201@gmail.com, 1325857783@qq.com

ABSTRACT
In order to investigate the current situation and influencing factors of visual programming learning of primary school students, this study took the sixth grade students who have just learned the Scratch course for half a year as the research object, and studied their learning interest, future career intention, motivation, cognitive load and programming self-efficacy. The findings are as follows: (1) students have strong interest in computer programming, high learning motivation, good self-efficacy in programming and low cognitive load; however, students are less willing to engage in computer programming relates occupations in the future. (2) there are significant differences between male and female students in future career intention and programming self-efficacy. (3) there is a significant positive correlation between learning interest, future career intention, motivation and programming self-efficacy, and a significant negative correlation between cognitive load and learning interest.

KEYWORDS
visual programming, learning interest, motivation, programming self-efficacy

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

小学生可视化编程学习现状及其影响因素研究

张敏 [1]，张屹 [2*]，刘欢欢 [3]，莫尉 [4]，王丹丹 [5]
[12345] 华中师范大学教育信息技术学院，中国
[4] 湖南理工学院教育科学学院，中国
1187516892@qq.com，zhangyi@mail.ccnu.edu.cn，1329891230@qq.com，mowei0201@gmail.com，1325857783@qq.com

摘要

为了调查小学生可视化编程学习的现状及其影响因素，本研究以刚接触 Scratch 编程课程半年的六年级学生为调查对象，对其编程学习兴趣、未来职业意愿、学习动机、认知负荷以及编程自我效能进行研究。研究发现：（1）学生对计算机编程有浓厚的学习兴趣，学习动机较高，编程自我效能感较好，认知负荷较低，但是学生对于未来从事计算机编程相关职业的意愿较低；（2）男女生在未来职业意愿、编程自我效能感方面存在显著差异；（3）学习兴趣、未来职业意愿、学习动机和编程自我效能感间呈显著的正相关关系，认知负荷和学习兴趣间呈显著的负相关关系。

关键字

可视化编程；学习兴趣；学习动机；编程效能感

1. 前言

计算思维是当前教育领域研究的热门课题。关于计算思维的定义众说纷纭，但目前大部分学者较为认可的就是卡梅隆大学周以真教授 Wing（2006）对计算思维的理解：计算思维是指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。随着大数据、人工智能等不断发展，计算思维被认为是必备的关键素养之一。目前，计算思维的培养已经不仅仅局限在高等教育阶段，也在不断的向 K-12 教育阶段和学前教育阶段扩大。国内外对于 K-12 阶段计算思维的培养十分重视，很多国家都将计算思维纳入到在其人才培养计划和课程体系中（陈鹏，2018）。例如：美国 2011 年将计算思维纳入到《CSTA K-12 标准（2011）修订版》；英国的"2013 年新课程计划"、澳大利亚 2015 年的"新课程方案"也都将计算思维纳入到其新信息技术课程中。国内 2017 新版《普通高中信息技术课程标准中》中将计算思维作为信息技术学科核心素养之一。利耶等（Lye & Koh, 2014）通过对 27 篇利用程序设计来培养计算思维的实证研究综述后得出：在幼儿、中小学、大学中都可以采用程序设计开发的方式培养计算思维。目前在 K-12 阶段采用的编程工具为可视化编程工具，如 Scratch、Alice 等。特别是 Scratch 凭借其图形化、操作简单、资源广阔、程序功能全面等特点迅速在国内外掀起一股 Scratch 热潮，迅速走入学生信息技术课堂 Scratch 也在世界各国被广泛作用于编程启蒙语言。赛斯-洛佩斯等（Sáez-López et al., 2016）邀请 107 名来自不同小学的 6 年级学生利用 Scratch 在课堂中创作多媒体作品，结果表明：学生的计算实践能力得到了显著的提升，对计算概念的理解不断深入。张立国（2018）通过分析 2015 年及 2016 年国际计算机学会提交的论文，得出可视化技术能够提升学习者计算思维不仅具有理论可行性，还具有实践可行性。计算思维不是一个独立存在的概念，它与算法思维、编程思维、数学思维和工程思维等概念交叉在一起（谢忠新和曹杨璐，2015）。当前，在小学阶段，利用可视化编程工具编写程序是培养学生算法思维和编程思维的主要方式。基于此，本研究聚焦于小学生可视化编程教育的学习现状和影响因素研究，为在中小学开展可视化编程课程提供借鉴意义。

2. 文献综述

可视化编程操作简单，结果直接可视。可视化就是把数据、信息和知识等抽象内容，以直观可示的方式表达出来，并对数据进行更深层次的认识和解读（李芒，2013）。编程就是计算机为解决某个问题而使用某种程序设计语言所编写的代码程序（郁晓华，2017）。目前常见的可视化编程工具有 Scratch、App Inventor、Alice 等。可视化编程的直观性和趣味性能够提升学生的学习兴趣，能够降低代码语句编写等基础性技术门槛，可有效聚焦于计算思维的培养和能力的发展（郁晓华，2017）。另一项研究发现，五年级和六年级的学生认为 Scratch 有用性高，对 Scratch 态度积极，能够激发学习动机（Maloney, 2008; Maloney, 2010; Sáez-López et al., 2016）。可视化编程（例如 Scratch, App Inventor）确实维持了学生的学习动机和兴趣（Nikou, 2014）。虽然计算思维不等同于编程，但编程教育是培养学生计算思维的有效途径之一，而 Scratch 等可视化编程工具的出现则为编程走入小学信息技术课程提供了落脚点。

个人效能会影响个人对活动的选择、投入的努力、面对障碍时的坚持和表现（Bandura, 1977; Schunk, 1989）。自我效能感高的人更愿意投入精力来应对具有挑战性的任务（Bandura, 1994）。有研究发现，在计算机科学中使用 Scratch 可以提高学生的认知水平和自我效能感，但并不会导致学生产生高度的学习焦虑，学生花在学习和创建新程序上的时间更少（Armoni, 2015）。编程自我效能感反映了一个人对自己运用编程知识和技能解决计算问题的能力的感知和判断，高编程自我效能感的学习者更愿意运用自己的知识和技能来解决计算问题（Kong, 2017）。

综上所述，可视化编程是培养学生计算思维的良好工具，可视化编程能够提升学生学习兴趣、激发学生学习动机、提升学生自我效能感，减少学习焦虑。

3. 研究方法

3.1. 研究对象

本研究以武汉市某小学刚刚接触计算机编程教育半年的六年级学生为调查对象，该小学自三年级开始开设信息技术课程，但课程的内容集中于画图、Word、Excel、PowerPoint 等软件的应用。由于六年级学生对计算机操作较为熟练，因此选择六年级学生开设 Scratch 课程。本研究意在了解通过半年的 Scratch 课程学习后，学生编程学习兴趣、未来职业意愿、编程学习动机、认知负荷和编程自我效能感的现在，为后续 Scratch 课程内容的设置、教学方法等提供借鉴意义。本研究采用调查研究法，发放问卷 101 份，回收有效问卷 72 份，问卷回收率为 71.3% 。其中男生 35 人，女生 37 人。

3.2. 研究工具

本研究调查问卷由学习兴趣、未来职业意愿、学习动机、认知负荷和编程自我效能感五部分组成，共 16 个题项。其中学习兴趣由"我认为学习计算机编程很有意思"和"我十分享受编程的过程"两道题目组成；未来职业意愿则主要是关注学生未来从事相关职业的意愿——"我未来想从事与计算机编程相关的职业"；学习动机主要参照丁永祥（2012）编制的小学生学习动机调查表，"在编程活动中得到好成绩，对我来说是最满足的事情"、"在编程活动中，我比较喜欢有挑战性的内容，因为这样我可以学到新的事物"、"我想要在编程课程中学到东西，分数不高也无所谓"等 6 道题目；在认知负荷部分，本研究改编自 Hwang（2013）提出的认知负荷调查问卷，包含 "编程课程的学习内容对我来说很难"、"我不得不花费很多的精力来解决编程学习活动中的问题"2 道题目"；编程自我效能感则采用 Kong（2018）一文中对编程自我效能感的测量题项，共 5 道题。

问卷采用李克特 5 级量表，通过 SPSS21.0 对问卷信度进行分析，由于未来职业意愿题项较少，因此没有单独分析其信度。学习兴趣克朗巴赫 α 系数为 0.905，学习动机部分克朗巴赫 α 系数为 0.679，认知负荷部分克朗巴赫 α 系数为 0.773。克朗巴赫 α 系数在 0.6-0.7 还可以接受；编程自我效能感部分克朗巴赫 α 系数为 0.846。问卷总体克朗巴赫 α 系数为 0.832，整体问卷具有较好的信度。

表 1 问卷信度

维度	题项数	克朗巴赫 α 系数
学习兴趣	2	0.905
未来职业意愿	1	/
学习动机	6	0.679
认知负荷	2	0.698
编程自我效能感	5	0.846

4. 数据分析

4.1. 总体现状分析

总体来看，学生对计算机编程有浓厚的学习兴趣、学习动机高、学生对于编程的自我效能感较好，认知负荷较低，但是学生对于未来从事计算机编程相关职业的意愿较低。由表 2 可知，编程初学者对编程保有浓厚的学习兴趣，有较强的学习动机。学习兴趣和学习动机均值分别为 3.89 和 3.54，这可能与教师采用游戏化的教学方式和编程工具本身特性有关。在本研究中所有编程初学者均是由同一位教师采用游戏教学法进行编程教学，教师通过将计算机科学概念与知识融合到小游戏中，学生通过制作游戏完成对相关知识与技能的习得。本研究中编程初学者是通过 Scratch 编程工具进行编程启蒙的，可视化编程的直观性和趣味性能够提升学生的学习兴趣（郁晓华，2017）。此外，由于学生没有任何编程经验，教师在课堂上所教授的内容难度较低，一般选用代码数目较少、难度较低、趣味性较强的游戏进行案例教学。这在一定程度上降低了学生的认知负荷，提高了学生的编程效能感。虽然学生对编程学习有浓厚的学习兴趣和较高的自我效能感，但是其未来职业意愿偏低，这可能与六年级的学生尚缺乏对未来职业的规划以及对计算机科学领域的了解有关。

表 2 编程学习效果

维度	均值	标准差
学习兴趣	3.89	0.95
未来职业意愿	2.54	1.26
学习动机	3.54	0.74
认知负荷	2.15	0.95
编程自我效能	3.55	0.89

4.2. 性别差异分析

为了解性别对编程学习效果的影响，本研究采用独立样本 t 检验对不同性别的编程初学者在学习兴趣、学习动机、未来职业志愿、认知负荷和编程自我效能感是否存在差异进行了分析，

表 3 男女生学习效果差异比较

学习效果	性别	均值	标准差	t 值
学习兴趣	男	3.94	0.95	0.47
	女	3.84	0.97	
未来职业意愿	男	2.89	1.18	2.33*
	女	2.22	1.25	
学习动机	男	3.68	0.79	1.63
	女	3.40	0.66	
认知负荷	男	2.20	1.11	0.40
	女	2.11	0.78	
编程自我效能感	男	3.79	0.80	2.27*
	女	3.32	0.93	

注：$^{*}p<0.05$, $^{**}p<0.01$, $^{***}p<0.001$

结果如表 3 所示。男生在学习兴趣、未来职业意愿、学习动机、认知负荷和编程自我效能感的得分均值均高于女生。其中，男生的未来职业意愿（M=2.89，SD=1.18）显著高于女生的未来职业意愿（M=2.22，SD=1.25），t（70）=2.33,p=0.023<0.05。男生的编程自我效能感（M=3.79，SD=0.80）显著高于女生的编程自我效能感（M=3.32，SD=0.93），t（70）=2.27,p=0.026<0.05。在学习兴趣、学习动机和认知负荷三个维度男生得分均高于女生，但是不存在显著性差异。

通过分析发现，在"我认为自己是一个擅长编程的人"题项中，男生得分（M=3.74，SD=1.07）显著高于女生得分（M=2.97，SD=1.36），t（70）=2.66,p=0.01<0.05。在"我有编程的技能"题项中，男生得分（M=3.88，SD=0.99）显著高于女生得分（M=3.05，SD=1.24），t（70）=3.12,p=0.003<0.01。具体情况如表 4 所示。

表 4 男女生编程自我效能感差异比较

编程自我效能感题项	性别	均值	标准差	t 值
我认为自己是一个擅长编程的人	男	3.74	1.07	2.66*
	女	2.97	1.36	
我有编程的技能	男	3.88	0.99	3.12**
	女	3.05	1.24	

注：*p<0.05,**p<0.01,***p<0.001

4.3. 相关分析

通过对学生学习兴趣、未来职业意愿、学习动机、认知负荷和编程自我效能感进行相关分析。

表 5 相关分析

皮尔逊相关	学习兴趣	未来职业意愿	学习动机	认知负荷	编程自我效能感
学习兴趣	1	0.329**	0.643**	−0.323**	0.578**
未来职业意愿	0.329**	1	0.425**	0.03	0.311**
学习动机	0.643**	0.425**	1	−0.087	0.616**
认知负荷	−0.323**	0.03	−0.087	1	−0.174
编程自我效能	0.578**	0.311**	0.616**	−0.174	1

如表 5 所示，未来职业意愿和学习兴趣、学习动机、编程自我效能感之间存在显著的正相关关系（Pearson correlation value=0.329**, p<0.01; Pearson correlation value=0.425**, p<0.01; Pearson correlation value=0.311**, p<0.01）；编程自我效能感与学习兴趣、未来职业意愿和学习动机存在显著的正相关关系（Pearson correlation value=0.578**, p<0.01; Pearson correlation value=0.311**, p<0.01; Pearson correlation value=0.616**, p<0.01）；认知负荷和学习兴趣呈现显著负相关（Pearson correlation value=-0.323**, p<0.01）。

5. 研究结论及建议

本研究采用问卷调查法对刚刚接触可视化编程教育半年的六年级学生的计算思维学习现状进行调研，分析了性别对学习兴趣、学习动机、未来职业意愿、认知负荷和编程自我效能感的影响，并对学习兴趣、未来职业意愿、学习动机和编程自我效能感进行相关分析。研究发现（1）学生对计算机编程兴趣浓厚、学习动机高、编程自我效能感较好，学生认知负荷较低，但是对未来从事计算机编程相关职业的意愿较低；（2）男生未来职业意愿、编程自我效能感高于女生且存在显著差异；Master 等（2017）的一项研究表明，男生和女生对计算机科学和工程学的思维定势在很小的时候就已经存在，这种思维定势会影响女生对这些领域的兴趣和自我效能。（3）学习兴趣、未来职业意愿、学习动机和编程自我效能感之间呈显著的正相关关系，认知负荷与学习兴趣之间呈显著的负相关关系。

在可视化编程课程中，如何激发学生学习兴趣，提高学生学习动机，增强学生编程的自我效能感，发挥可视化编程工具培养学生计算思维的优势至关重要。基于此，本文提出以下建议：

一、面向学生的编程学习活动应该以学生为中心，从学生的视角来组织课堂活动。对于刚刚接触编程的初学者来说，编程是一项具有挑战性的活动，如果在一开始不能引起学习者的学习兴趣，那么在以后的学习中，学习者很容易对其产生抵触情绪。目前，中小学课堂中利用趣味游戏作为知识载体，学生学习兴趣和学习动机在初始活动中能够维持在较高水平，但是教育者还应关注如何维持他们在学习活动中的学习兴趣和努力程度，激发学生的内在学习动机。此外，对于接触编程的初学者来说，教育者在教学过程中，要注意问题任务粒度的划分，确保在符合学生认知水平和实际问题解决能力的基础上不断提升复杂度，提升学生的编程自我效能感，获得一种成就感的体验。

二、针对性别差异在计算机科学领域的客观存在，教育者需要为女孩更多地提供编程体验机会、激发学习兴趣、提高课堂参与度。在课堂中，教师要更多关注到女孩的表现，给予及时的鼓励和反馈，增强她们的自信心和自我效能感。在本研究中男女生在未来职业意愿和自我编程效能感方面存在显著差异。自我效能感是职业意愿的重要预测因素，人们更愿意选择自身有竞争力的职业，并且回避自认为不擅长的职业（Lent, 1994）。因此，要改变性别差异在计算科学领域的存在，提升女生的编程自我效能感至关重要。学校编程社团男女比例失衡也是某些学校存在的问题之一，如何更好地发挥女生的主动性则变得至关重要。编程活动中，脚手架可以帮助女生的参与和信心，在一项涉及创造游戏的中学女生的研究中发现她们学习过程中展示了足够水平的复杂编程活动（Denner et al., 2012）。因此，设计适当的活动可以成为吸引和鼓励女生对计

算机科学兴趣的有前景的途径之一,并在活动中辅以适当的脚手架促进学生思维的表达。

当然，本研究还存在很多不足，在研究的样本量方面，本研究的样本量较少，仅关注了很少一部分学生。其次，本研究缺少纵向的比对，因此在以后的研究中关注不同学习年限的学生其学习兴趣、学习动机、未来职业意愿、编程自我效能感的变化，并探索影响小学生计算机科学相关职业意愿的因素。

6. 基金项目

国家自然科学基金2018年面上项目 促进小学生计算思维培养的跨学科STEM+C教学理论与实证研究 71874066；

华中科技大学附属小学合作项目“智慧教室中的教学创新促进小学生的深度学习”（合同编号：XM0120170101）

7. 参考文献

丁永祥（2012）。**小学生英语学习兴趣和动机调查与分析**。上海：华东师范大学。

陈鹏、黄荣怀、梁跃和张进宝（2018）。如何培养计算思维——基于2006-2016年研究文献及最新国际会议论文。**现代远程教育研究**，**1**，98-112。

李芒、蔡旻君、蒋科蔚和王妍莉（2013）。 可视化教学设计方法与应用。 **电化教育研究**，**34**（**3**），16-22。

谢忠新和曹杨璐. (2015). 中小学信息技术学科学生计算思维培养的策略与方法. **中国电化教育**，**11**，116-120。

郁晓华、肖敏、王美玲和陈妍（2017）。 基于可视化编程的计算思维培养模式研究——兼论信息技术课堂中计算思维的培养。 **远程教育杂志**， **35**（**6**）， 12-20。

张立国和王国华（2018）。计算思维：信息技术学科核心素养培养的核心议题。 **电化教育研究**， **39**（**5**），115-121。

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to “Real” Programming. *ACM Transactions on Computing Education (TOCE), 14*(4), 25.

Bandura, A. (1977). Self-efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological Review, 84*(2), 191-215.

Bandura, A. (1994). Self-efficacy. *In V. S. Ramachaudran (Ed.), Encyclopedia of Human Behavior*, 71-81. New York: Academic Press.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer Games Created by Middle School Girls: Can They be Used to Measure Understanding of Computer Science Concepts? *Computers & Education*, *58*(1), 240-249.

Hwang, G. J., Yang, L. H., & Wang, S. Y. (2013). A Concept Map-embedded Educational Computer Game for Improving Students’ Learning Performance in Natural Science Courses. *Computers & Education*, *69*(1), 121-130.

Kong, S. C., Sheldon, J., & Li, K. Y. (Eds.). (2017). *Proceedings of International Conference on Computational Thinking Education 2017.* Hong Kong: The Education University of Hong Kong.

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A Study of Primary School Students' Interest, Collaboration Attitude, and Programming Empowerment in Computational Thinking Education. *Computers & Education*, *127*, 178-189.

Lent, R. W., Brown, S. D., & Hackett, G. (1994). Toward a Unifying Social Cognitive Theory of Career and Academic Interest, Choice, and Performance. *Journal of Vocational Behavior, 45*(1), 79-122.

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.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by Choice: Urban Youth Learning Programming with Scratch. *ACM, 40*(1), 367-371.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16.

Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming Experience Promotes Higher Stem Motivation among First-grade Girls. *Journal of Experimental Child Psychology, 160*, 92-106.

Nikou, S. A., & Economides, A. A. (2014). Transition in Student Motivation during a Scratch and an App Inventor Course. *Proceedings of 2014 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 1042-1045.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual Programming Languages Integrated across the Curriculum in Elementary School: A Two Year Case Study using “Scratch” in Five Schools. *Computers & Education*, *97*, 129-141.

Schunk, D. H. (1989). Self-efficacy and Cognitive Achievement: Implications for Students with Learning Problems. *Journal of Learning Disabilities, 22*, 14-22.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Design and Development of Coding Poker Cards

Sheng-yi WU
Department of Science Communication, National Pingtung University, Taiwan
digschool@gmail.com

ABSTRACT

Since its inception, computational thinking has gradually been accepted by many countries as a critical element to the curriculum of elementary schools. Hence this study aimed to develop coding poker cards by the principles of game-based learning. That is, coding poker cards are used to increase learners' motivation and improve their cognitive skills. Meanwhile, this study employs augmented reality (AR) to verify these poker cards and exhibit real program behaviors. Lastly, this study hopes to develop a set of coding poker cards featured with low-cost, portable, and being able to provide real-time cooperation and face-to-face interaction using AR technologies.

KEYWORDS

computational thinking, coding, poker cards, augmented reality

1. MOTIVATION AND PURPOSE

The concept of computational thinking was presented by Prof. Wing at Carnegie Mellon University in 2006. As she pointed out, computational thinking can be employed to solve problems, design systems, and understand human behaviors. Some scholars have also stressed that computational thinking is a necessary skill that an individual need to acquire in modern times. By this token, it is very important for children to acquire computational thinking ability since childhood. An easy way to develop this ability is to learn by writing a program (Buitrago Flórez, Casallas, Hernández, Reyes, Restrepo, Danies, 2017). Most young learners who have zero experience in programming struggle with the text-based user interface (TUI) when they are flummoxed by its complexity (Costelloe, 2004; Powers, Ecott & Hirshfield, 2007). So far the benefits of "game-based learning" for young learners have been recognized by a considerable amount of studies.

With the rising popularity of educational board games, the computer science unplugged project (Bell, Alexander, Freeman, & Grimley, 2009) has also begun to gain attention, while related materials and ways of application have appeared on the market. To disseminate the concepts of program logic, this study presents a set of coding poker cards featured with low-cost and portable, which can also provide real-time cooperation and face-to-face interaction using AR technologies.

2. GAME DESIGN OF CODING POKER CARDS

The game mechanics of coding poker cards are mainly about card players trying to choose a route from a range of choices when they embark on a trip. By so doing, they need to use cards to discover the program logic, which indicates the principle of such routes. Meanwhile, card players try to consider how to use a minimum number of cards to get to the destination. Two to four players aged above eight can form a group for a game, which lasts 20-40 minutes. The program logic behind coding poker cards consists of sequence, event, repetition, parallel, naming, operator, and data application. The flowchart of the game is shown in Figure 1:

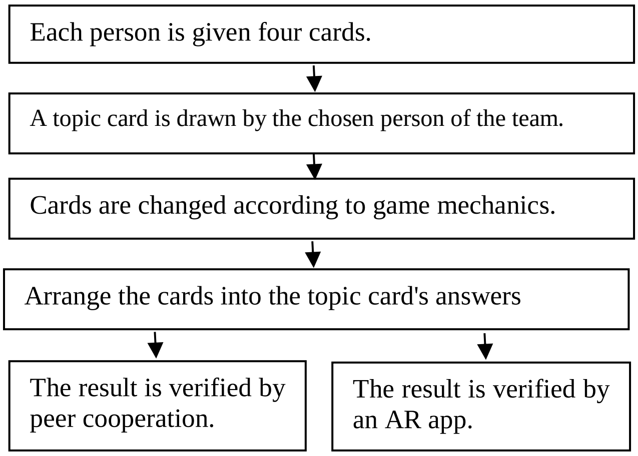


Figure 1. How to play coding poker cards.

The number of players should be decided at the very start. Each person is given four cards. Once all cards are distributed, a topic card is drawn by the person who has won the rock-paper-scissors hand game. The number of topic cards is between 2 to 4 (Figure 1). Two sides of the topic card are for the question and the answer, separately. Once it is drawn, the side with the question is up. Moreover, the maximum number of cards (with answers to them) is ten. Topics are contextually formulated.

Then card players can engage in "folding" or "drawing cards" according to some game mechanics. During the above said process, a person may draw a card at each turn, and the maximum is ten cards. When a card is drawn, the player may also take his new card through folding or changing cards. We also add some game mechanics to make this game even more enjoyable, such as getting more cards quickly, two cards being exchanged, and drawing cards from others' hands. It is up to the card player which topic card she/he will be choosing. Once the player has collected all the cards of which the problem-solving tasks have been completed, she/he needs to wait until the next turn to declare she/he has accomplished the mission. When it is accomplished, the card player is required to give explanations by order of the poker cards (Figure 3). Other teammates shall verify the result.

Figure 2. Example of topic cards.

Figure 3. Player collected all the cards according the tasks.

Card players may decide which topic card she/he is choosing from all the cards. As the person has collected all the cards of which the problem-solving tasks have been completed, she/he needs to wait until the next turn to declare she/he has accomplished the mission. When it is accomplished, the card player is required to give explanations by order of these poker cards. Other teammates shall verify the result. If a mobile phone is at hand, then player can be scanned via AR technologies for verification (Figures 3).

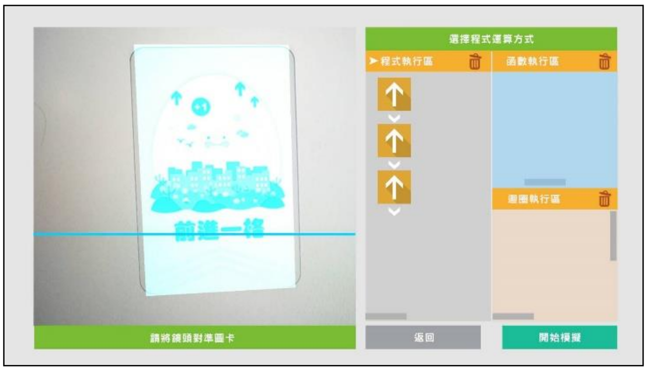


Figure 4. AR scan for verification.

3. CONCLUSION

The importance of computational thinking in education has been internationally accepted in recent years. Computational thinking may help children develop problem-solving skills and form a logical thinking model. Therefore, we create coding poker cards based on the principle of “game-based learning.” That is, coding poker cards are used to increase the learners’ motivation, facilitate “flow,” and then improve their cognitive skills. Meanwhile, AR is used to verify these poker cards and exhibit real program behaviors. Lastly, this study hopes to develop a set of coding poker cards featured with low-cost, portable, which can provide real-time cooperation and face-to-face interaction using AR technologies. We will explore learners’ performance after playing coding poker cards, using scales and behavioral models in an empirical study.

4. REFERENCES

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.

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a Generation’s Way of Thinking: Teaching Computational Thinking through Programming. *Review of Educational Research*, *87*(4), 834-860.

Costelloe, E. (2004). *Teaching Programming the State of the Art.* CRITE Technical Report. Retrieved January 2, 2019, from https://www.scss.tcd.ie/disciplines/information_systems/crite/crite_web/publications/sources/programmingv1.pdf

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*. New York: ACM, 213-217.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Promoting Computational Thinking Skills in the Context of Programming Club for K-12 Pupils with the Engaging Game Adventure in Minecraft

Jussi KOIVISTO[1], Jari LARU[2*], Kati MÄKITALO[3]
[1] Codeschool Finland, Oulu, Finland
[23] Faculty of Education, University of Oulu, Finland
jussi@codeschoolfinland.fi, jari.laru@oulu.fi, kati.makitalo@oulu.fi

ABSTRACT

The aim of this study is to explore how the game adventure in Minecraft promote computational thinking in the context of after-school K-12 programming club. Earth 2.0 map was designed to include problem-based puzzles and engaging post-apocalyptic game narrative (see tasks section). Earth 2.0 was tested in the authentic context with 60 pupils and five teachers. Preliminary findings provides support for using Minecraft as a tool for learning computational thinking concepts and practices.

KEYWORDS

minecraft, serious gaming, computational thinking, programming with games

1. INTRODUCTION

Wing (2006) defines computational thinking as a thought process involved in programming. Computational thinking is also considered an essential skill for 21th century students. Brennan & Resnick (2012) have presented three major dimensions of computational thinking (CT): 1) concepts: e.g. sequences and loops 2) practices: e.g. testing and debugging or reusing and remixing 3) perspectives: expressing or questioning. They illustrate their framework with practical examples with Scratch - a programming environment which engages users to creative problem solving activities by programming, like Earth 2.0 designed in our experiment.

However, based on literature review done by Lye & Koh (2016) most of studies (85% in their literature review) examined the learning outcomes in terms of computational concepts (e.g. variables or conditions) although computational thinking entails also practices and perspectives as suggested above. In this study, emphasis was put to concepts and practices. Our Minecraft map for learning CT, Earth 2.0 was designed to include two of the concepts suggested by Brennan & Resnick (2012)

2. AIM OF THE STUDY

The aim of this study is to explore how the game adventure in Minecraft can be used to promote computational thinking in the context of after-school K-12 programming club.

3. METHODS

The design rationales for Earth 2.0 Minecraft coding game were recurring difficulties on finding a functional and motivating way to teach programming in context of after-school programming club.

In order to solve this issue, pedagogically and theoretically grounded game experience was designed and evaluated in authentic context together with the code-club participants.

3.1. Participants

Participant groups enrolled on after-school programming club (five clubs, 8-20 participants (7-12 years old, 62 participants altogether) for three months with first four sessions on Minecraft. In the context of this study, all five club groups used the prototype version of the Earth 2.0 computational thinking game.

3.2. Tasks and Pedagogical Design

Earth 2.0 Minecraft world is based on storyline where robots are to be programmed for reconstructing the earth, where all activities are suddenly stopped. Players are scientist-astronauts who are ordered to go back on earth and check the situation. There is information about someone, who is trying to sabotage players' attempts to rehabilitate the planet earth.

This narrative is further divided into four main puzzles (quests) and one bonus puzzle (quest) which all are separate Minecraft worlds and one bonus world. First, player has to learn to use the programming tool, Beginners Turtle, in Tutorial.

Then they are introduced to basic programming concepts and practices in specific order (see the table 1). Concepts and practices of computational thinking (Brennan & Resnick 2012) are distributed to three sequential task-groups: Quests 1, 2 and 3. Finally, a fourth task-group, Bonus Quest, is included for quicker and more advanced players. See more detailed descriptions in the Table 1. Layout of an individual puzzle is introduced in Figures 1 and 2.

Table 1. Tasks to promote Computational thinking included in Earth 2.0.

Game phase	Task	Structures and statements used
Tutorial	Practice using the (Beginners) Turtle to open a door to Portal room (to advance to Quest 1)	Only movement commands
Quest 1, Puzzles 1-3	Use movement commands to move the Turtle and advance	Only movement commands
Quest 1, Puzzle 4	Use loop to move the Turtle long distances	While-true-do

Quest 2, Puzzle 1	Build a bridge	Repeat -structure
Quest 2, Puzzle 2	Remix bridge program to build stairs followed by a bridge	Repeat -structure
Quest 2, Puzzle 3	Use loop to move the Turtle long distances	While-true-do
Quest 2, Puzzle 4	Remix bridge program to repair broken bridge	Repeat -structure
Quest 3, Puzzle 1	Practice to use conditionals and loops together to dig through a cave	While-true-do, If-then-else
Quest 3, Puzzle 2	Program automated Turtle to avoid obstacles	While-true-do, If-then-else
Quest 3, Puzzle 3	Remix previous program to build a bridge while avoiding obstacles	While-true-do, If-then-else
Quest 3, Puzzle 4	Program automated Turtle to solve a labyrinth	While-true-do, If-then-else if
Quest 3, Puzzle 5	Reuse and debug previous program to solve bigger labyrinth	While-true-do, If-then-else if
Quest 3, Puzzle 6	Remix bridge, obstacle and digger programs to reach the exit	While-true-do, If-then-else if
Bonus Quest, Puzzle 1	Reuse bridge program and stair program to create path to the exit	Repeat -structure, While-true-do
Bonus Quest, Puzzle 2	Create intelligent bridge-builder -program to create path	While-true-do, If-then-else if

Figure 1. Quest 2, Puzzle 1. A1: Player encounters a problem. A2: Player programs the Turtle to build a bridge. A3: Player executes the program and Turtle builds a bridge for the player.

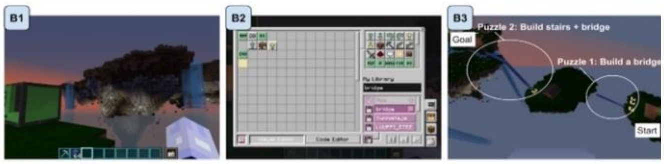


Figure 2. Quest 2, Puzzle 2. B1: Player encounters a slightly different problem. B2: Player remixes the bridge program to build stairs. B3: Overview of Puzzles 1-2 in Quest 2, both puzzles completed.

3.2 Data collection and analysis

To assess conceptual understanding of the computational thinking, the pupils completed identical online pre- and post-test surveys with a pre-/post-test quasi experimental design. Specifically, the conceptual knowledge measurement includes eleven questions that are developed based on the Ericson's and McKlin's (2012) Scratch test. Detailed procedures are described in the poster.

4. RESULTS

Paired samples t-test was conducted to compare pre- and post-test means. Results showed that participants gained higher scores in the post-test (M=11.67) than in pre-test (M=8.13). $t(60) = 6.71$, $p<.000$. All other test measures were significant, except comparison of the pre- and post-test means in the group of young pupils (7-9 years old), in which average gain between pre- and post-tests was only 0.04 and Wilcoxon signed rank test was not significant.

Basic commands in the Minecraft coding were familiar for all pupils (pre: n=49, post n=60), while in the questions concerning *conditionals* values were average when compared to all questions in the test and in the concept concerning sequences was quite low in all three questions. This can be explained also by play mechanics, because turtle could be moved also by using remote-controller, without programming.

More detailed results of pre- and post-tests will be presented in the poster at the conference. In addition to statistical tests, also correct responses between pre- and post-tests were compared.

5. REFERENCES

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, 1-25.

Ericson, B., & McKlin, T. (2012). Effective and Sustainable Computing Summer Camps. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*. ACM, 289-294.

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.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Investigating the Elementary School Students' Skills of Computational Thinking and Self-Efficacy through a Robot Programming Project

Chien-yuan SU[1*], Song HAN[2], Yue HU[3]
[123] Department of Curriculum and Learning Science, Zhejiang University, China
bredysu@gmail.com, 18645625763@163.com, hy_zju@126.com

ABSTRACT

This study attempts to integrate project-based learning (PBL) into a children's programming learning activity and to investigate its effort on improving the students' computational thinking and self-efficacy of programming. The whole activity is divided into five stages, which 1) to decide the problem, 2) to explore a possible solution, 3) to collect relevant information, 4) to try to solve the problem, and 5) to present the results. 41 fifth and sixth graders at Liyang Foreign Language School in Jiangsu participated in this study. These participants were assigned to an experimental group with PBL and a control group without PBL to proceed programming learning activity. Learning performance, including computational thinking tests and self-efficacy of programming, was measured.

KEYWORDS

programming education, project-based learning, computational thinking, self-efficacy.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

小學生專題式程式設計對運算思維和自我效能的影響

蘇建元 [1*]，韓嵩 [2]，胡玥 [3]
[123] 浙江大學課程與學習科學系，中國
bredysu@gmail.com，18645625763@163.com，hy_zju@126.com

摘要

本研究設計出"機器人大救援"專題式程式設計學習活動，來培養小學生的運算思維能力以及提高對程式設計的自我效能感。整個活動進行共分為五個階段，分別為：1）決定問題，2）探討可能的解決辦法，3）搜集相關資料，4）嘗試解決問題，及 5）呈現實作成果。為瞭解該活動設計效果，本研究以江蘇溧陽外國語學校的小學高年級共四十一位學生為對象，採實驗設計方式進一步去檢驗該專題式學習模式及傳統課堂教學模式下對小學生運算思維和自我效能感的影響。

關鍵字

程式設計教育；運算思維；專題式學習；自我效能感

1. 引言

近年來，全球各地興起了一股"程式設計學習"風潮，許多國家都開始倡導程式設計學習應從小扎根，並將其列為中小學階段的必修課程。一些過去的研究認為盡早讓學生去學習程式設計不僅能促進其邏輯思考、組織推理、判斷等能力（Grover & Pea, 2013；Kalelioğlu, 2015；Lye & Koh, 2014），也能提高學生自我效能感（self-efficacy）和運算思維能力（Swaid, 2015；Yukselturk & Altiok, 2017）。為了讓低齡的初學者能更容易的學習程式設計，一些圖形化程式設計學習工具如 Scratch、Kenrobot 或 Mixly 等被開發出來，讓學生能透過簡單積木式圖形操作界面去動手編輯程式，製作出許多豐富的數位作品。目前，在國內許多中小學的資訊課程時常可見到這類型的程式設計學習活動，但多數教學過程往往是學生仿效授課教師的操作，教師比較少放手讓學生實際去動手規劃、去發現問題與克服問題。一些教育學者強烈建議教學過程可採"專題式學習"（Project-Based Learning, PBL）的方式讓學生合作去進行相關的學習活動，實際解決真實情景中的問題並引導以及養成學生自主探究、解決問題等能力（Chan, 2008；Torp, 2002）。不過比較可惜的是過去所見的專題式程式設計學習成功案例多數是在大專院校的學生所進行（Rivera, Chotto, & Salazar, 2014），在國內中小學實際教學現場的應用與具體操作仍不多見，專題式策略如何去引導以及設計到目前小學生的程式設計學習活動，且對小學生的學習效果是否造成影響，這些都值得進一步去探討與瞭解。因此，本研究嘗試以專題式學習作為策略，去探討如何設計並應用到小學機器人程式設計學習活動中，並進一步探究這樣的學習活動設計對小學生運算思維和自我效能感的影響。

2. 文獻探討

2.1. 專題式學習

專題式學習是讓學生對專案主題進行深入探究的有效教學方法。有學者對其歸納出一些具體實施作法，如 Krajcik 和 Shin（2014）描述在科學課堂上專題式學習的六個特徵：1）引出問題，2）提出學習目標，3）科學實踐，4）合作活動，5）提供技術支援，6）公開彙報。林奇賢（2018）提出 PBL 課程設計流程並應用到小學數學課程之中：1）決定問題、2）探討可能的解決辦法、3）搜集相關資料，4）嘗試解決問題、5）呈現實作成果。

2.2. 運算思維

Wing（2006）首次提出運算思維概念以來，許多研究陸續去探討學生在真實情境中運算思維能力及規劃因應策略、作法來提升學生這方面的能力。美國國際教育技術協會（International Society for Technology in Education，ISTE）（2015）認為運算思維是由創造力（Creativity）、演算思維（Algorithmic thinking）、合作學習（Cooperativity）、批判性思維（Critical Thinking）、問題解決能力（Problem Solving）五個方面所構成。

2.3. 自我效能感

自我效能感（self-efficacy）被認為是"人們對自己組織並執行某項行動用以完成某項工作的可能性判斷"以及"個人執行必要行為以滿足特定工作要求的能力信念"。自我效能感不僅影響個人設定的目標水準、努力程度，也影響個人行動和表現（Bandura, 1977）。過去研究認為學習者的電腦自我效能會影響其在程式設計時認知、行為和動機的選擇和運用。若能增加學生的自我效能感，可以提高他們對程式設計的學習效果（Ozyurt, 2015）。

3. 研究方法

3.1. 研究對象和教具選擇

本研究以 41 位江蘇溧陽外國語小學的五~六年級生作為實驗對象，將學生以隨機分派方式分為實驗組 20 人（女生 2 人，男生 18 人），對照組 21 人（女生 1 人，男生 20 人），實驗組和對照組都均分為 4 組進行活動，每組 5~6 人。本研究選用 mBlock 及 mBot 做為學生軟硬體支援的程式設計編寫工具及硬體設備，並提供平版讓學生能實際進行操作演示。

3.2. 活動設計

本研究根據林奇賢（2018）提出的 PBL 五步驟設計出以"機器人大救援"為主題的小學生專題式程式設計學習活動，活動細節如下所示：

1）決定問題：授課教師先以圖片和視頻來營造出小學生需要進行的機器人救援活動，透過引導情境來讓學生思考問題，並利用現有的工具與設備來進行物資救

援。2）探討可能的解決辦法：學生以故事情節方式描繪救援可能遭遇的難題，並著手透過流程圖研擬可能的解決辦法，同時檢驗並模擬、測試可能的解決方案。3）搜集相關資料：學生搜索授課教師提供在線上學習平臺的學習資料，透過理解並推敲出可行的解決辦法。4）嘗試解決問題：學生根據收集的在線資料，嘗試動手實作，藉由操作、調整、除錯、修改等反覆檢驗並解決問題。5）呈現實作成果：各組合作製作出救援機器人雛形後，以小組競爭方式讓學生進一步去闡述設計原理和展示所設計的救援機器人。

3.3. 測量工具

本研究分別採用 Ozgen 等人（2017）編制的運算思維量表及 Volkan 等人（2017）編制用以檢測程式設計自我效能感量表來檢測小學生的運算思維能力及自我效能感評估。量表均採李克特氏 5 點量表設計，區分為"非常不同意"、"不同意"、"一般"、"同意"、"非常同意"。

3.4. 實驗設計

本研究的實驗設計規劃設定讓實驗組採專題式學習模式進行，而控制組則由教師主導的講授模式進行，兩組學生皆進行相同的活動主題，整個學習活動歷時 15 週，每週一堂課的時間進行，實驗流程如圖 1。

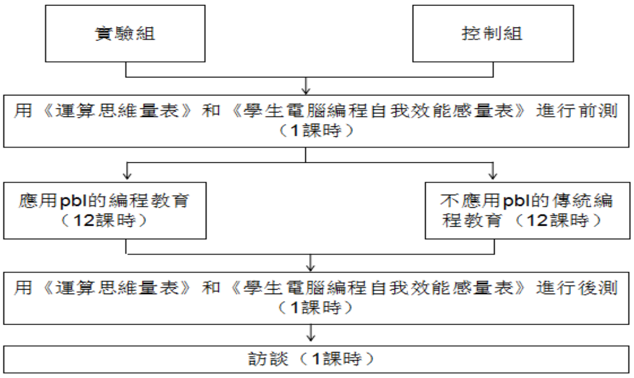


圖 1 實驗流程

3.5. 資料分析方法

本研究現階段尚在進行實驗數據收集，待資料收集完畢，預計採配對樣本 T 檢定和單因數共變數分析（one-way ANCOVA）方法來進行資料分析。

4. 結論

本研究設計出主題為"機器人大救援"的小學生專題式程式設計學習活動，學習者操作 Scratch 和 Arduino 去改造 mBot 機器人來完成一連串的救援任務，過程中讓小學生去動手改造 mBot 機器人，以情境式學習、小組專題合作的互動學習方式來促進學生自主探究，問題解決，並習得程式設計的基本知識。後續待實驗資料收集與分析後，進一步印證與瞭解所設定之小學專題式程式設計學習對小學生運算思維和自我效能感的影響，本研究所採取的作法亦能提供其他關於專題式學習或小學程式設計教學等的研究參考。

5. 致謝

感謝浙江省哲社重點項目的經費補助（No.18NDJC026Z）

6. 參考文獻

林奇賢（2018）。**新世代的創新學習模式互聯網+PBL 理論與實施策略**。臺北市：高等教育文化事業有限公司。

Bandura, A. (1977). Self-efficacy: Toward a Unifying Theory of Behavioral Change. *Psychological Review, 84*(2), 191.

Chan, L., & L. J. (2008). Technology Integration Applied to Project-based Learning in Science. *Innovations in Education and Teaching International, 45*(1), 55-65.

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

ISTE. (2015). *CT Leadership Toolkit*. Retrieved January 2, 2019, from http://www.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4.

Kalelioğlu, F. (2015). A New Way of Teaching Programming Skills to K-12 Students: Code. org. *Computers in Human Behavior, 52*, 200-210.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A Validity and Reliability Study of the Computational Thinking Scales (CTS). *Computers in Human Behavior, 72*, 558-569.

Kukul, V., Gökçearslan, Ş., & Günbatar, M. S. (2017). Computer Programming Self-efficacy Scale (CPSES) for Secondary School Students: Development, Validation and Reliability. *Educational Technology Theory and Practice, 7*(1) , 158-179. doi:10.17943/etku.288493.

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.

Ozyurt, O. (2015). An Analysis on Distance Education Computer Programming Students' Attitudes regarding Programming and their Self-efficacy for Programming. *Turkish Online Journal of Distance Education, 16*(2), 111-121.

Rivera, S. M., Chotto, M. C., & Salazar, G. A. (2014). A Proposal for Implementing PBL in Programming Courses. *In Proceedings of 2014 XL Latin American Computing Conference (CLEI)*. Montevideo, NY: IEEE, 1-11. doi: 10.1109/CLEI.2014.6965195

Swaid, S. I. (2015). Bringing Computational Thinking to STEM Education. *Procedia Manufacturing, 3*, 3657-3662.

Torp, L. (2002). *Problems as Possibilities: Problem-based Learning for K-16 Education.* Alexandria, VA: Association for Supervision and Curriculum Development.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35. doi: 10.1145/1118178.1118215

Yukselturk, E., & Altiok, S. (2017). An Investigation of the Effects of Programming with Scratch on the Preservice IT Teachers' Self-efficacy Perceptions and Attitudes towards Computer Programming. *British Journal of Educational Technology, 48*(3), 789-801

Computational Thinking and Unplugged Activities in K-12

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Effects of Plugged and Unplugged Advanced Strategy on Primary School Children's Outcomes in Scratch Learning

Wei-chi LIAO[1], Jung-chuan YEN[2*]
[12] Department of Mathematics and Information Education, National Taipei University of Education, Taiwan
heart1543@gmail.com, jcyen.ntue@gmail.com

ABSTRACT
The purpose of this study is to examine the effects of different advanced strategies of plugged and unplugged on the elementary students' programming learning motivation, learning achievement of Scratch and the ability of computational thinking. A quasi-experimental design was adopted and conducted a six-hour teaching experiment for four weeks. The subjects were 78 students in the sixth grade of two primary schools. An MANOVA and ANCOVA analysis were employed for statistical analysis of the data. The results show that: The unplugged group significantly outperformed the plugged group in relevance and satisfaction in learning motivation. The unplugged group significantly outperformed the plugged group in Scratch learning achievement. However, there is no significant difference between the two groups in the performance of computational thinking ability. This study suggests to provide unplugged advanced learning activities before the formal programming course, should be more to promote learners' motivation and learning achievement.

KEYWORDS
programming education, computational thinking, scratch, advanced strategy, learning motivation

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

插電與不插電程式教學前導策略對國小程式設計學習成效之影響

廖韋綺[1]，顏榮泉[2*]
[1]國立臺北教育大學數學暨資訊教育研究所，臺灣
[2]國立臺北教育大學數學暨資訊教育學系，臺灣
heart1543@gmail.com，jcyen.ntue@gmail.com

摘要

本研究旨在探討插電與不插電程式教學前導策略對國小程式設計學習動機、Scratch 學習成就及運算思維能力之影響。本研究採準實驗研究法，針對兩間國小六年級共 78 位學生進行為期四週共六小時的教學實驗，並分別以多變量變異數分析及共變數進行統計分析。結果顯示：不插電教學組在學習動機中的相關性及滿足感顯著優於插電教學組；不插電教學組在 Scratch 學習成就上顯著優於插電教學組；運算思維能力的表現上則無顯著差異。本研究建議正規程式設計教學活動前若能提供不插電之前導學習活動，應有助於提升學習者在程式設計的學習動機與學習成就表現。

關鍵字

程式設計教育；運算思維；前導策略；學習動機

1. 前言

隨著快速變遷的資訊科技世代，如何運用科技工具來因應新世代的生活運作模式，儼然成為現今資訊社會公民的基本素養。近年來，愛沙尼亞、美國、新加坡等國家皆以此趨勢來發展其資訊教育的政策。新加坡「智慧國度 2025 計畫」中重視透過資訊科技提升教育品質；而臺灣在 2014 年頒佈的《十二年國民基本教育課程綱要總綱》即將於 2019 年正式實行，總綱中為了因應生活型態的演變，在國高中階段獨立設立科技領域，其中資訊科技課程即是以培養學生運算思維之素養為主軸，學生應用資訊科技工具來學習有效地邏輯思考，並藉由同儕、團隊溝通合作的過程，激發創新思考與解決問題的能力（國家教育研究院，2016），使學習不再侷限於單一學科的知識與技能，而是著重於學習內容與生活密切地連結。

程式設計是最常被用於實踐資訊素養理念的途徑之一。愛沙尼亞是全球推動程式教育最積極的國家，2012 年推動的「虎躍計畫」（ProgeTiiger），從國小一年級就開始進行程式設計的教學。然而臺灣的課綱未明定國小應實施資訊科技課程，但部分縣市已將程式設計列入市訂資訊課程。而目前學習程式設計的途徑主要分為插電與不插電的教學活動，插電的教學活動指的是需運用到電腦，使用線上平台或軟體來教授程式設計，目前國小較常應用視覺化程式語言的工具來進行學習，如：Scratch、Blockly、Kodu 等。許多研究透過問題導向或引導策略來教授視覺化程式語言，結果顯示上述教學模式應能有效提升學習者的學習成效與態度（楊書銘，2008；韓宜娣，2011）；另外，不插電亦即不使用電腦的程式教學活動（Bell, Witten, Fellows, Adams, & McKenzie, 2015），也是近年來的教育趨勢，如：卡牌遊戲、桌上遊戲、大地遊戲等。學習者通常能積極參與與討論不插電編成活動，並理解基本的程式概念（Alamer, Al-Doweesh, Al-Khalifa, & Al-Razgan, 2015）。方廷宇（2018）的研究顯示不插電資訊課程應能有效提升運算思維的學習成效。然而上述的研究中，皆單獨使用插電或不插電其中一種教學策略進行探討，兩種教學策略的實施成效是否存在差異？何種策略更有助於學習成效的提升？至今此議題的相關研究仍屬少見。因此本研究欲探討插電與不插電的教學策略對學習者學習程式設計之影響，以供日後國小程式設計教學之參考。

研究者實務的教學經驗中發現，即使是利用大家普遍公認適合國小的 Scratch 軟體學習程式設計，仍有許多學習者對於完成教師指定的課堂任務感到困難。黃嘉文（2010）的研究指出，雖然透過視覺化的程式語言應能有效降低學習程式設計的障礙，但對於初學者而言，他們不一定能立即地瞭解程式設計的流程與架構，若沒有提供一個學習情境讓學習者進行有意義的思考與體驗，容易造成學習者產生挫折，甚至大幅降低學習程式設計的興致。Ausubel（1963）認為新的學習必須要能夠與個體原有的認知結構連結，才能有效且有意義地學習新知。後續的實證研究指出：利用前導組織教學策略能有助於提升學生的學習態度或成效表現（Mayer, 1979; Gurlitt, Dummel, Schuster, & Nückles, 2012）。

綜上所述，本研究將在進行 Scratch 教學前，透過不同教學前導策略（插電教學組與不插電教學組），讓學習者學習基本的程式概念，探討此奠基活動對國小六年級學習者程式設計學習動機、Scratch 學習成就以及運算思維能力之影響。研究問題如下：

(4)不同程式設計教學前導策略（插電教學組與不插電教學組），對程式設計學習動機是否達顯著差異？

(5)不同程式設計教學前導策略（插電教學組與不插電教學組），對 Scratch 學習成就是否達顯著差異？

(6)不同程式設計教學前導策略（插電教學組與不插電教學組），對 Bebras 國際運算思維能力測驗是否達顯著差異？

2. 文獻探討

2.1. 臺灣現行資訊科技課程與運算思維

目前世界各國的生活已經無可避免地與資訊科技共存，未來更是如此。而資訊科技進步的速度大躍進，原先制式化的學習型態早已經無法適應快速變遷的資訊世

代，取而代之的是要具備「資訊素養」，才能在面對資訊科技的變化時，能根據自身需求，有效地從大量資訊當中篩選出有用的訊息，並靈活運用新的資訊科技於日常生活當中。劉正達和李孝先（2010）也提到資訊素養除了個體本身的背景能力（例如：外語、數學和科學等）之外，更重要的就是能將資訊應用於生活之中的能力。教育部（2016）在《2016-2020 資訊教育總藍圖》中也提到，有效地應用資訊科技熟悉所學習的內容，並在不同情境中應用、解決問題，是身為數位時代公民所應具備的關鍵能力。這一再地顯現出資訊科技教育的重要性。

臺灣資訊科技教育是以培養運算思維為主要理念。運算思維（Computational Thinking）一詞最早是由 Wing（2006）提出，她指出運算思維不僅僅是電腦科學家專有的能力，而是每個人普遍都適用的態度與技能，她將運算思維定義為將一個複雜的問題利用電腦科學的基本概念進行拆解、系統化地重新組織成個體能夠解決問題的思維模式；ISTE和 CSTA（2011）認為運算思維是一個問題解決的過程，包含：形成一個可以用電腦和其他工具來解決問題的方法、有邏輯地組織和分析資料、透過抽象化模型來表示資料、透過演算法思考自動化解決問題，以及結合步驟和資源以最有效的整合方式，確認、分析、實作可行的解決方案，最後則是一般化和遷移這個問題解決過程到多元的問題中。臺灣也有學者對運算思維定義了一套說法。林育慈、吳正己（2016）認為運算思維就是一種能有效應用運算方法與工具解決問題之思維能力；賴和隆（2016）指出運算思維可以反映出資訊科學的基本思考方法，它是一種獨特的問題解決過程。

運算思維與生活密不可分，透過運算思維能培養個體邏輯性地處理訊息、解構問題，並建立問題模型，最後整合出有效、可執行的解決方案。隨著生活型態不斷在改變，這樣的能力更有助於每個人面對及適應未來多元的挑戰。因此培養學生系統化地高層次思考，將習得的運算思維能力應用在學科，甚至是日常生活問題上，是目前資訊教育課程的首要目標。而程式設計能讓運算思維更具體化（劉明洲，2017），它是實踐培養運算思維的重要工具與媒介之一。國家教育研究院（2016）將程式設計列入資訊科技課程的學習內容，目的是培養學生具備資料結構以及遞迴、排序等重要演算法的能力。接下來的兩個小節，研究者將分別探討適合國小使用的插電教學策略和不插電教學策略之相關研究。

2.2. 程式設計課程之插電教學策略

插電教學策略指的是透過電腦，像是利用應用軟體、網路平台的操作，教授程式設計的內容（謝宗翔和顏國雄，2017）。透過插電教學策略學習程式設計的發展較為普遍，但至今尚未有正式的學術定義。研究者將本研究所指的插電教學策略定義為「透過電腦，利用 Hour of Code 網站的實際操作，幫助學習者學習程式設計的基本概念」。

圖形化介面的程式設計軟體或平台廣受國中小，甚至是坊間選擇教材時最重要的考量，如：Scratch、Code.org、Blockly Games、KODU 等，因其有別於傳統重視語法的程式語言，將程式語法分類成一個個以顏色作區別的程式積木，學習者僅須透過滑鼠拖曳的方式，將積木堆疊組合，就能立即呈現程式，這種學習方式大幅降低初學者學習程式設計的門檻。本研究選擇 Code.org 的 Hour of Code 課程作為插電教學組的前導策略，因其程式撰寫模式與 Scratch 相近，且教材的故事性與豐富度較完整，因此研究者認為 Code.org 較能吸引學習者的學習動機，並引領學習者進入程式設計的體驗與學習。

在 LOGO 程式設計中採用引導式發現的教學策略，應能有助於學生產生學習上的轉移（Mayer, 2004）。郭文明（2015）則透過教學簡報引導國小三年級學習者掌控流程，並提供鷹架─「程式指令概念圖」的前導組織策略進行 Scratch 教學，結果顯示應能提高學習者的學習態度。綜合上述，教學者若能透過適當的教學引導策略，或許能有助於學習者進行程式設計的學習。

2.3. 程式設計課程之不插電教學策略

CS Unplugged（Bell, Witten, Fellows, Adams, & McKenzie, 2015）將不插電教學活動定義為不仰賴電腦，不受環境和設備限制，使用紙、筆、卡牌、教具等，甚至是團體動態活動，教授資訊科學的基本概念，學習者通常需要透過實際動手操作，甚至是團隊合作才能完成學習任務，而在體驗學習活動的過程中，同時也能學習新知，實踐做中學的理念。研究者根據上述內容，將本研究所指的不插電教學策略定義為「不使用電腦，而是透過卡牌實際操作的桌上遊戲，幫助學習者在互動過程中學習程式設計的基本概念」。

教育型桌上遊戲是以「教育」為核心，以遊戲的形式將專業知識融入其中，使學習者能在動手操作的過程中專注地學習，又不失趣味性。與程式設計有關的教育型桌上遊戲打破學程式要用電腦的限制，將程式概念融入在遊戲機制中，讓學習者在自己能夠掌握的遊戲狀態下輕鬆地學習。本研究採用交大程式老爹團隊研發的程式教育桌上型遊戲─「海霸」作為不插電教學組的前導策略，它透過海盜爭奪寶藏的故事背景，讓學習者透過卡牌執行程式指令，以最快的速度找到對方寶藏的藏匿地點。藉由遊戲機制的包裝以及可重複體驗遊戲歷程的特性，讓學習者無意地熟悉遊戲玩法，並架構出自己的遊戲策略，進而增進對遊戲所隱含之基本程式概念的理解，如：迴圈、條件判斷以及布林邏輯概念。

程式設計不插電的教學活動通常會以某種形式呈現，並試圖讓學習者自主挑戰問題（Taub, Ben-Ari, & Armoni, 2009）。此種教學策略在國內外文獻中皆證實對程式設計和資訊科學的學習有良好的正向影響（Lambert & Guiffre, 2009；蔡雯欣，2018）。紀小涵（2017）分別針對資訊組長及國小五年級的學習者進行質性晤談。學生對於使用海霸來學習程式設計的接受程度高；而資訊教師針對與桌遊結合的程式設計課程採正向的態度，但考慮到現實資訊課的課程編制和教師掌握學習狀況，仍是課程實踐需要克服的問題。

3. 研究方法

3.1. 研究對象與設計

本研究採準實驗研究法，因受限於班級人數之影響，研究對象為臺北市兩間國小六年級之學習者，依原班組成之立意抽樣，各抽選出兩個班級，共計 78 位研究對象，並以班為單位隨機將研究對象分為插電教學組 38 人、不插電教學組 39 人。依變項為研究對象經為期四週、共六堂程式設計教學實驗之程式設計學習動機、Scratch 學習成就及運算思維能力前、後測表現。

自變項		依變項
教學前導策略 插電教學組 不插電教學組	→	程式設計學習動機 Scratch 學習成就 運算思維能力

圖 1 研究設計架構圖

3.2. 研究流程

所有教學實驗皆由研究者自行教授。教學實驗前，兩組皆進行 Scratch 學習成就測驗（O1）及 Bebras 國際運算思維能力測驗前測（O2）。教學實驗期間分為兩個階段，每階段各三節課，每節課 40 分鐘。第一階段兩組分別實施「Hour of Code─Minecraft 探險家」、「海霸」的程式設計前導教學活動（X1、X2），課程結束後，兩組皆進行程式設計學習動機問卷前測（O3）；接著第二階段則實施相同的 Scratch「電流急急棒」創作遊戲（X3）。教學實驗課程結束後，隨即施測 Scratch 學習成就測驗（O4）、Bebras 國際運算思維能力測驗（O5）以及程式設計學習動機問卷後測（O6）。研究設計流程如表 1 所示：

表 1 研究設計流程

組別	前測	教學實驗一	前測	教學實驗二	後測
插電教學組	O1 O2	X1	O3	X3	O4 O5 O6
不插電教學組	O1 O2	X2	O3	X3	O4 O5 O6

3.3. 教學活動設計

插電教學組採用「Hour of Code 一小時玩程式」中的「Minecraft 探險家」程式教學前導策略。此學習活動以知名的遊戲「Minecraft」進行情境包裝，學習者拖曳、組合視覺化程式積木後，電腦會立即依照學習者組合的程式碼，控制主角在虛擬的世界中，完成砍伐木材、建造房屋、種植作物、判斷熔岩位置並採集煤礦等 14 道關卡任務，若程式執行出現 Bug，則系統會自行跳出提示回饋，幫助學習者重新思考。在循序漸進的關卡任務中，學習者能學習到序列、迴圈、判斷等程式概念，並透過反覆的除錯與引導挑戰以最有效率的方式通過每一道關卡，來深化與精緻化程式邏輯的演算思考。圖 2 為插電教學組學習者專注於組合程式積木。

圖 2 插電教學組學習者專注於積木式程式設計學習

不插電教學組則採用同儕互動性較高的程式設計教育桌上遊戲─「海霸」作為教學前導策略。該款桌遊以海盜爭奪寶藏為情境背景，將基本的程式概念融入遊戲中，學習者需利用移動卡及魔法卡（loop 卡、If/else 卡、海神卡以及媽祖卡），思考如何利用有限的手牌執行最佳的程式指令，讓船隻在海平面上移動、避開障礙物、消除漩渦，最終目標是到達對岸取得黃金寶藏。此外，為增加遊戲的刺激性與趣味性，學習者也可以利用具陷害功能的魔法卡阻礙對方前進。遊戲過程每 3-4 人一組，每組安排一位小幫手，主要負責在學習者執行程式出現 Bug 時，扮演如同程式系統 Debug 的功能，給予立即性的提示回饋，讓學習者重新思考出牌的邏輯，自行修正或澄清卡牌組合出來的程式運算結果。圖 3 為不插電教學組學習者進行程式設計教育桌遊活動畫面。

圖 3 不插電教學組進行程式設計教育桌遊之活動

本研究之 Scratch 創作遊戲教學活動以「電流急急棒」為課程主題，將學習者於前導活動學習到的基本程式概念，應用於專案作品中。課程共分為三堂課，首先介紹 Scratch 程式介面、指令積木區的八大類別、舞台及角色設置等，並提供實際操作的機會。接著讓學習者體驗「電流急急棒」並分析遊戲內容。接下來的兩堂課，學習者兩兩一組，根據教師提供的 Scratch 專案檔，共同討論電流急急棒中障礙物以及球移動和路徑偵測的程式指令，並動手嘗試、實踐與除錯。學習者實際操作演練後，研究者再進行統一講解與釐清概念。圖 4 為學習者創作遊戲時對程式邏輯的討論。

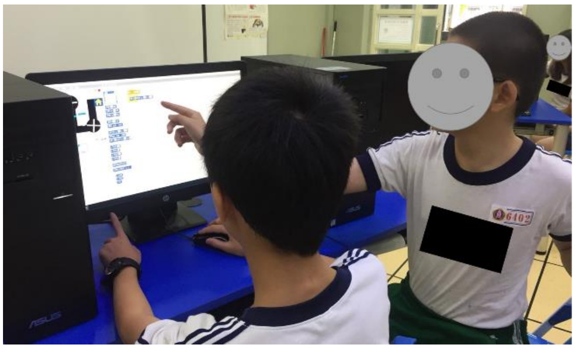


圖 4 學習者運用 Scratch 設計遊戲時對程式邏輯的討論

3.4. 研究工具

本研究之程式設計學習動機問卷採孫琇瑩（2000）改編自 Keller 未出版的 IMMS（Instructional Materials Motivational Scale），研究者將此工具的題項內容與 Keller（2010）的 IMMS 做比較，內容相符合，因此未再進行修改題項內容。Keller 的 ARCS 動機模式有四大要素：Attention（注意力）、Relevance（相關性）、Confidence（信心感）以及 Satisfaction（滿足感），問卷採 Likert’s 五點評定量表，計分方式採正向題分別給予 5 分、4 分、3 分、2 分、1 分，而反向題則相反。所得總分越高的學習者代表其學習動機越積極。本問卷共 36 題，整體總問卷 Cronbach’s α=0.86，屬可信的範圍。

研究者自行編製的 Scratch 學習成就測驗卷根據實際授課內容設計，將試題分為三個層次：基本認識、理解語法以及應用與除錯。基本認識層次屬程式要素的基本知識，為瞭解 Scratch 操作介面與指令積木所代表的意義；理解語法層次即依照題目的敘述，選出符合指令積木語法或流程的選項；應用與除錯層次則是配合題意，選擇能讓程式正確執行的指令積木組合。測驗共 10 題，每題 10 分，滿分為 100 分。測驗題目範例如圖 5 所示。

圖 5 Scratch 學習成就測驗試題範例

運算思維能力測驗卷的編製則根據 2016 至 2017 年 Bebras 國際運算思維能力測驗，篩選出合適的題目，題目的合適性考量到題目隱含的資訊科學概念是否與教學實驗課程相近的資訊科學內涵相符，包括指令序列、迴圈、條件限制、錯誤偵測等概念；難易度則根據易、中、難編製出易 2 題、中 2 題、難 1 題，分別為 15、20、30 分，總分為 100 分。

4. 結果與討論

4.1. 不插電教學組在學習動機中的相關性和滿足感顯著高於插電教學組

表 2 為不同教學前導策略學習者對程式設計學習動機之描述性統計摘要。在 Likert’s 五點評定量表的問卷中，不插電教學組學習者在四個向度上的平均數皆高於插電教學組。

表 2 程式設計學習動機之描述性統計摘要

依變項	自變項	平均數	標準差
注意力	插電教學組	3.49	.58
	不插電教學組	3.67	.84
相關性	插電教學組	3.32	.57
	不插電教學組	3.67	.73
信心感	插電教學組	3.23	.67
	不插電教學組	3.51	.84
滿足感	插電教學組	3.49	.84
	不插電教學組	3.89	.91

接著先針對兩組進行共變量矩陣等式的 Box’M 檢定，同質性檢定結果未達顯著水準（F = 1.20，顯著性為.285），表示各組間變異數無顯著差異存在，排除實驗班級先備知識差異對後測的影響，因此可繼續進行多變量變異數分析（MANOVA）。受試者間效應項檢定分析結果如表 3 所示。注意力（F = 1.11，p = .295）及信心感（F = 2.57，p =.113）兩個向度皆未達顯著水準，表示兩組不同教學前導策略的學習者在注意力和信心感兩個向度的學習動機沒有顯著差異。而相關性（F = 5.68）和滿足感（F = 4.06）兩個向度的顯著性則分別為.020 和.048（＜.05），皆達顯著差異。由表 2 結果得知，不插電教學組的學習者在相關性和滿足感兩個向度的學習動機顯著高於插電教學組。

表 3 程式設計學習動機之受試者間效應檢定

動機向度	型III 平方和	df	平均平方和	F 值	p	Eta 平方
注意力	.58	1	.58	1.11	.295	.015
相關性	2.41	1	2.41	5.68	.020*	.070
信心感	1.48	1	1.48	2.57	.113	.033
滿足感	3.14	1	3.14	4.06	.048*	.051

* $p < .05$

不插電教學組所使用的海霸桌遊，將基本的程式概念融入在卡牌中，學習者能不斷藉由反覆出牌的遊戲過程，增進自己對卡牌內容的理解，進而內化遊戲中所附隱含的程式概念，在後續進行 Scratch 學習時，較能從既有的先備經驗提取相關的知識。而文獻指出學習者在自主的環境下，易樂於獲取知識挑戰自己，並進入學習樂趣和成就感的良性循環（徐新逸、項志偉，2016）。本研究不插電教學組的學習者在海霸的遊戲過程中，需自主控制船的移動方向或自行採取其它的遊戲策略，途中除了經歷海上的阻礙外，不時還會遭遇同儕的陷害，阻擋其順利到達敵方所埋藏的寶藏位置，因此學習者需不斷地透過手牌，思考如何克服這些障礙以獲得遊戲的勝利。在高度互動與自主控制的

遊戲過程中，即使學習者最終沒有獲得遊戲勝利，也能從出牌、陷害的互動中獲得滿足感。

4.2. 不插電教學組學習者 Scratch 學習成就顯著優於插電教學組

表 4 呈現兩組教學前導策略學習者 Scratch 學習成就前、後測之得分情形。在總分皆為 100 分的前、後測試題中，不插電教學組學習者（51.54）前測平均分數低於插電教學組學習者（52.63），而不插電教學組學習者（68.72）後測平均分數高於插電教學組學習者（61.32）。顯示不插電教學組在教學實驗後，學習者於 Scratch 學習成就表現的進步幅度較大。

表 4 Scratch 學習成就前、後測之描述性摘要

教學策略	前測		後測	
	平均數	標準差	平均數	標準差
插電教學組	52.63	15.89	61.32	15.80
不插電教學組	51.54	16.47	68.72	19.22

接著，本研究以 Scratch 學習成就前測之分數為共變量，進行後測分數的共變數分析（ANCOVA），以檢驗兩組研究對象是否受先備知識差異的干擾，而造成研究結果的偏誤。組內迴歸係數同質性檢定結果未達顯著水準（F = 3.75，p = .057），故可繼續進行兩種不同教學前導策略之共變數分析。

表 5 Scratch 學習成就後測共變數分析摘要表

變異來源	離均差平方和	df	平均平方和	F 值	p
共變項	2455.53	1	2455.53	8.73	.004
組間效果	1166.20	1	1166.20	4.15	.045*
組內	20814.58	74	281.28		
總數	350300.00	77			

* $p < .05$

表 5 結果顯示：兩組教學前導策略在排除前測成績（共變項）對後測成績（依變項）的影響後，後測分數達顯著差異（F = 4.15，p = .045< .05）。調整後之平均數，不插電教學組學習者（68.91）的 Scratch 學習成就表現顯著優於插電教學組學習者（61.12）。

本研究所採用的兩種不同程式設計教學前導策略─Hour of Code 以及海霸桌遊，除了具備遊戲的性質外，最重要的是皆強調知識的傳遞，且課程內容所涵蓋的程式概念幾近相同，但在 Scratch 學習成就的表現上，不插電教學組卻顯著優於插電教學組，研究者認為此研究結果可能與本研究海霸桌遊的特色有關。首先是桌遊抽象思考的遊戲過程，學習者需使用手上限有的卡牌，思考與自行決策要如何出牌並預想卡牌的執行結果，才能讓船隻越快抵達寶藏地點，每回合反覆練習的過程能訓練學習者進行抽象思考、思維邏輯以及整合基本的程式執行流程概念。此外，海霸桌遊的遊戲互動除了具有某種程度的競爭性外，同時也提供同儕雙向學習的機會。學習者在主動參與遊戲的過程，藉由自己的出牌經驗和觀察、理解同儕的出牌指令中，加深學習者對程式指令和語法的理解，也提供學習者練習與仿效的機會，以發展出讓自己獲勝的遊戲策略。研究者推論上述海霸桌遊的特色可能是導致兩組在 Scratch 學習成就表現上顯著差異的原因。范丙林（2011）的文獻中也可得到此研究結果的支持，他提及學習者在桌遊的遊戲過程當中不自覺地反覆練習，潛移默化地吸收新知，進而強化學生的學習成效。

4.3. 兩組學習者運算思維能力無顯著差異

表 6 為兩組教學前導策略學習者運算思維能力前、後測之描述性摘要。前、後測總分皆為 100 分的運算思維能力測驗中，無論是前測或後測，不插電教學組平均分數皆高於插電教學組，而兩組在後測的平均數皆低於前測的平均數。接著進行共變數分析檢定，以檢驗兩組運算思維能力之差異。

表 6 運算思維能力前、後測之描述性摘要

教學策略	前測		後測	
	平均數	標準差	平均數	標準差
插電教學組	51.71	32.24	34.87	21.26
不插電教學組	65.00	31.14	38.72	20.22

首先研究者以「運算思維能力前測」為共變量，排除前測成績的差異，以後測分數為依變數，進行組內迴歸係數同質性檢定。結果未達顯著水準（F = .583，p = .448），故可繼續進行兩種不同教學前導策略之共變數分析。表 7 結果顯示 F 值為.01，顯著性為.913，代表兩組研究對象經過本研究所提供之不同程式設計教學前導策略，對其運算思維能力幾乎沒有任何差異。

表 7 運算思維能力後測共變數分析摘要表

變異來源	離均差平方和	df	平均平方和	F 值	p
共變項	7926.81	1	7926.81	24.11	.000
組間效果	3.93	1	3.93	.01	.913
組內	24333.43	74	328.83		
總數	136925.00	77			

* $p < .05$

從表 6 描述性摘要表可得知，兩組研究對象在前後測分數的標準差有縮小的趨勢，但在平均分數卻下降許多。研究者推測其可能原因為：本研究探討的運算思維能力採用 Bebras 國際運算思維能力挑戰賽的題目來檢驗，其試題多以生活情境為主，研究者預想學習者能透過原有的先備知識以及教學實驗課程所使用到的思維模式進行解題，但可能因本研究教學實驗時程較短，學習者還未能有效地將課程所習得的思維模式遷移至其他情境或日常生活中，導致研究結果未顯著提升。另外，試題內容敘述較繁複、冗長，研究對象需耐心閱讀並理解題目內容，才能從中擷取解決問題的重要資訊，並思考解題方案，最終完成作答。此試題模式對於現今數位世代的學習者來說較不擅長，因其生活環

境已熟悉透過電子裝置的媒介來接收資訊，而研究對象若採用速讀或略讀的閱讀方式進行作答，容易錯失細節資訊內容，進而影響測驗的結果。此外，隨著閱讀習慣的改變，研究對象對於大量的文字敘述較容易感到煩躁、沒有耐心，因此也可能因前測經驗而造成研究內在效度的威脅。

5. 結論與建議

本研究旨在探討插電與不插電程式教學前導策略對國小學生程式設計學習成效之影響。依據研究目的與資料分析結果，本研究所獲得的主要結論為：在程式設計學習動機中，不插電教學組學習者在相關性及滿足感的向度上顯著高於插電教學組學習者；不插電教學組學習者之 Scratch 學習成就顯著優於插電教學組；在運算思維能力的表現上，兩組則未達顯著差異。因此，本研究建議程式設計教學前，若能提供學習者不插電的程式設計前導活動，以此奠基將有助於學習者建立學習內容的連結和滿足感，並提升其學習成就。

6. 致謝

本研究承蒙科技部 MOST 106-2511-S-152-001 與 MOST 107-2511-H-152-009 專題研究計畫之經費補助，謹此致謝。

7. 參考文獻

方廷宇（2018）。**桌遊對程式思維學習成效與動機之研究—以國小三年級學童為例**（未出版碩士論文）。臺北市：國立臺北教育大學。

林育慈和吳正己（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**，5-20。

紀小涵（2017）。**導入桌遊於程式教學之可行性：分析「海霸」初玩者的體驗**（未出版碩士論文）。新竹市：國立交通大學。

范丙林（2011）。桌上遊戲應用於環境教育之研究。**100 年度臺北教育大學發展學校重點特色計劃案成果報告書**。臺北市：臺北教育大學。

孫琇瑩（2000）。**不同程度動機提升策略對國小學童網頁教材學習動機之影響**（未出版碩士論文）。花蓮縣：國立花蓮師範學院，。

徐新逸和項志偉（2016）。翻轉教室融入國小六年級資訊課程對批判性思考能力之影響。**課程與教學季刊**，**19**（4），23-60。

國家教育研究院（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**，國家教育研究院 105 年 2 月 4 日教研課字第 1051100273 號函更新二版。

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

教育部（2016）。**2016-2020 資訊教育總藍圖**。臺北市：教育部。

郭文明（2015）。**前導組織策略對國小三年級學生 Scratch 程式設計學習態度與學習成效之影響**（未出版碩士論文）。新北市：淡江大學。

黃嘉文（2010）。**以 Cyber-Physical 環境支援程式設計學習之探究**（未出版碩士論文）。桃園市：國立中央大學。

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**（未出版碩士論文）。臺北市：臺北市立教育大學。

劉正達和李孝先（2010）。國中小教師資訊素養與數位落差現況之研究。**學校行政**，**66**，61-83。

劉明洲（2017）。創客教育、運算思維、程式設計～幾個從「想」到「做」的課程與教學設計觀念。**臺灣教育評論月刊**，**6**（1），138-140。

賴和隆（2016）。應用運算思維於高中資訊教學設計之分享。**教育脈動**，**6**，143-155。

謝宗翔和顏國雄（2017）。**偷插電的資訊科學-教師手冊**。台北：中華民國軟體自由協會。

韓宜娣（2011）。**鷹架支持與自我效能對國小學生程式設計學習表現與學習態度之影響**。臺北市：臺灣師範大學。

Alamer, R. A., Al-Doweesh, W. A., Al-Khalifa, H. S., & Al-Razgan, M. S. (2015). Programming Unplugged: Bridging CS Unplugged Activities Gap for Learning Key Programming Concepts. *Proceedings of the 5th International Conference on E-Learning, ECONF 2015.* IEEE, 97-103.

Ausubel, D. P. (1963). *The Psychology of Meaningful Verbal Learning*. New York: Grune & Stratton.

Bell, T., Witten, I. H., Fellows, M., Adams, R., & McKenzie, J. (2015). *Computer Science Unplugged: an Enrichment and Extension Programme for Primary-aged Children*. Retrieved January 14, 2019, from https://classic.csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf.

Gurlitt, J., Dummel, S., Schuster, S., & Nückles, M. (2012). Differently Structured Advance Organizers Lead to Different Initial Schemata and Learning Outcomes. *Instructional Science, 40*(2), 351-369.

ISTE, & CSTA (2011). *Operational Definition of Computational Thinking for K-12 Education.* Retrieved January 14, 2019, from http://www.iste.org/docs/ct-documents/computation al-thinking-operational-definition-flyer.pdf.

Keller, J. M. (2010). *Motivational Design for Learning and Performance the ARCS Model Approach*. New York: Springer Science & Business Media.

Lambert, L. & Guiffre, H. (2009). Computer Science Outreach in an Elementary School. *Journal of Computing Sciences in Colleges, 24*(3), 118-124.

Mayer, R. E. (1979). Twenty Years of Research on Advance Organizers: Assimilation Theory is Still the Best Predictor of Results. *Instructional Science, 8*(2), 133-167.

Mayer, R. E. (2004). Should There be a Three-strikes Rule Against Pure Discovery Learning? - The Case for Guided Methods of Instruction. *American psychologist, 59*(1), 14.

Taub, R., Ben-Ari, M., & Armoni, M. (2009). The Effect of CS Unplugged on Middle-school Students' Views of CS. *ACM SIGCSE Bulletin, 41*(3), 99-103.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Exploring Evidence that Board Games can Support Computational Thinking

Ching-yu TSENG[1], Jenifer DOLL[2*], Keisha VARMA[3*]
[123] Dept. of Educational Psychology, University of Minnesota, The United States
tseng038@umn.edu, dollx066@umn.edu, keisha@umn.edu

ABSTRACT

Computational Thinking (CT) has caught the attention of researchers, educators, and policymakers in many fields, and has been recognized as an important skill in this increasingly complex society. One challenge emerging in education is finding a way to embed computational thinking curricula into K-12 education. Researchers and educators are exploring ways to provide CT instruction. The study in this presentation investigates an innovative way to teach the CT skill abstraction. According to Grover & Pea's article (2013), "abstraction involves defining patterns, generalizing from specific instances, and dealing with complexity." This study explores the types of strategies students use when they play a game that requires multiple CT skills. Three hundred and sixty-five middle school students played two card games: Ghost Blitz vs. Sushi Go! and completed pre- and post-assessments which were designed based on the definition of abstraction to compare the participants' performance. We analyzed students' gameplay strategies to examine whether participants spontaneously utilized abstraction skills to make a plan to reach their goals when they were playing.

KEYWORDS

abstraction, pattern recognition, game-based learning, K-12 education, unplugged activities

1. INTRODUCTION

Since the computer was introduced to this world, people had experienced significant changes and challenges around their life. It not only alters concrete surroundings and gives more complexity but also influences our thought process to align with algorithm and computing for addressing more complicated problems and corporate with technologies. Many scholars have argued that computational thinking is an important 21st-century skill for K-12 students (e.g. Cetin & Dubinsky, 2017). Nonetheless, a challenge emerges for educators: How to embed CT into K-12 education (Guzdial, 2008)?

To consider this question, we need to verify what elements and features constitute computational thinking. Studies of computational thinking show that it can provide a way to formulate real world's complexity into the systematic and well-structured problematic constitution and assist people to design solutions (Jansen, Kohen-Vacs, Otero, & Milrad, 2018). This thought process is similar to the approach of computer scientists when they are problem-solving and coding. An important point for educators and researchers is to focus on the thinking skills of computer scientists (Grover & Pea, 2018). They should also consider suitable and meaningful complex problems to specific age group to practice CT skills (Jansen et. Al, 2018). The type of activity can influence people's learning, so providing motivational tools would also be important. Therefore, the present study focused on pattern recognition and abstraction and combined with game-based learning context (board games) to explore the evidence that non-programming environment can bolster the learning of CT.

2. BACKGROUND

2.1. Computational Thinking

Jeanette Wing defined computational thinking as a thought process that involves formulating problems and solutions which can be easily carried out by humans and machines (Wing, 2011). Her arguments gave researchers and educators ideas to study in many fields which are outside the computer sciences. Since many scholars recognize CT is an important and necessary survival skill for students to face future challenges (e. g. Cetin & Dubinsky, 2017), educators make an effort to design CT curricula and activities. However, there is still uncertainty about which CT thinking skills are utilized in the process of problem-solving. This issue is under debate. Nevertheless, some scholars developed propositions about the content of CT by observing and organizing the programmers' behaviors when they are solving problems in the programming environment. A framework proposed by Grover and Pea in 2018 focuses on the thought process programmers engage in as they are solving problems. *"Keep in mind the framing of 'thinking like <domain expert> for <domain-specific>' thinking competencies."* p. 22 was their core idea to define CT. They also mentioned there are two facets in CT framework: *concepts*, and *practices*. They utilized abundantly life examples which made readers and educators can easily understand and develop pedagogy without the computer. Based on the framework of CT, this study adopted the concepts of pattern recognition, abstraction, and generalization with the practices of CT to design the experiment.

2.2. Pattern Recognition, Abstraction and Generalization

One of important elements in CT is abstraction which relates to high-level thought process (Wing, 2011), and also is an ability to simplify the complexity for problem solving (Grover & Pea, 2018). However, based on four stages in the development theory of Piaget, it will be difficult for young children to be able to learn abstract (Kramer, 2007). If educators would like to embed CT into K-12 education, the age-related ceiling of learning abstraction is an obstacle to design relevant curriculums. Nevertheless, recent study found evidence that young students can utilize the rationale of abstraction in their general learning process, such as when labeling a diagram (Waite, Curzon, Marsh, & Sentence, 2016). Some studies which recruited elementary schoolers to do experiment within programming context proved children could understand how to program their ideas by instruction and discussion (e.g. Harel & Papert, 1990).

Abstraction relates to decomposing a problem (patterns), hiding the unnecessary elements (pattern recognition), and extracting the common patterns from specific examples (Kramer, 2007). According to Grover & Pea's article (2013), "abstraction involves defining patterns, generalizing from specific instances, and dealing with complexity." p. 39. Align with these definitions, present study designed assessments and learning context for participants. Authors are more interested in the learning effect within non-programming environment. One study organized various unplugged activities and analyzed what kind of CT skills students can learn from them (Brackmann, Román-González, Robles, Moreno-León, Casali, & Barone, 2017). Therefore, board games would be the main learning implement in this study.

2.3. Game-based learning and Gameplay Strategy

Game-based learning has been developing for decades and has proved that it can motivate students to learn (Schifter, 2013). A study gave "Pandemic", which is a collaboration strategy board game, to participants and analyzed their playing process in alignment with CT skills (Berland & Lee, 2011). They provided that the behavior of players when they were discussing the next step in the game can be associated with CT. That means the process of making a gaming strategy might be able to stimulate students to learn and utilize abstraction. These articles provided evidence that teachers can utilize unplugged activities to teach CT skills for students.

When talking about unplugged activities, the most popular way to motivate students is to use games. Game-based learning has been developing for decades and has proved that it can motivate students to learn (Schifter, 2013). A study gave "Pandemic", which is a collaboration strategy board game, to participants and analyzed their playing process in alignment with CT skills (Berland & Lee, 2011). The present study used two different card games: Ghost Blitz which was intended to provide the opportunity for participants to practice pattern recognition and generalization, and Sushi Go! which was intended to not provide any chance for students to learn abstraction skills.

3. RESEARCH DESIGN

The present research aligned with the framework of CT, which was argued by Grover and Pea in 2018, focused on pattern recognition and abstraction concepts, and utilized specific board games to construct non-programming context to examine the learning effect of CT skills.

3.1. Research question

This research focused on exploring whether playing board games which focus on pattern recognition could bolster participants to utilize CT skills for making their winning strategy.

3.2. Participants

There were 365 middle school students at 12-13 years old who were recruited in this study. Participants were randomly assigned into an experimental group (N=217) who played Ghost Blitz and a control group (N=147) who played Sushi Go!.

3.3. Hypotheses

There are two hypotheses in this research. The first hypothesis was that participants who were in the experimental group would outperform participants who played the control game in pattern recognition. The secondary hypothesis was that the experimental group would have better performance than the control group in the skill of generalizing from specific instances.

3.4. Materials

Ghost Blitz (Figure 1) is a card game that requires students to engage in pattern recognition and provides opportunities for learning generalization. There are five wooden objects on the table. It contains two possible scenarios, one where color and shape completely match one of the five objects in the game, and one where the card completely leaves out the color and shape of just one of the five objects. When playing this game, participants have to recognize the color and shape on the card, decide if "the correct answer" is completely matched to a shape and color or if a specific shape and color combination has not shown up, then grasp the correct item from the table as soon as possible. Participants repeatedly practiced pattern recognition skills during this game and had chance to spontaneously learn to generalize features from specific instances, which was a key to make a strategy for how to play and win this game.

Figure 1. Ghost Blitz.

Figure 2. Sushi Go!

Sushi Go! is also a card game which is related to collecting information (Figure 2). Players choose one card from their hand and put it on the table, then each player passes all remaining cards in their hand to the player on their left side and repeats the process until all cards have been chosen. Each player then scores their collection by calculating the value of specific sets of cards they were able to gather during the game. Players have to observe others' choice and try to collect cards which have the highest score in the round to win. This study assumed there is no relationship between the game mechanics and the set of abstraction skills, just observation and collection.

3.5. Procedure

In this study, there are three stages for participants, a pre-test, a structured time for playing the games, and a post-test, which was completed during school hours.

3.6. Measures

The assessments included geometry questions for pattern recognition (Figure 3), math word problems for pattern generalization, and one question in the posttest asking participants to write down their gaming strategies.

Assessments design in generalization was based on the meaning of decomposition and generalization for problem-solving to find open-ended problems in math. For example, buying different and the most balloons for decoration in a budget. The maximum points in the first two parts was 14 for the maximum, 0 for the minimum. For the question of gaming strategy, the study distinguished three different strategy categories for two games to find if there were any behavior models that would be similar to abstraction skills.

Figure 3. Ask participants to distinguish the similarities and differences between these geometry shapes.

3.7. Method

To analyze the data, this study conducted the two-sample t-test first to see if all participants had the same math competences, then the paired t-test for testing if there were significant differences between pre- and posttest in two different sections, pattern recognition and generalization, of experimental and control groups. Categorizing gameplay strategies was the last part of data analysis for finding other evidence to support the results.

4. RESULTS

4.1. Pattern Recognition

Regarding pattern recognition, due to assessments' inclusion of math questions, the data must be clarified that all participants had the same math ability in pretest before the start of analysis. The result showed that there was no statistically significant difference in math competency (p=0.625) between experimental (M=3.26, SD=2.02) and control group (M=3.47, SD=2.18).

The result in pattern recognition was statistically significant in the experimental group's scores between pre-and posttest (M= 0.35, SD=1.73, p=0.003, d=0.202) but was not in the control group (M=0.16, SD=1.63, p=0.229, d=0.099), respectively. This outcome supports the hypothesis that experimental group would outperform control group in pattern recognition (See Figure 4).

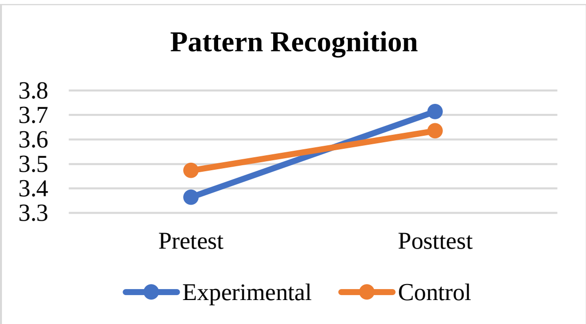


Figure 4. The mean of pre- and posttest in two groups.

4.2. Generalizing from specific instances

Regarding generalizing from specific instances, the results showed that all participants had the same math competence in pretest (p=0.902) no matter experimental (M=4.82, SD=2.64) or control group (M=4.79, SD=2.56). The analysis tests groups individually, the results showed that both experimental (M= 0.02, SD=2.32, p=0.884, d=0.01) and control group (M=0.24, 2.15, p=0.182, d=0.11) are not statistically significant differences between pre- and posttest. This outcome does not support the hypothesis that experimental group would outperform control group in generalizing from specific instances (See Figure 5).

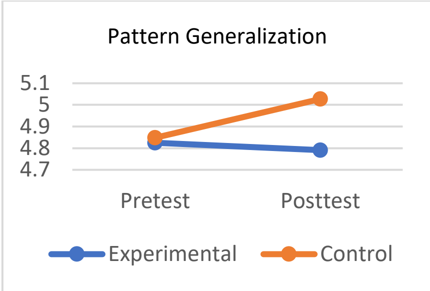


Figure 5. The mean of pre- and posttest in two groups.

4.3. Gameplay Strategies

This research divided the gaming strategy of participants when playing games into three main categories: Non-strategy, simple strategy, and multiple /complex strategy.

According to the categorized outcomes, they represented that Sushi Go! triggered more practice on thinking multiple plans than Ghost Blitz. They might give the explanation why the mean of control group who was assigned to play Sushi Go! had better performance of generating patterns than the experimental group in posttest. However, this result needs more evidence to verify the reliability.

5. DISCUSSION

Based on statistic results, researchers noted that there were no significant differences between pre- and posttest test in two groups respectively in generalization. Two facets, game mechanic and assessments may provide the explanation and need to be considered in future work.

5.1. Game Mechanic

Our results show that Ghost Blitz did provide practice for students to learn pattern recognition. However, in the part of generalization, it did not assist participants in generalizing from specific instances. In contrast, Sushi Go! did not show evidence to support that it can bolster students to learn pattern recognition, but the mean of the control group in the posttest of generalization was higher than in the experimental group. These results are associated with a game mechanic which is worth noticing in the study.

According to the definition of abstraction which relates to make gameplay strategies. We did a simple comparison from participants' strategies and compared with the game mechanics to see what the reason is to get this result from generalization part. Game mechanics and their correspondences to abstraction (See table 1 & 2):

Table 1. Ghost Blitz.

Game mechanics	Abstraction
Recognize the color and shape	Pattern recognition
Compare card and five objects	Remove the unnecessary patterns

Finding two scenarios	Generalizing from specific instances
Strategy to win this game	Goal/the complexity

Table 2. Sushi Go!

Game mechanics	Abstraction
Understand each card's function	Rules understanding, non-abstraction
Combination and collection	Finding your own patterns and specific instances in this game, then extracting common features from your own specific examples
Decide how to collect the highest point to win this game	Making a plan

From these two games' mechanics, they show that Ghost Blitz provides a sound module of patterns and examples (two scenarios) for participants to practice pattern recognition and generalization (formulate a model to win). However, students who played Sushi Go! only received a clear goal, but they had to set up their own patterns and create their own instance to extract the general patterns. In other words, Sushi Go! provides an ambiguous set of patterns for participants, but according to the data analysis, it still provides training for students to practice the generalizing from specific instances.

5.2. Assessments

The design of assessments could also influence the results. Ghost Blitz and Sushi Go! are all visual type games, but the questions in the second part of the pre- and posttest for generalization were constituted by words. The transformation from visual to verbal could impede participants when answering the questions. Moreover, the answer of word problems is difficult to define, it may produce some uncertainty in the process of coding data.

6. CONCLUSION AND FUTURE WORK

6.1. Conclusion

This study has shown that board games can provide chances for players to learn pattern recognition. However, it also illustrated that the association of the type of board games and assessments will influence the results.

6.2. Limitation and Future Work

Computational thinking is viewed as a way to deal with complexity; however, this study utilized board games with simple rules which may be one of limitations to explore the function of board games can provide for learning CT skills. Assessments also are the limitation for evaluating the learning effect. Although CT is related to math problem-solving skill, there is a concern that participants may utilize their math competences to answer questions.

In future work, the game mechanics will be an important point to choose for experiment. The standard tests for evaluating abstract reasoning and pattern recognition will be also considered in the research to produce more reliable evidence.

7. REFERENCES

Berland, M., & Lee, V. R. (2011). Collaborative Strategic Board Games as a Site for Distributed Computational Thinking. *International Journal of Game-Based Learning*, *1*(2), 65-81.

Brachmann, C. P., Román-González, M., Robles, G.,Moreno-León, J., Casali, A., & Barone, D. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM. 65-72.

Cetin, I., & Dubinsky, E. (2017). Reflective Abstraction in Computational Thinking. *The Journal of Mathematical Behavior*, *47*, 70-80.

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

Grover, S., & Pea, R. (2018). Computational Thinking: A Competency Whose Time has Come. *Computer Science Education: Perspectives on Teaching and Learning in School*, 19-38.

Guzdial, M. (2008). Education Paving the Way for Computational Thinking. *Communications of the ACM*, *51*(8), 25.

Harel, I., & Papert, S. (1990). Software Design as a Learning Environment. *Interactive Learning Environments*, *1*(1), 1-23.

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

Kramer, J. (2007). Is Abstraction the Key to Computing? *Communication of the ACM*, *50*(4), 37-42.

Schifter, C. C. (2013). Games in Learning, Design, and Motivation. In M. Murphy, S. Redding, & J. Twyman (Eds.), *Handbook on innovations in learning* (pp.149-164). Philadelphia, PA: Center on Innovations in Learning, Temple University; Charlotte, NC: Information Age Publishing.

Waite, J., Curzon, P., Marsh, W., & Sentence, S. (2016). Abstraction and Common Classroom Activities. Proceedings of the 11th Workshop in Primary and Secondary Computing Education. ACM, 112-113.

Wing, J. (2011). Research Notebook: Computational Thinking—What and Why. *The Link Magazine*, 20-23.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Preliminary Study on Designing Learning Activity of Mathematics Path via Computational Thinking for the Elementary School Students with Learning Disability

Ya-chi CHANG[1], Sung-chiang LIN[2]
[12] Department of Mathematics and Information Education, National Taipei University of Education, Taiwan
g110627002@grad.ntue.edu.tw, lschiang@mail.ntue.edu.tw

ABSTRACT
In recent years, many countries promote information education courses based on computational thinking as the core in the basic education. This kind of courses can train students' ability of systematical thinking and problems solving. However, it is an important research issue of combining computational thinking with other subjects to design learning strategies for students with learning disability. In this study, a mathematical learning activity with computational thinking will be designed for the elementary school students with learning disability according to their special demand such as reading disorder, disorder of written expression, math disability, etc. This activity will teach students how to use computational thinking to recognize the direction and path of the map and to attend mathematics trails activities.

KEYWORDS
computational thinking, learning disability, mathematics trails

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

結合運算思維在國小學習障礙學生的數學步道教學活動設計初探

張雅琪[1]，林松江[2*]
[1]國立臺北教育大學數學暨資訊教育研究所，臺灣
[2]國立臺北教育大學數學暨資訊教育學系，臺灣
g110627002@grad.ntue.edu.tw，lschiang@tea.ntue.edu.tw

摘要

近年來許多國家在基礎教育階段推動以運算思維為核心的資訊教育課程，以培養學生能系統性思考與問題解決能力，而運算思維如何結合其他學科，為學習障礙學生設計合適的教學仍是重要的議題。因此，本研究為國小學習障礙學童設計融入運算思維之數學教學課程，並考量學習障礙學生的特殊需求，教導學生如何使用運算思維概念與空間、地圖方向辨別、路線規劃等概念，進行校園數學步道活動之前導教學活動。

關鍵字

運算思維；學習障礙；數學步道

1. 前言

現今資訊社會發展日新月異，生活中處處皆有資訊設備，幫助人們生活更加便利，資訊科技的發展也影響先進國家的競爭力與影響力，各國愈來愈重視國內資訊教育課程，以美國 2015 年通過每位學生成功法案（Every Student Succeeds, Act）與 2016 年由白宮提出的 Computer Science for All 計劃，將資訊教育與語文、算數、寫作等主科並列為學生重要學習科目，在這一波波教育改革浪潮下，各國為符合科技時勢發展，紛紛在最新訂定的資訊教育課程中表明「運算思維」的重要性，並將其相關概念融入課綱中（林育慈和吳正已，2016）。而臺灣近年正值十二年國教課綱修訂，課綱中的主軸—核心素養期望學生在學習過程中學習適應生活、能系統性思考與具備問題解決能力（教育部，2015），在十二年國教科技領域課綱草案中，將運算思維視為資訊科技課程主軸，以學生的生活經驗、需求以及學習興趣為基礎，在問題解決與實作的過程中培養學生「設計思考」與「運算思維」的知能（教育部，2018）。

但對於有學習障礙學生來說，運用在這些有特殊教育需求學生的運算思維教學活動設計與研究，相對一般學生的教學活動設計來說仍屬少數，相對應的教材與教學設計也相對缺乏（廖晨惠、郭伯臣、白鎧誌和鄔珮甄，2018）。而數學則是學習過程中重要的基本科目，除了訓練學生的基本運算能力外，更培養學生的邏輯思考、推理、以及問題解決的能力，這些能力的培養對一般學生來說已不容易更何況是針對數學學習障礙的學生。因此，對於具有特殊需求的學習障礙學生而言，如何結合運算思維概念設計適合的數學教學活動，仍是相當重要的研究議題。

2. 文獻探討

2.1. 運算思維

2.1.1. 運算思維定義

自從運算思維被提出後，許多學者對於運算思維定義有其不同的看法，目前尚未取得一致的共識，但是大多數學者均認為運算思維應包含抽象化（Abstraction）、資料表示（Data Representation）、問題解析（Problem Decomposition）及演算法思維（Algorithm Thinking）（Wing, 2006; Grover & Pea, 2013; Google, 2019）。Barr 與 Stephenson（2011）則提出運算思維運用於各領域之範例，並將運算思維分為資料搜集、資料分析、資料表示、問題解析、抽象化、演算法思維、自動化、同步、模擬等，並將各項元素對應至資訊科學、數學、科學、社會研究與語言藝術，如下所述：

1) 資料搜集（Data Collection）：透過發現問題蒐集可以分析的資料。
2) 資料分析（Data Analysis）：將問題進行分類。
3) 資料表示（Data Representation）：將問題進行拆解，例如在資訊科學方面可以使用資料結構，將資料以陣列（Array）、鏈結串列（Linked list）、堆疊（Stack）、佇列（Queue）、圖片（Graph）和雜湊表（Hash table）等進行配置。
4) 問題解析（Problem Decomposition）：定義問題可以用哪種解決方法。
5) 抽象化（Abstraction）：透過有系統的包裝，並將問題利用現有的方法解決。
6) 演算法思維（Algorithms & Procedures）：以資料科學為例，透過學習經典演算法，並針對某一特定領域的問題進行實作演算法。
7) 自動化（Automation）：將問題透過自動化的方式解決，如生活中常碰到的，天黑時電燈要自動亮起。
8) 同步（Parallelization）：把可以同時執行之狀況，同步進行。
9) 模擬（Simulation）：模擬實際情形。

2.1.2. 運算思維在一般教學現場之研究

在運算思維的教育相關研究中，有學者提出學生在閱讀、數學、算術皆需要運用運算思維，（Wing, 2006），在一般教學現場也有許多學者將運算思維融入教學活動中（楊冰清和張進寶，2018；黃淑賢、陳虹如、葉芯妤、蔡一帆和施如齡，2018），在認知發展較成熟

的高中生與大學生課堂中，不論是進行插電或不插電的教學，研究結果皆顯示能提升學生的學習動機與學習成效。而現今學生自小開始接觸電子與資訊類產品，大多能輕鬆熟練使用，國小教師可以盡早善用此優勢教導學生，提升學生的運算思維能力（Wing, 2008）。

2.2. 學習障礙

在國小課業輔導的過程中，發現學習障礙的學生在學習上有許多困難（例如：書寫困難、文字理解困難），學科學習成就不佳，長久下來易造成學生學習意願低落（侯禎塘，2004），依 107 學年度國小就學的身心障礙學童中統計資料中可知學習障礙所占比例約 35%，是所有障礙別中人數最多，每個學童的障礙類型不同，異質性偏高，其學業成就不佳並非因智力不足。而根據教育部（2002）身心障礙及資賦優異學生鑑定標準第十條對學習障礙的定義統稱神經心理功能異常而顯現出注意、記憶、理解、推理、表達、知覺或知覺動作協調等能力有顯著問題，以致在聽、說、讀、寫、算等學習上有顯著困難者，雖然學習障礙是終生伴隨的障礙，但是只要經過適當、有效的教育，學習障礙者也能發揮其潛能，如發明家愛迪生、美總總統威爾遜、作家安徒生，都是歷史上雖有學習障礙卻非常著名的偉人（柯華葳，2000）。

2.2.1. 運算思維在學習障礙教學活動之研究

近幾年有學者提出將運算思維融入特殊教育需求學生的課程是值得探究的（Snodgrass, Israel, & Reese, 2016）。根據 Braefoot（2016）提出針對特殊教育需求學生進行運算思維教學設計，有幾項原則：（1）運算思維為電腦運算課程中最重要的核心概念，特殊教育需求學生可透過肢體動作或結合教具，建立物件與序列概念；（2）教師依照特殊教育需求學生日常所需學習的任務設計課程，增進學生的問題解決能力，例如將任務進行分解步驟與調整，此能力也能運用在數學計算或運用計算的學習。（3）科技可以幫助特殊教育需求學生進行學習、資訊、休閒等教學活動，也能幫助學生達成其原本能力所無法達成的任務。國內有學者嘗試將運算思維融入學習障礙學生的數學教學中，以不插電的教學模式結合學生實際生活情境進行數學教學活動（廖晨惠等人，2018）。

2.3. 數學步道

數學步道是一種動態而非靜態式的數學教學活動，有別於讓學生在教室中進行紙筆計算或是教具的操作理解就可以完成的，教師必須帶領學生走出教室，到校園數學步道中的一個地方，舉例來說：有一個學習活動設計在學校的中庭，學生就必須到中庭才能完成這個學習活動，當學生的學習離開了中庭，就非該數學步道的學習活動，而是一般的數學學習活動（王佩蓮，1995；林碧珍，2001；張怡貞和簡淑貞，1998；Winicky-Landman, 1999）。

蔡寶桂（2000）指出數學步道即是利用校園內的環境，包含球場、圓柱、校門等設計成富有思考及創意性的日常生活數學問題，讓數學的學習脫離了僵化的公式和數字計算，也讓校園成為隨手可得的教學題材。當沿著事先設計好的路線走時，教師可依當下情境提出合適的數學問題，學生和陪同的家長可以一起進行腦力激盪討論、或是個人思考、也可以和帶領的老師相互討論，解決這一連串的數學問題（黃敏晃，2005）。在真實的校園生活情境中，應用學生已學過的數學知識，在數學步道中進行探索學習。

數學步道的設計與實施，可依以下幾點原則（曹雅玲和陳鴻綸，2007）：

1、數學步道的設計應依校園環境的特性結合數學教學的目標以設計活動。

2、數學步道的設計不應只有活動單的設計，應包含活動說明、活動內容及學習者的想法與作法的記錄等部分來設計活動。

3、數學步道設計中的活動說明應闡示學習者應具有的能力、可能的解題策略和教學應注意的事項，以利學習活動的進行。

4、數學步道設計中的每個單元，可包含不同類別的活動，如計算與非計算等。如此，可使活動的進行更多元化、 更生動活潑，避免枯燥乏味，甚至可和遊戲互相結合。

5、數學步道的設計應配合課本教學目標的進度，使學生能將課本和生活結合。

6、數學步道的設計可配合天候、時間，針對每一個環境作不同的教學設計。

7、數學步道的實施中，學生的解題策略應經過分組的共同討論，進行腦力激盪 及分析判斷，而非個人認知所論定。

8、數學步道的實施中，學生的解題策略可行或不可行，教師可藉由學生的發表，開發其思維的盲點，而使其豁然開 朗。

9、數學步道的實施中，同年齡的學生仍有差異，同一活動在不同學習者的進行中，仍需配合該學習者的數學概念及認 知發展。

10、數學步道的實施中，應視單元的內容和環境的特性，配合天候、時間來進行。

3. 研究發展與設計

本研究設計結合運算思維與數學步道活動，進行國小學習障礙學童之數學教學課程活動設計，並使用質性研究觀察法，在教學過程當中透過錄影與觀察紀錄的方式記錄學生的學習過程與反應狀況，並在課程結束後收回學生的學習單分析學生的作答狀況，對應學生的運算思維能力與數學概念，並在課程結束後進行訪談，了解學生在學習過程中的認知與想法。

3.1. *結合運算思維之校園數學步道教學活動*

本研究對象為學習障礙生，主要是數學障礙、書寫障礙、閱讀障礙等學習性學習障礙，另一方面，研究也

指出學生對數學不感興趣且低成就的學生參與度低（蔡寶桂，2000），因此，本研究運用數學步道在設計教學活動時也考量這三個類別的特殊需求，設計合適的教學輔助工具，以提高其學習興趣。而如前所述，學習障礙包含閱讀障礙的學生，這類學生對於文字閱讀相對困難，本研究設計參考廖晨惠等（2018）學者於 2018 年的研究，在前導教學時使用紙牌等圖形化教具，減輕閱讀障礙的學生對於文字閱讀的困難。如圖 1 所示為校園數學步道地圖，圖 2 則是路徑卡。

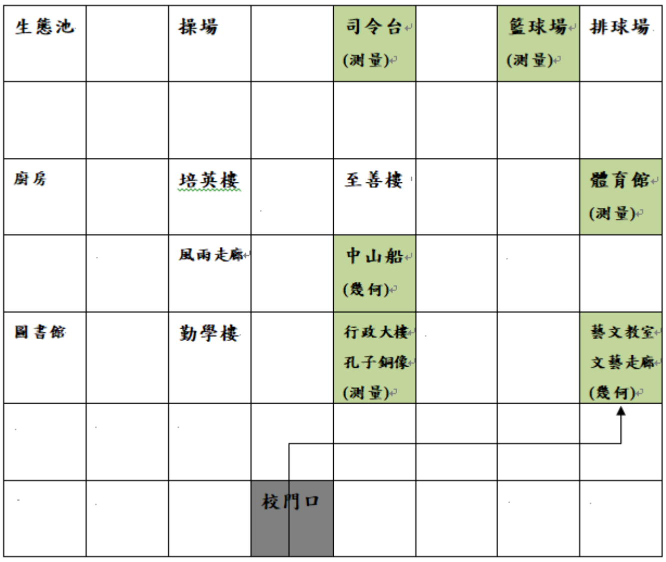


圖 1 校園數學步道地圖

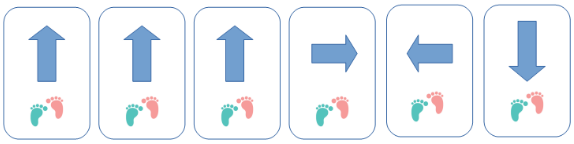


圖 2 路徑卡

此外，本研究除考量閱讀障礙學生受限於閱讀能力，需教師提供題目的說明，搭配使用實體教具讓學生去操作外，也參考 Braefoot（2016）的運算思維教學活動設計，並考量其他類型學習障礙學生的需求，進行相對應的設計，以減輕學習障礙學生的學習負擔。表 1 為本研究設計之數學步道教學流程、運算思維、數學概念對應表，表 2 則是本研究設計的前導課程教學流程，結合運算思維元素引導學生思考解析問題、進行模擬等，以有限的方向紙牌排出路徑，排列出如何到達指定景點關卡。

數學步道教學活動能讓學生驗證及練習在課堂上所學的知識和技能，透過結合運算思維的元素進行設計，協助學生達到「做中學，學中做」的目標，並書寫數學日記以記錄進行數學步道教學活動的收穫及感想，除了可以提高學習樂趣外，也能透過分享數學概念，提供學生實際運思的機會，培養運算思維的能力。

表 1 數學步道前導教學流程、運算思維、數學概念對應

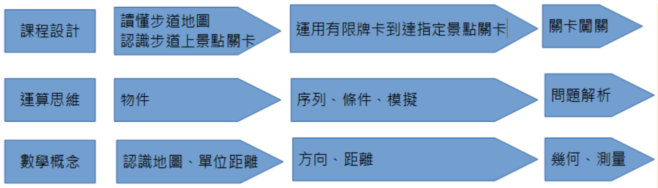


表 2 校園數學步道教學活動範例

教學活動流程	教具	時間	運算思維概念
1.引起動機 辨認轉彎方向、辨識地圖上的方位。 認識地圖上的各個校園建築與景點，能辨識在校園的哪個方位。 2.主要活動 教師示範，從校門口出發，運用有限的紙牌到達指定景點關卡，請學生思考是否有其他路徑也可以到達。 老師:我們現在來玩一個遊戲，要按照老師的指定地點，用你手中的紙牌排出等一下要走的路徑。 任務一：校門口在地圖的下方，是南方，我們要去藝文教室要怎麼走？先往北方？ 任務二：還有其他路徑可以到藝文教室嗎？ 讓學生操作，運用紙牌到達指定景點關卡，教師給予該景點較簡化的問題，讓學生在教室中練習該指定景點關卡所需的數學概念。 教師：藝文教室外有好多圖形？你看到了甚麼？甚麼圖形的數量最多？	校園數學步道地圖 學習單 路徑方向卡	30 分鐘	物件 序列 條件 模擬 問題解析

4. 結論與討論

本研究結合運算思維的數學步道教學活動設計，在教學前、教學中與教學後都需有周密的構想，而在教學活動中，也需考量學生程度以及不同的學習障礙類型，而在設計教學活動前，也需謹慎選擇地點場景，並可進一步配合競賽或獎賞，提升學生學習興趣，讓學生不僅可透過正向回饋中提升學習興趣，也能從競賽活動中學習到較抽象的概念，培養其運算思維的能力，可做為本研究設計未來強化與調整的方向參考。

5. 參考文獻

王佩蓮（1995）。**環保生活與校園環境步道**。臺北：臺北市立師範學院環境教育中心。

林育慈和吳正已（2016）。運算思維與中小學資訊科技課程。**教育脈動**，**6**，5-20。

林碧珍（2001）。以「數學步道」的設計協助職前教師發展數學連結的能力。**國教世紀**，**198**，15-26。

柯華葳（2000）。**學習障礙學生輔導手冊**。臺南：國立臺南師範學院。

侯禎塘（2004）。特殊教育需求兒童數學學習困難之特質、教學策略與創意遊戲數學之應用。**特殊教育論文集**，47-66。

曹雅玲和陳鴻綸（2007）。數學步道活動在數學教學之應用。**科學教育月刊**，**302**，21-37。

教育部（2015）。**十二年國民基本教育領域課程綱要核心素養發展手冊**。臺北：教育部。

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

教育部（2002）。**身心障礙及資賦優異學生鑑定辦法**。臺北：教育部。

黃敏晃（2005）。漫談數學步道。**大中至正**，**4**，8-14。

黃淑賢、陳虹如、葉芯妤、蔡一帆和施如齡（2018）。**創客奇蹟–遊戲任務導向之運算思維活動設計初探**。發表於 2018 年運算思維教育國際會議，香港教育大學，香港。

張怡貞和簡淑貞（1998）。校園數學步道在啟蒙數學教育上的應用。**教育研究**，**64**，10-24。

楊冰清和張進寶（2018）。**不插電的計算思維教學活動在高中課堂教學中的應用—以《二進制卡牌》課程為例**。發表於 2018 年運算思維教育國際會議，香港教育大學，香港。

廖晨惠、郭伯臣、白鎧誌和鄔珮甄（2018）。**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**。發表於 2018 年運算思維教育國際會議，香港教育大學，香港。

蔡寶桂（2000）：透過 WEB-BBS 進行數學步道之溝通、解題。**竹縣文教**，**22**，6-11。

Barr, V., & Stephenson, C. (2011). Bringing Computational Thinking to K-12. *ACM Inroads*, *2*(1), 48-48. doi:10.1145/1929887.1929905

Braefoot. (2016). *Activities for Pupils with Special Educational Needs.* Retrieved January 15, 2019, from https://barefootcas.org.uk/activities/sen/

Google. (2019). *Exploring Computational Thinking*. Retrieved January 18, 2019, from https://edu.google.com/resources/programs/exploring-computational-thinking/

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

Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional Supports for Students with Disabilities IK-5 Computing: Findings from a Cross-case Analysis. *Computers and Education*, *100*, 1-17.

Winicky-Landman,G. (1999). *Assignments for Mathematical Tour of Haifa.* Israel: Technion.

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 on the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Designing Unplugged Activities for Learning Computational Thinking in the Context K-2 Pupils' Afterschool Coding Club

Eunice Eno Yaa Frimponmaa AGYEI [1], Jari LARU[2*,] Kati MÄKITALO[3]
[123] Faculty of Education, University of Oulu, Finland
eagyei@student.oulu.fi, jari.laru@oulu.fi, kati.mäkitalo@oulu.fi

ABSTRACT

This paper presents a study of developing computational thinking (CT) practices by designing and testing unplugged and plugged coding materials for 1st and 2nd grade pupils between the ages of 6 and 8. Materials were used as a tool in unplugged phase of afterschool coding club. Based on the analysis, it is evident that pupils were able to apply the concepts used in unplugged learning activities in their ScratchJr projects followed unplugged part of the coding club.

KEYWORDS

computational thinking, unplugged coding, instructional design

1. INTRODUCTION

CT involves identifying a problem, evaluating the problem, and reasoning to design solutions that solve the identified problem (Wing, 2008). "Unplugged" is a term that describes computational thinking activities carried out without computers (Bell, Alexander, Freeman, & Grimley, 2009). Learners study computing concepts through activities and games without interacting with a computing device using simple materials such as sheets of paper and crayons to learn CT concepts (Bell, Alexander, Freeman, & Grimley, 2009).

2. AIM OF THE STUDY

The aim of the study was design computational thinking lessons for K-2 students participating into after school code club in the Finnish primary school. The pedagogical design included both unplugged and plugged activities for learning basics of the computational thinking

In this paper unplugged part of the design will be presented, because it was considered crucial for learning computational thinking in the context of non-experienced child-programmers.

3. CONTEXT

The context for this experiment was an after-school programming club in a primary school in Northern Finland. Participants were 17 pupils (8 girls and 9 boys aged 6-8). Original idea was to select only pupils who are non-experienced on coding, but only 13 out of the selected 17 fulfilled the criterion. 4 participants were randomly selected from the rest of the applicants in order to fill all seats available.

The first author of this paper was the designer and instructor of the activities in the coding club. The sessions were arranged in a computer laboratory within the host school.

4. INSTRUCTIONAL DESIGN AND MATERIALS

Instructional design for unplugged coding activities consists of a series of processes employed to understand, improve and develop educational materials that impact knowledge and skills (Gagne & Briggs, 1974).

4.1. Design of Unplugged Coding AActivities

Unplugged activities are designed with the aim of demonstrating CT concepts through authentic real-life examples to capture the interest of learners and motivate them.

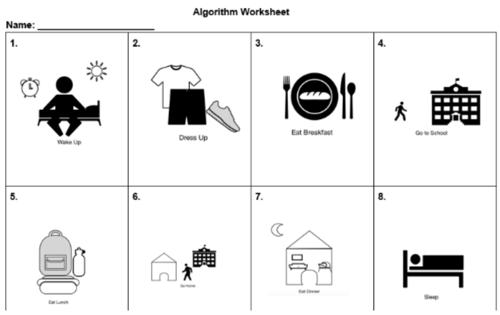


Figure 1. Algorithm worksheet.

Algorithm WWorksheet: This unplugged activity focuses on daily routine of learners starting from the first thing they do in the morning to the last thing they do at night. A predefined set of stickers (see Figure 6) were designed to be used alongside a worksheet (storyboard) as can be seen in Figure . This activity enforces the concept of orderly sequences in algorithms.

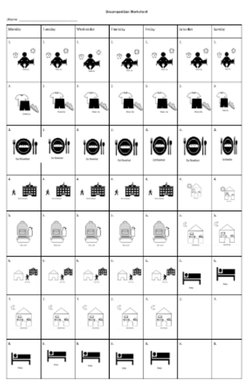


Figure 2. Decomposition worksheet.

Decomposition: This activity focuses on teaching learners how to break down large tasks into several, small, easily solvable forms. Here, the weekly activities of learners are broken down into daily activities. Figure shows a weekly worksheet organize their weekly activities into an orderly daily sequence.

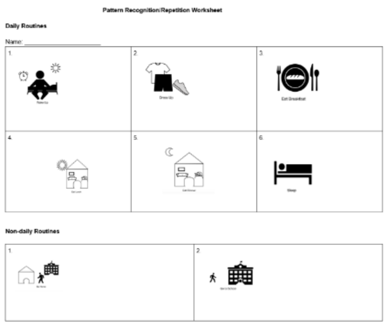


Figure 3. Pattern recognition worksheet.

Pattern Recognition: This worksheet (see Figure 3) follows the decomposition worksheet, focusing on the identification of repeated and non-repeated routines within one's weekly activities.

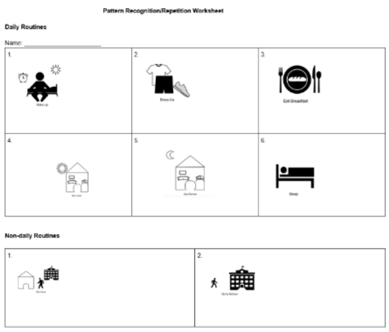


Figure 4. Decisons worksheet.

Decisions: This activity focuses on making decisions based on a condition. Decisions are made on a daily basis regardless of age. This worksheet seeks to teach decision making by performing an acidity based on a condition. See Figure .

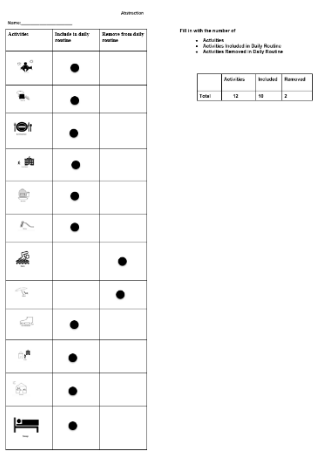


Figure 5. Abstraction worksheet.

Abstraction: This activity focuses on teaching the concept of abstraction. The worksheet (see *Figure 5*) used for this activity is designed to support the inclusion and exclusion of activities from one's daily set of activities listed in the algorithm worksheet

Figure 6. Stickers for worksheets.

Figure 6 represents sample stickers designed for the worksheets. These stickers can be customized to include options for gender-specific stickers to encourage authentic learning, and motivate pupils to learn CT

5. RESULTS AND CONCLUSIONS

Preliminary results from the code club reveal that unplugged learning materials were successful method to teach principles of computational thinking.

Based on the analysis, it is evident that pupils were able to apply the concepts used in unplugged learning activities in their ScratchJr projects followed unplugged part of the coding club. Future research will continue with un-plugged design, but include also physical unplugged activities in addition to paper-based work.

6. REFERENCES

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.

Gagne, R. M., & Briggs, L. J. (1974). Principles of Instructional Design. Oxford, England: Holt, Rinehart & Winston.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. (2009). Ubiquitous Computational Thinking. *Computer, 42*(10), 100-102.

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

Computational Thinking and Subject Learning and Teaching in K-12

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Research on Gamified Collaborative Learning in the Cultivation of Computational Thinking

Yuyu LIN [1], Jiansheng LI [2*]
[12] Nanjing Normal University, China
1845115460@qq.com, 2869753244@qq.com

ABSTRACT

Computational thinking is one of the indispensable information literacy of today's students, and large researches have demonstrated that gamified collaborative learning has a significant positive effect on students' learning. Therefore, this paper designed a collaborative learning environment under Kodu to explore the influences of gamified collaborative learning on computational thinking from two aspects of individual factors (learning interests and attitudes, self-efficacy), and participation, the effect of socially shared regulation in collaborative learning. This investigation took two classes students of 10th grade as the sample and collected data via survey questionnaires and students' discussion recordings, trying to provide new ideas for cultivating students' computational thinking.

KEYWORDS

gamification instruction, collaborative learning, computational thinking, educational games

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

游戏化协作学习在运算思维培养中的应用研究

林育瑜[1]，李建生[2*]
[12]南京师范大学教育科学学院，中国
1845115460@qq.com，2869753244@qq.com

摘要

运算思维是当今学生不可或缺的信息素养之一，大量研究表明游戏化协作学习对学生的学习有显著的积极作用。因此，本文设计了一个 Kodu 游戏环境下的协作学习环境，以高一两个班级的学生为研究样本，通过调查问卷和录音收集数据，从个体因素（学习兴趣和态度、自我效能感）和协作学习过程中的参与度、社会调节学习效果两方面探究游戏化协作学习对学生运算思维的影响，为培养学生的运算思维提供新思路。

关键字

游戏化教学；协作学习；运算思维；教育游戏

1. 问题提出

学生运用自身的知识体系去发现问题、分析问题和解决问题的过程就是思维的过程。运算思维代表了这个过程，它在分析和协调的基础上，通过相关操作步骤，解决某个问题，这是一种思维模式。培养学生的运算思维有利于提高自身的推理创新能力，形成良好的运算思维体系，从而强化对信息技术知识的吸收（Cooper, 2000）。运算思维的最基本层次是理解算法和使用算法解决问题。

目前，国内外对运算思维的研究集中在信息技术课程、数学课程或编程中如何培养学生的运算思维，主要包括利用数学题目、相关案例，以及开发相关工具等。但在这些方式中，教学策略较为传统，学生容易囿于教师预设的框架下，结果常常是学生积极性不高、主动性不足，或是问题太难放弃了，或是解决了问题就止步不前，缺乏新意。

另一方面，研究表明，游戏化协作学习不仅可以增加社会互动，支持真实活动，鼓励通过更深层次的讨论来解决问题，而且可以促进知识模式的重构或共建（Chen, Wang, & Lin, 2015）。该学习方式不仅有助于激发学生的学习兴趣，让玩家自己参与学习，有效地增强课堂教学效果（黄燕梅，2015），而且使得学生能在社会环境中获得同伴的认可，促进学生的自我效能感。另外，Cagiltay（2010）认为，计算机教育游戏可能是提供更有趣的学习环境以获取知识的有效方式。而社会调节学习是游戏化协作学习的重要组成部分。

针对上述问题和需求，本研究利用游戏化协作学习来培养学生的运算思维。该教学方式以社会调节学习理论（Socially shared regulation of learning，简称 SSRL）为基础，探索游戏化协作学习对学生运算思维的影响，旨在通过有效干预学生的协作过程，促进学生的学习参与度，提高学生的学习兴趣、学习态度和自我效能感，进而内化为学生的运算思维。本研究中所探讨的"运算思维"指的是学生能在给定的任务环境下，理解任务，将一个较为复杂的总任务分解为一个个具体可操作的子任务，并且监督子任务的操作执行，最终通过分析、设计、修正来较好地完成总任务。

2. 文献回顾

社会调节学习（SSRL），指在小组活动的学习过程中，小组成员协商讨论，互相合作，实现知识建构（Hadwin, Jarvela, & Miller, 2011）。在社会调节学习中，学习目标和学习标准由小组成员共同构建，学习结果也是通过小组协作共同完成。Lee A（2014）分析了基于计算机支持的协作学习过程中发生的讨论，对 SSRL 过程进行划分，主要分为七个方面，即计划和目标设定、时间计划、小组分工、任务监督、内容监督、任务评价和内容评价。研究表明，学生在小组协作中的参与度与交互中 SSRL 发生的次数呈正相关（Winne & Perry, 2000）。Volet 等人（2009）发现，学生平等参与互动促进其高质量认知调节。并且，高质量的以学习任务为中心的社会互动，表现出较多的 SSRL 和凝聚力。

游戏化协作学习能使学习环境从传统的以教师为中心向以学生为中心转变，在该环境中学生能够更加积极主动地参与学习（Watson, Mong, & Harris, 2011）。以往研究中，游戏化协作学习主要集中在使用某个计算机教育游戏平台，让学生在虚拟学习环境下进行协作学习，从中探究学生的动机、态度、兴趣、技能、学习行为等。研究表明，游戏化协作学习能有效提高学生的自我效能感、学习兴趣和态度（Burguillo, 2010），从而促进学生的问题解决能力和协作技能（Sánchez & Olivares, 2011）。也有研究发现，学生在游戏协作环境下对解决问题的看法有所改善，问题解决能力得到明显提升（Chang et al., 2012），并且对需要较强逻辑推理能力的学科，如数学，也更为积极（Ke, 2014）。只有零星的研究提出，基于游戏的学习能够促进学生在编程课程中的学习参与度和运算思维（Baytak & Land, 2010）。

综上所述，虽然游戏化协作学习对个体因素的影响、对学生逻辑思维和问题解决能力的研究较多，但游戏化协作学习对运算思维的影响并不清楚。因此，本研究基于社会调节学习（SSRL）理论来研究游戏化协作学习对运算思维的影响，探究游戏化协作学习与运算思维的关系，以及游戏化协作学习是如何影响个体因素和学习参与度。另外，由于本研究的实验为课堂实验，不存在需要学生计划合作时间的问题，而小组分

工常常是低质量讨论调节，因此本文的协作学习调节模式主要为计划和目标设定、任务监督、内容监督、任务评价和内容评价五个过程，教师将在这五个方面进行干预。基于此，提出如下几个问题：

1. 经过游戏化协作学习后，学生的运算思维水平是否提高了？

2. 游戏化协作学习是如何影响学生的个体因素（学习兴趣和态度、自我效能感）的？

3. 学习参与度、SSRL 成绩和运算思维成绩的关系是怎样的？

4. 个体因素（学习兴趣和态度、自我效能感）和参与度、SSRL 成绩的关系是怎样的？

5. 个体因素（学习兴趣和态度、自我效能感）和运算思维成绩的关系是怎样的？

3. 研究方法

3.1. 样本来源

本研究于福建省漳州市云霄第一中学完成。被试是高一两个班共 95 名学生，高一（7）班 48 人（对照组，男生 29 人，女生 19 人），高一（8）班 47 人（实验组，男生 29 人，女生 18 人），年龄均在 15-18 岁之间。

3.2. 试验样本的比较

对两个班级的运算思维前测成绩进行独立样本 T 检验。结果表明，t 检验中 Sig(双尾)为 0.407，大于 0.05，说明两个班级运算思维成绩水平不存在显著差异。因此，两个班可以作为本研究的实验样本。

3.3. 研究情景

计算机教育游戏平台的选择。微软公司设计的 Kodu 是一款游戏设计软件，专门为学生提供计算机编程的早期入门（Maclaurin, 2011）。与 Scratch 和 Alice 等同类型软件相比，Kodu 语言完全是事件驱动的，有利于学习（Touretzky, Gardner-Mccune, & Aggarwal, 2016）。在 Kodu 中，学生不仅可以成为游戏玩家，还可以成为游戏设计师，这为他们可以设计自己的游戏并从过程中学习提供了机会。因此，本研究利用 Kodu 作为本次课程的游戏环境。另外，本研究选取信息技术课程进行本次实验，历时六周，每班每周一次课（45 分钟）。为了避免由不同教学者带来的差异影响，两班均由同一教学者进行教学。

3.4. 实验过程

两个班级在第一次课均接受了 kodu 基本操作界面和编码方式的介绍，并被给予了一定时间熟悉 kodu 的界面和操作，同时，学生填写自我效能感、学习兴趣和态度问卷，并进行了运算思维测试。接下来的四次课，两个班级均在 Kodu 环境下分小组协作学习，完成教师布置的学习任务，每个小组均有一位同学负责全程录音。总的三次学习任务，第二、第三次任务分别在第一、第二次任务的基础上提高难度。不同的是，实验班的小组协作学习受教师根据 SSRL 五个环节加以干预，而对照班的学习过程中，教师并不进行干预。具体 SSRL 五个环节的干预方式为：（1）计划和目标设定。教师布置完一节课的任务后，小组讨论计划，并填写计划表。（2）任务监督。小组讨论过程中，教师实时督促小组完成的进度。（3）内容监督。教师适时询问小组完成的内容。（4）任务评价。任务结束后，教师提问小组对本组任务完成的评价。（5）内容评价。教师提问小组任务内容的完成情况，并要求小组填写任务内容评价问卷。两个班级均在第 2 次课和第 4 次课的最后十分钟进行运算思维测试（测试一和测试二），并在第 5 次课完成小组合作作品。在第 6 次课，两个班级均为自主游戏开发时间，每人无需局限在教学者布置的学习任务中，完全自主发挥。在任务活动完成后，提交个人作品。然后进行运算思维测试（测试三），完毕，再次填写自我效能感、学习兴趣和态度后测问卷。

在学习活动完成后，对收集的数据和录音进行分析，录音部分剔除与学习内容无关的对话。其中，运算思维成绩=（作品成绩+三次测试成绩平均分）/2；鉴于合作作品人均参与为 1/3，所以作品成绩=合作作品成绩*0.3+个人作品成绩*0.7；参与度得分=个体参与讨论次数；SSRL 成绩=高质量讨论次数*2+低质量讨论次数*1。成绩由授课教师和笔者共同评定取平均值。其中，作品成绩的评分者间信度为 0.917，SSRL 讨论质量的评分者间信度为 0.954。剔除某些自变量信息缺失的问卷和试卷，最后剩余有效样本为对照班 45 人，实验班 45 人。

3.5. 测量工具

本研究中的测量工具包括运算思维四次测试卷、学习兴趣和态度问卷、自我效能感问卷，还有判定 SSRL 五个过程讨论质量的评价方法。

运算思维四次测试卷由教授信息技术十余载的任课教师依据其丰富的经验和笔者共同根据运算思维的层次编制而成，主要包括理解概念、监督任务执行、分析任务、创造算法、纠错程序和设计复杂的游戏。题型涉及选择题和主观题。选择题涉及概念理解、任务监督、任务分析以及纠错程序，如“Kodu 是对象，它的属性是？”。主观题涉及创造算法和设计复杂的游戏，如“针对问题添加一只章鱼后，kodu 消灭了一只章鱼后游戏就胜利了，应该如何改进？”。每份测试题项难度逐渐提升，总分均为一百分。经测试，运算思维四次测试的信度系数分别为 0.851，0.869，0.89，0.832。

学习兴趣和态度问卷改编自 Hwang G J 和 Chang H F 关于移动学习环境中形成性评价影响学习兴趣和态度的测试问卷（Hwang & Chang, 2011），针对本研究的具体内容，对问卷进行适当的改编。兴趣问卷包含 12 道题，包括对游戏化学习、协作学习以及游戏化协作学习的兴趣三个方面，经测试，该部分问卷的信度系数为 0.841。态度问卷包含 14 道题，包括对游戏化学习、协作学习、游戏化协作学习以及对学习编程的态度四个方面，经测试，该部分问卷的信度系数为 0.938。

自我效能感问卷由 Pintrich 等人编制（Pintrich, et al., 1991），包含 8 个题项，采用李克特七级计分法，从非

常不同意到非常同意，分别记 1-7 分。重测信度系数为 0.93。总分越高，表示个体的自我效能感越高，学生对于完成这门课程的自信心越高。

判定 SSRL 五个过程讨论质量的评价方法采用 Lee A 关于社会调节学习的评价指标（Lee, 2014），如表 1 所示。

表 1 讨论质量评价指标

调节过程	高质量	低质量
计划和目标设定	小组讨论明确详细的任务计划 明确次级目标 达成关于目标和计划的共同协议	发布指导性问题作为计划或发布重述的任务提示 发布一个简单的目标计划
任务监督	讨论了任务难点和对任务的理解 根据最初的计划，进行元认知监督 发现有效解决策略	检查错误操作等 时间监督 在没有讨论或协议的情况下检查了每个指导性问题的完成情况
内容监督	通过分析推理监督任务反馈的准确性 质疑其他成员意见，阐述理由 达成协议的任务解决方案	没有建设性的意见 没有原因地就某个问题达成一致或不一致的意见
任务评价	确定次级目标的完成，并讨论原因	浅层次地检查所有问题的完成度，没有任何理由
内容评价	说明目标达成的具体原因 检查相关但未提及的课程概念 成员对任务内容给予确认	浅层次地评估任务内容完成度和概念的使用，没有详细的解释 没有作出评价

3.6. *分析方法*

本研究将检验两个班级的运算思维成绩差异、以及前后测差异来确定游戏化协作学习对运算思维的影响。进而分别检验个体因素（学习兴趣和态度、自我效能感）的前后测差异，来确定游戏化协作学习对个体因素的影响。接着检验两个班级的参与度和 SSRL 成绩是否存在差异，以及参与度、SSRL 与运算思维的关系来检验采用 SSRL 五个环节干预指导协作学习的效果。最后分析个体因素（学习兴趣和态度、自我效能感）对参与度、SSRL 成绩以及运算思维成绩的影响。本研究的所有分析都利用软件 SPSS 22 进行。

4. 研究结果

4.1. *游戏化协作学习对运算思维的影响*

为了排除运算思维前测成绩的影响，分析两个班级在学习活动后的运算思维水平是否存在显著差异，将运算思维前测成绩作为协变量，后测成绩为因变量，进行协方差分析（ANCOVA）。结果显示，其显著性为 0.687，大于 0.05，说明实验班和对照班的方差相等，即两个班级的运算思维后测成绩不存在显著差异。但是，经事后分析，LSD 检验显示实验班得分（67.25）高于对照班（62.11），并且前后测均值的增长（18.95）显著高于对照班（9.4）。

由于两个班级的后测均值相较于前测均值都有显著提高，为了确定游戏化协作学习的作用，将两个班级作为一个整体，对运算思维前、后测成绩进行配对样本T检验。结果表明，配对样本的均值存在显著差异，因为显著性（双尾）为0.000，小于0.05，说明游戏化协作学习对提高学生的运算思维有一定的作用。

4.2. *游戏化协作学习对个体因素的影响*

4.2.1. *学习兴趣*

为分析两个班级在学习活动后的学习兴趣是否存在显著差异，将学习兴趣前测成绩作为协变量，后测成绩为因变量，进行协方差分析（ANCOVA）。结果表明，其显著性为 0.948，大于 0.05，说明两个班级的学习兴趣后测成绩不存在显著差异，但两个班级的后测均值相较于前测均值都有所提高（对照班前测均值：46.64，后测均值：49.56；实验班前测均值：46.27，后测均值：47.67）。

4.2.2. *学习态度*

为了分析两个班级在学习活动后的学习态度是否存在显著差异，将学习态度前测成绩作为协变量，后测成绩为因变量，进行协方差分析（ANCOVA）。结果表明，其显著性 P 值为 0.935，大于 0.05，说明实验班和对照班的方差相等，即两个班级的学习态度后测成绩不存在显著差异，并且两个班级的后测均值相较于前测均值都有所下降（对照班前测均值：59.08，后测均值：58.18；实验班前测均值：59.02，后测均值：57.20）。

4.2.3. *自我效能感*

为了分析两个班级在学习活动后的自我效能感是否得到改善，分别对两个班级学习前后的自我效能感成绩进行配对样本 T 检验。结果表明，对照班和实验班的 P 值均大于 0.05，即两个班级的自我效能感在学习活动前后没有得到显著改善。但相比于对照班前后测自我效能感的均值下降（前测均值：45.52，后测均值：45.1），实验班的自我效能感一定程度上得到了提升（前测均值：45.71，后测均值：46.48）。也就是说，教学者在 SSRL 五个环节的干预一定程度上提高了学生的学习信心和期望。而对照班自我效能感后测均值下降的原因可能是虽然同样是小组协作学习，但缺乏逻辑引导，同伴学习的讨论质量不高，导致学习期望不足。

4.3. *参与度、SSRL 成绩与运算思维成绩的关系*

4.3.1. 参与度与 SSRL 成绩

为了解两个班级在不同的教学实施下参与度和 SSRL 成绩是否存在显著差异，将两个班级的参与度和 SSRL 进行独立样本 T 检验。结果表明，两个班级在参与度（P=0.012）和 SSRL 成绩（P=0.010）上存在显著差异。说明在教师依据 SSRL 五个过程进行教学干预的情况下，学生的参与度和 SSRL 有明显的提升。

为进一步了解参与度与 SSRL 成绩的关系，将参与度和 SSRL 成绩进行相关分析。结果表明，对照班和实验班的学习参与度与 SSRL 成绩均呈显著的正相关。其中，对照班 $r=0.967$，$p<0.01$；实验班 $r=0.957$，$p<0.01$。即参与度越高，学生的 SSRL 成绩越高。

4.3.2. 参与度、SSRL 成绩与运算思维成绩的关系

为了解两个班级参与度、SSRL 成绩与运算思维成绩的关系，将参与度、SSRL 和运算思维成绩进行两因素方差分析。

结果表明，对照班参与度、SSRL 及两因素交互作用的显著性均高于显著性水平 0.05，所以参与度和 SSRL 两因素对对照班的运算思维并没有显著影响。实验班的参与度显著性为 0.025，小于显著性水平 0.05，SSRL 显著性为 0.065，大于显著性水平 0.05，所以实验班的参与度对运算思维成绩有显著影响，而 SSRL 成绩对运算思维成绩的影响不显著。

4.4. 个体因素和参与度、SSRL 成绩的多元方差分析

进一步分析两个班级个体因素（学习兴趣和态度、自我效能感）对参与度、SSRL 成绩的影响，将个体因素和参与度、SSRL 成绩进行多元方差分析。结果表明，对照班个体因素对参与度、SSRL 成绩没有显著影响，P 值均大于 0.05。实验班个体因素中的学习兴趣、学习态度以及学习兴趣和态度的共同作用均对参与度、SSRL 成绩有显著影响。

4.5. 个体因素和运算思维成绩的多因素方差分析

为明确两个班级个体因素（学习兴趣和态度、自我效能感）对运算思维成绩的影响，将个体因素和运算思维成绩进行多因素方差分析，如表 2。

表 2 个体因素和运算思维成绩的多因素方差分析结果

因变量：运算思维成绩

班级	来源	F	显著性
对照班	修正的模型	11.380	.034
	自我效能感	2.595	.230
	学习兴趣	21.878	.014
	学习态度	2.788	.213
a. R 平方 = .994（调整的 R 平方 = .907）			
实验班	修正的模型	2.416	.022
	自我效能感	7.782	.042
	学习兴趣	.395	.851
	学习态度	2.086	.248
a. R 平方 = .963（调整的 R 平方 = .564）			

由表 2 可知，对照班运算思维成绩在自我效能感和学习态度上的显著性分别为 0.230 和 0.213，说明运算思维成绩在自我效能感和学习态度上都不存在显著差异，但在学习兴趣上的显著性为 0.014，说明在学习兴趣上存在显著差异，即学生的学习兴趣越高，其运算思维成绩越好。而实验班的运算思维成绩在学习兴趣和态度上的显著性均大于 0.05，说明运算思维成绩在学习兴趣和态度上都不存在显著差异，但在自我效能感上的显著性为 0.042，说明在自我效能感上存在显著差异，即学生的自我效能感越高，其运算思维成绩越好。

5. 结论

本研究基于社会调节学习理论设计了一个 Kodu 游戏环境下的协作学习环境，与大多数探索学科培养运算思维的研究不同，本研究更倾向于从个体因素（学习兴趣和态度、自我效能感）和协作学习过程中的参与度、SSRL 调节效果两方面来发现培养学生运算思维的效果。

从运算思维成绩上看，虽然对照班和实验班的运算思维成绩不存在显著差异，但两个班级整体的运算思维成绩前后测配对样本的均值存在显著差异，即经过游戏化协作学习，学生的运算思维有了很大的提高，说明该学习方式具有一定的积极作用。基于此，分别对两个班级的个体因素（学习兴趣和态度、自我效能感）和参与度、SSRL 成绩进行分析。

在学习兴趣和态度方面，两个班级均没有显著差异，但学习兴趣的后测均值相较于前测均值都有所提高，而学习态度的后测均值相较于前测均值有所下降。即，在游戏化协作学习后，虽然学生的学习兴趣有所提升，但从协作学习过程中的讨论对话中可以发现，协作学习遇到的分歧会让学生的学习态度产生变化。例如，小组其中一名学生认为："我觉得我们可以先试试一只 Kodu 打章鱼的效果。"而小组另一名成员则认为："我们可以直接打对战啊，多爽！"因此，虽然学生对游戏任务的兴趣不减，但协作体验感影响学习态度，最后可能变成小组只有一两个学生共同完成任务，而另一个则选择自己完成，从学生的小组协作学习评价问卷中也印证了这一点。所以，在学习活动开始前分组的时候，除了让学生自行选择，教学者还应考虑学生个性，尽可能让小组成员个性多元化。

另外，虽然两个班级的自我效能感在学习活动后均没有得到显著提高，但实验班的后测均值相较于前测有所提升，而对照班的后测均值有所下降。也就是说，对于协作学习过程进行 SSRL 五个环节的指导干预一定程度上促进小组的讨论，学生从同伴获取的认同感比来自教师的认同感更为强烈，进而建立更高的信心和期望（Lucia et al., 2009）。受同伴认可的次数越多，自我效能感也就越高。相反，对照班虽然同样是小组协作学习，但缺少教学者的干预，学习讨论质量不高，学习期望不足。

从参与度和 SSRL 成绩上看，实验班学生的学习参与度明显比对照班积极，SSRL 成绩也比较高，又一次证明了教学者采用 SSRL 五个环节进行教学干预的有效作用。然而，参与度、SSRL 成绩和运算思维成绩的分析结果

显示，参与度对运算思维成绩有显著影响，而 SSRL 成绩对运算思维成绩的影响不显著。这或许是因为，运算思维讲究层次化，随着学生参与讨论次数的增多，所涉及的运算思维的逻辑层次也就越多，因此成绩也就越高。而 SSRL 成绩反映的可能只是某一层次学生的讨论质量，因此对运算思维成绩的影响也较为单一。此外，通过个体因素和参与度、SSRL 成绩的分析可以看出，学习兴趣对参与度具有显著影响，说明可以通过提高学生的学习兴趣促进学习参与度，进而提高运算思维水平。

从个体因素和运算思维成绩的分析结果上看，对对照班而言，兴趣是最好的老师，引导他们在本课程积极学习，获得运算思维成绩的提高。而对实验班而言，自我效能感促使他们在学习过程中逐渐从同伴那里获得自信，相互协作完成学习任务来获得运算思维成绩的提高。

总的来说，基于 Kodu 的游戏化协作学习在培养学生的运算思维上有显著的效果。另一方面，尽管 SSRL 五个过程的教学干预指导对培养运算思维的效果不够明显，这可能是由于班级小组太多，教学者无法监督到每一个小组的学习进程，从而错失给予干预指导的良机，但其对提高学生的自我效能感和学习参与度有一定的作用。

6. 参考文献

黄燕梅 (2015)。中职英语小组合作学习模式下的游戏化教学实践。**考试周刊，102**，104-105。

Baytak, A., & Land, S. M. (2010). A case study of educational game design by kids and for kids. *Procedia - Social and Behavioral Sciences*, *2*(2), 5242-5246.

Burguillo, J. C. (2010). Using game theory and competition-based learning to stimulate student motivation and performance. *Computers and Education*, *55*(2), 566-575.

Cagiltay, N. E. (2010). Teaching software engineering by means of computer-game development: challenges and opportunities. *British Journal of Educational Technology*, *38*(3), 405-415.

Chang, K. E., Wu, L. J., Weng, S. E., & Sung, Y. T. (2012). Embedding game-based problem-solving phase into problem-posing system for mathematics learning. *Computers & Education*, *58*(2), 775-786.

Chen, C. H., Wang, K. C., & Lin, Y. H. (2015). The comparison of solitary and collaborative modes of game-based learning on students' science learning and motivation. *Journal of Educational Technology & Society*, *18*(2), 237-248.

Cooper, S. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Hadwin, A., Jarvela, S., & Miller, M. (2011). Self-regulated, co-regulated, and socially shared regulation of learning. *Handbook of Self-Regulation of Learning and Performance*. New York: Routledge, 65-84.

Hwang, G. J., & Chang, H. F. (2011). A formative assessment-based mobile learning approach to improving the learning attitudes and achievements of students. *Computers & Education*, *56*(4), 1023-1031.

Sánchez, J., & Olivares, R. (2011). Problem solving and collaboration using mobile serious games. *Computers & Education*, *57*(3), 1943-1952.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: a case study on mathematics learning during design and computing. *Computers & Education*, *73*(1), 26-39.

Lee, A. (2014). Socially shared regulation in computer-supported collaborative learning. *Dissertations & Theses - Gradworks*.

Lucia, A. D., Francese, R, Passero, I., & Tortora, G. (2009). Development and evaluation of a virtual campus on second life: the case of seconddmi. *Computers and Education*, *52*(1), 220-233.

Maclaurin, M. B. (2011). The design of kodu:a tiny visual programming language for children on the xbox 360. *Acm Sigplan Notices*, *46*(1), 241-246.

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: impact on educational effectiveness and student motivation. *Computers & Education*, *52*(1), 1-12.

Pintrich, P. R., et al. (1991). *A Manual for the Use of the Motivated Strategies for Learning Questionnaire (MSLQ).* Michigan: The Regents of the University of Michigan.

Touretzky, D. S., Gardner-Mccune, C. & Aggarwal, A. (2016). Teaching "lawfulness" with kodu. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 621-626.

Volet, S., Vauras, M., & Salonen, P. (2009). Self- and social regulation in learning contexts: an integrative perspective. *Educational Psychologist*, *44*(4), 215-226.

Watson, W. R., Mong, C. J., & Harris, C. A. (2011). A case study of the in-class use of a video game for teaching high school history. *Computers and Education*, *56*(2), 466-474.

Winne, P. H, & Perry, N. E. (2000). Measuring self-regulated learning. *Handbook of Self-Regulation*. Academic Press, 531-566.

Yang, Y. T. C., & Wu, W. C. I. (2012). Digital storytelling for enhancing student academic achievement, critical thinking, and learning motivation: a year-long experimental study. *Computers & Education*, *59*(2), 339-352.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Teaching Research on Cultivating Pupils' Computational Thinking in Scratch Course

Zhi-lin LI[1*], Hong YU[2], Yu-xiao XU[3]
[123] School of Education and Information Technology, South China Normal University, China
851095185@qq.com, 454534852@qq.com, 1525661031@qq.com

ABSTRACT

This paper starts with the analysis of the problems existing in the development of primary school students' computational thinking, and explains the necessity of developing computational thinking training in the teaching of Scratch programming in primary schools. In the Scratch program design teaching, the cultivation method, training method and evaluation direction of computational thinking are discussed. Proposed a "new courses - Process Design - Instruction Teaching - Evaluation Improvement - Integrated innovation - Share summary" to the teaching process and to formulate and analyze problems, abstract modeling, algorithm design, optimization, migration as the six dimensions of the evaluation to solve computational thinking. Combined with classroom teaching cases, it shows that students develop problems, analyze problems, abstract modeling, algorithm design, optimization schemes, and migration method solving capabilities, thus contributing to the formation of computational thinking.

KEYWORDS

computational thinking, Scratch, teaching, primary school

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

在 Scratch 课程培养小学生计算思维的教学研究

李芷琳[1*]，余红[2]，徐玉晓[3]
[123]华南师范大学教育信息技术学院，中国
851095185@qq.com，454534852@qq.com ，1525661031@qq.com

摘要

本文从分析小学生计算思维培养的发展存在的问题入手，说明在小学 Scratch 程序设计教学中展开计算思维训练的必要性。探讨了在 Scratch 程序设计教学中，计算思维的培养方式、培养方法和评价方向。提出了“导入新课-流程设计-指令教学-评价改进-综合创新-分享总结”来的教学流程以及以制定问题、分析问题、抽象建模、算法设计、优化方案、迁移方法解决作为计算思维评价的六个维度的观点。并结合课堂教学案例，说明培养学生制定问题、分析问题、抽象建模、算法设计、优化方案、迁移方法解决能力，从而促成计算思维的形成。

关键字

计算思维；Scratch；教学；小学

1. 研究背景与问题的提出

计算思维目前是国际教育领域和计算机领域的研究热点。2006 年，美国卡内基·梅隆大学周以真（Jeannette M.Wing）首次提出“计算思维”的概念后，计算思维引起广泛关注。发达国家如美国、澳大利亚、英国、加拿大等率先开展了计算思维培养与教育活动。在美国，“卡内基·梅隆大学与微软合作建立卡内基梅隆计算思维中心（Carnegie Mellon Center for Computational Thinking）。”除此之外，还有计算机科学教师协会（CSTA）、国际教育技术协会（ISTE）等通过举办各类关于计算思维的会议来剖析计算思维，并提出了许多培养学生计算思维的目标和案例。“在澳大利亚，每年政府都会在全国范围内推广 Bebras Australia Computational Thinking Challenge ”，鼓励全体国民尤其是中小学生通过在线资源来学习计算思维。在英国，2012 年提出计算思维将作为“学校计算机和信息技术课程”的一项关键内容，并在 2014 年将计算思维作为新的计算机科学课程的主要内容。由此表明，计算思维已在全世界引起广泛关注。

与此同时，计算思维也在国内引起广泛关注。在国内，2010 年在“C9 高校联盟计算机基础课程研讨会”上发表的《C9 高校联盟计算机基础教学发展战略联合声明》中提出要把培养学生的计算思维能力作为计算机基础教学的核心任务。（王荣良，2012）

2017 年版的《普通高中信息技术课程标准》更是将计算思维作为高中信息技术学科的核心素养之一，并将“学会运用计算思维识别与分析问题，抽象、建模与设计系统性解决方案”作为课程目标之一。（中华人民共和国教育部，2018）计算思维作为信息技术课程改革的主要内容之一，在中小学的信息技术课程中，计算思维将作为一个重点被研究者所关注。

计算思维与其他思维不同，任友群等人认为，计算思维是一种独特的解决问题的过程，是一种可以帮助人们更好地理解和分析复杂问题的思想方法，从而形成具有形式化、模块化、自动化、系统化等特征的问题解决方案。（任友群、隋丰蔚和李锋，2016）当在课上学生动手实践把问题分解成小问题，规划执行的顺序，辨认出其中的模式，评估解决方案，关注重要的细节时，实际上就是给自己武装了解决问题的技能，这些技能可以帮他们学习数学，科学和其它学科，甚至于解决日常生活问题。除此之外，计算思维对于生活在科技和 AI 智能自动化时代孩子们来说，更重要的让他们拥有一种生活技能。未来的职业人士需要有效地应用和创造科技。在这个前提下，计算思维就变成了一个必须品，远远超越了追求个人兴趣的意义 。

但由于很多从业教师对计算思维的定义以及如何将计算思维融入教育的理解和认识不同，计算思维教育的落地有较大的偏差。因此，如何将计算思维很好地融入课堂教学迫在眉睫。

2. 国内外计算思维培养的现状研究

2.1. 国外研究现状

2006 年周以真教授发表了题为 Computational Thinking 的文章，提出了一种建立在计算机处理能力及其局限性基础之上的思维方式——计算思维，并得到了微软公司的大力支持。2010 年，美国德保罗大学胡博米尔·佩科维奇（Ljubomir Perkovic）教授等基于 ACM 主席丹尼的七项“伟大的计算原理”概念分类的基础上构建了计算思维的组成类别及其相应的关键词：计算、通信、协调、回溯、自动化、评估、设计。2011 年，美国圣菲研究所李丽娟提出“使用-修改-创新”（Use-Modify-Create）框架，代表学生在计算思维中的认知和实践活动的三个阶段，该框架被认为是支持和深化青少年在 NSF（National Science Foundation）项目中获得计算思维的进程模式。2012 年，韩国借助 LOGO 语言的算法学习项目对小学教师计算思维进行职前培训。2013 年，南安普敦大学的辛西娅·塞尔比（Cynthia Selby）和约翰·沃拉德（Jhon Woollard）提出计算思维包括算法思维、评估、分解、抽象、概括五方面要素。英国计算机协会组织相关专家对计算思维进行研讨，提出了计算思维在欧洲发展的行动纲领。2014 年，英国保罗·柯松（Paul Curzon）教授等开发了一个在课程上发展计算思维的教学框架，该框架分为四步循环：步骤 1，确定学习单元的原因以及主题，为下一步评估框架的建构提供依据，在学习过程中重复步骤 2-4；步

骤 2，确定课程评价框架和预期成果，使全部学生能基本完成；步骤 3，学习完成学习单元活动所需的计算思维概念；步骤 4，使用计算思维概念来识别可能需要的技术以支持完成学习活动。这些表明，计算思维已经引起国外学者的广泛关注并成为重点研究对象。

2.2. 国内研究现状

国内学者也对计算思维的概念及其培养提出了自己的见解。

任友群等人认为，计算思维是一种独特的解决问题的过程，是一种可以帮助人们更好地理解和分析复杂问题的思想方法，从而形成具有形式化、模块化、自动化、系统化等特征的问题解决方案。（任友群、隋丰蔚和李锋，2016）李锋等人则从不同的视角对计算思维进行解读，

从认知特征角度认为，计算思维是一种具有技术原科学性特征的、与信息化社会相适应的心理工具；从表现特征角度认为，计算思维是一种基于信息技术解决问题的能力；从信息环境角度认为，计算思维是头脑内部信息系统和外部自然信息系统的合理互动过程。（李锋和王吉庆，2013）当前，国内比较权威的当属“新版普通高中信息技术课程标准”对计算思维的界定：计算思维是以计算机领域的学科方法界定问题、抽象特征、建立结构模型、合理组织数据，通过判断、分析与综合各种信息资源，运用合理的算法形成解决问题的方案，总结利用计算机解决问题的过程与方法，并可迁移到与之相关的其他问题解决中的一种学科思维。国内学者在计算思维的培养模式也做出了一定研究。2010 年，中国科学院牟琴教授构建了基于计算思维的探究的教学模型，运用该模型在 C 语言程序设计课程中有效促进了学生计算思维的训练。裴佳通过在高校课堂中应用翻转课堂教学模式，为高校“大学计算机”课程教学寻找一种新的能促进计算思维训练的途径。2014 年，中国科学院王丹力研究的 T-maze 编程工具，针对 5-9 岁儿童的计算思维培养，包括抽象、问题分解和创造性等计算思维方面的培养。兰州大学郭守超等探索提出了以教师为设计者、组织者、引导者，以 App Inventor 为学习工具，通过师生合作、生生合作等形式，学生主动利用计算思维思想解决问题，来培养计算思维能力的学习模型。但国内学者对于计算思维的研究主要集中在高校，而思维训练是一个长期的过程，从小学开始进行思维训练是非常有必要的，迫切需要在小学阶段加大研究力度。

3. 小学生计算思维培养的教学研究思路

3.1. 教学研究的理念

根据国际教育技术协会与计算机科学技术教师协会联合高等教育、工业、K-12 教育中的领导者共同定义的计算思维，即计算思维包括了制定问题、分析数据、抽象、设计算法、选择最优方案、推广六个要素。Scratch 程序设计软件是美国麻省理工学院（MIT）自主设计开发的一套面向 8 岁以上的少年使用的简易编程工具。其编程只需像搭积木似的拖曳定义好的编程模块就可以实现传统的程序编写功能。小学 Scratch 的程序教学是以 Scratch 的程序设计软件为依托，通过问题驱动的方式进行教学，旨在培养小学生的计算思维。

3.2. Scratch 程序设计教学设计策略

3.2.1. 设计原则

3.2.1.1. 任务驱动原则

以范例程序为模板，引导学生从范例程序入手，通过分析、抽象建模、算法设计等步骤自主完成程序的编写。

3.2.1.2. 开放性原则

鼓励学生在完成所要求程序的基本功能后运用积累的编程经验丰富程序的功能或者程序呈现的效果。

3.2.1.3. 思行合一原则

鼓励学生发散思维挖掘创意丰富程序，也鼓励根据实际情况对创意进行筛选，以保证程序的现实逻辑性和算法的逻辑性。

3.2.1.4. 真实性原则

以现实生活的例子作为程序编写的基础，培养学生善于发现生活的意识。

3.2.2. 基本内容设计

3.2.2.1. 活动目标确定

对象是小学六年级学生，这些学生已经掌握了 Windows 画图软件使用、互联网信息浏览和下载等基本的计算机操作技能。

3.2.2.2. 项目与工具的选择

Scratch 程序设计以及 Scratch 程序软件

3.2.2.3. 过程设计

面向思维发展的 Scratch 程序设计按照活动从模仿到创造的过程，学生先通过模仿教师的范例程序进行程序编写，在根据自身的程序编写经验，丰富程序，从而达到创造新程序的过程。设计的 Scratch 程序设计教学流程如下：

表 1 Scratch 程序设计教学流程

教学环节	教学策略	计算思维培养目标
新课导入	关注程序功能及效果的提问	制定问题能力 分析问题能力
流程设计	关注程序运行过程流程图的设计	抽象建模能力
指令教学	关注新指令用法讲解	算法设计能力
评价改进	关注脚本作品的评价	优化方案能力
综合创新	关注脚本作品的创新	迁移解决方法能力
分享总结		

3.2.2.4. 评价设计

本研究结合 Scratch 程序设计的特点，将计算思维的六要素进行修改，将制定问题、分析问题、抽象建模、算法设计、优化方案、迁移方法解决作为计算思维评价的六个维度。进一步将六个维度细分为 10 个维度，从而构建计算思维评价维度。

表 2 计算思维评价维度

一级维度	二级维度
制定问题	了解程序主要能够实现的功能
	了解程序主要呈现的效果
分析问题	正确回答程序中每个角色间的关联
	正确回答哪些角色是需要编写脚本
	正确回答程序运行的直观效果
抽象建模	能用流程图对程序的功能实现过程进行绘制
算法设计	根据流程图自主选择合适的 Scratch 程序指令来实现程序的功能
优化方案	能分析现有程序脚本设计的错误、漏洞
	能选择最佳解决问题的程序脚本设计
迁移方法解决	能对程序功能进行丰富，增加更多的程序功能和效果

3.3. 以 Scratch 程序设计教学《赛跑》为例的教学设计

3.3.1. 教学主题

Scratch 程序设计以《赛跑》为主题。学生首先要学习“跑步竞赛”的规则和流程，在此基础上发现问题-怎样运用 Scratch 程序设计来完成整个赛跑中不同角色的功能；定义问题与分析问题-赛跑的程序过程的流程图设计；解决问题-赛跑程序的编写、测试、分享、评价、优化。学生在整个过程中将进行数据抽取-抽象模型-算法优选-编程模拟，经历一个计算思维的训练过程。

3.3.2. 教学对象

教学对象为小学六年级学生，他们课前已经上过 2 次 Scratch 程序设计课程，对 Scratch 程序设计的界面比较熟悉，对 Scratch 程序的一些简单指令和算法也有了一定的了解，有了一定的 Scratch 程序设计的基础。

3.3.2. 教学过程设计

根据计算思维培养的几个方面进行教学设计，形成了以“导入新课-流程设计-指令教学-评价改进-综合创新-分享总结”这六个步骤教学过程。

表 3 教学过程设计

教学环节	实施意图与措施	对应计算思维培养目标
导入新课	教师运行“赛跑”的范例程序，通过电子导学问卷，提问学生范例程序的功能以及哪些角色需要进行脚本编写，角色与角色之间的关联是什么。教师可以通过课堂云端数据知道学生对范例程序的了解程度。	制定问题能力、分析问题能力：针对给定的任务进行需求分析
流程设计	教师根据学生导学问卷完成的情况，对题目进行分析，帮助学生理清程序的运行过程。引导学生对流程图进行分析，并在电子导学问卷中进行不同流程图的区分，选择正确的流程图。这一环节主要是让学生进行抽象建模。	抽象建模能力：针对给定的简单任务，能够提取问题的基本特征，并用流程图画出完成任务的关键过程
指令教学	教师选择正确的流程图，并按照流程图的顺序进行程序脚本的编写，每进行一个步骤前都对学生进行提问，若学生不知运用哪些 Scratch 指令可实现功能时，教师介绍新指令的使用方法，并引导学生在一定的时间内自主完成程序的编写。通过老师的讲解，学生学会选择合适的程序进行脚本编写。	算法设计能力：运用基本算法设计解决问题的方案，能使用编程语言或其他数字化工具实现这一方案
评价改进	教师展示 2-3 份学生作品，让学生分析这几份作品中的不同之处，引导学生思考多种算法的可能性。最后让学生自主修改完善自己的作品。学生通过观看别人的作品，知道自身作品不足，从而进行思考并完成对作品的完善。	优化方案能力：对基于信息技术的问题解决方案，能够依据信息系统设计的普遍原则进行较全面的评估，并采用恰当的方法迭代优化解决方案
综合创新	教师进行分层教学。所有学生必须完成程序的基本功能，学有余力的学生可运用之前积累的程序编写经验尝试为程序增加功能如添加角色、改变赛跑的比赛方式等来丰富程序。	迁移解决方法能力：按照问题解决方案，选用适当的数字化工具或方法获取、组织、分析数据，并能迁移到其他相关问题的解决过程中
分享总结	学生将自己的作品发送至云端并为自己觉得优秀的作品投票，教师根据投票数展示得票	

	数最高者的作品，并请设计者对作品进行讲解。最后教师进行小结。通过观看其他人作品，吸收别人作品的精华，为自己现在或以后的作品积累经验。	

3.3.2. 评价设计

学生在整个 Scratch 程序设计的过程中经历了明确问题、分析问题、解决问题到完善问题，实践了计算思维的过程。

计算思维是隐性的能力，但学生在作品创作的过程以及最终成果可以体现计算思维水平，教师可通过作品的展示环节和作品的制作环节对学生相应的计算思维水平进行评价。

制定问题能力和分析问题能力以及抽象建模能力主要通过电子导学问卷的数据结果来表现。

通过问卷星的统计，电子导学问卷的答题情况统计如下表所示，因第一第二题为学生基本信息，故不加入数据统计。

表 4 电子导学问卷的答题情况统计

题号	内容	计算思维培养目标	得分率
3	小黄狗具有哪些功能	制定问题能力	0.90
7	小黑狗具有哪些功能		0.92
4	对“控制小黄狗移动”的过程提问	分析问题能力	0.85
5	对“控制小黑狗移动”的过程提问		0.86
8	对“触发失败提示和胜利的提示”的过程提问		0.85
9	对“两只狗来回移动”的过程提问		0.84
11	选择“控制小黄狗移动”的正确流程图		0.80
12	选择“控制小黑狗移动”的正确流程图		0.81
14	选择“触发失败提示和胜利的提示”的正确流程图		0.83
15	选择“两只狗来回移动”的正确流程图		0.82
6	选择“控制小黄狗移动”的正确流程图	抽象建模能力	0.78
10	选择“控制小黑狗移动”的正确流程图		0.75
13	选择“触发失败提示和胜利的提示”的正确流程图		0.70
16	选择“两只狗来回移动”的正确流程		0.80

算法设计能力、优化方案能力以及迁移解决能力主要通过学生的作品来表现。

表 5 作品创作要求

作品应实现的功能		计算思维培养要求
基本要求	按“红点”小黄狗、小黑狗按不同的速度同时移动	算法设计能力 优化方案能力
	小黄狗、小黑狗在碰到“红线”后停止	
	按“红点”小黄狗、小黑狗回到起始位置	
	按“红点”小黄狗、小黑狗来回跑三圈后停止	
要求以外的创意或功能		迁移解决方法能力

学生整体的评价以电子导学问卷和作品随机抽取评价的方式进行，结果如下：

表 6 计算思维评价维度具体表现

一级维度	二级维度	学生具体表现
制定问题	了解程序主要能够实现的功能	全部学生完成电子导学问卷，90%的学生知道范例程序有哪些功能，编程目标明确，制定问题能力较好。
	了解程序主要呈现的效果	
分析问题	正确回答程序中每个角色间的关联	80%学生对“用修改数值让不同角色有不同赛跑速度”的过程理解清晰，分析问题能力较好。
	正确回答哪些角色是需要编写脚本	
	正确回答程序运行的直观效果	
抽象建模	能用流程图对程序的功能实现过程进行绘制	70%学生选择正确的流程图，抽象建模能力较好。

算法设计	根据流程图自主选择合适的 Scratch 程序指令来实现程序的功能	随机抽取 10 名学生的录屏视频，视频显示所有学生都按照要求对程序进行仿编，成功仿编的有 7 位，占 70%。说明学生算法设计能力较好。
优化方案	能分析现有程序脚本设计的错误、漏洞	将这 10 位学生作品进行分析后发现，8 名学生尝试进行完成或改进程序功能，有 8 名学生成功完成或改进程序功能，占 80%，说明学生优化方案能力较好。
	能选择最佳解决问题的程序脚本设计	
迁移方法解决	能对程序功能进行丰富，增加更多的程序功能和效果。	对 10 名学生录屏视频进行观察，发现有 8 名学生运用之前学过的东西，对程序的功能进行添加，且这 8 名学生都成功地完成了程序的新功能添加，占 80%，说明学生迁移方法解决能力较好。

4. 结论与探讨

Scratch 程序设计创设的情境“赛跑”取自生活情景。通过软件本身的卡通化形象等为计算思维的培养营造了一个丰富有趣的情境。计算思维贯穿《赛跑》课堂的全程，程序的编写过程分别锻炼了学生不同的计算思维能力内容。作为一个小学生计算思维培养的尝试，得出以下结论：

4.1. 小学生计算思维的评价

计算思维不仅是信息技术学科核心素养之一，同时也是每个人适应现代社会的必备思维方法。基于不同的研究视角，国内外学者对计算思维的定义及构成要素的分析不一。通过对现有计算思维构成要素的分析，结合 Scratch 程序设计项目的特点，提取计算思维的制定问题、分析问题、抽象建模、算法设计、优化方案、迁移方法解决六要素作为小学生计算思维培养的六个评价维度。这六个评价维度下的二级维度的设置以小学生这个年龄段直观、具象的特点为基准，应充分发挥小学生的观察能力，并在此基础上有意识地引导小学生将现实的具象物体以抽象的方式进行算法设计。

4.2. Scratch 程序活动的设计方法

在斯滕伯格思维培育理论和建构主义理论的指导下，对计算思维要素进行分析，作为活动目标及活动评价的依据；提出 Scratch 程序活动的设计方法，包括面向思维发展的设计原则、基本内容设计、教学流程设计、评价设计。以《赛跑》为例，通过实验研究过程中学生完成电子导学问卷的分析数据以及随机抽取的 10 名学生的录屏视频表明，面向计算思维发展的 Scratch 程序活动设计与实施小学生计算思维的提升有显著效果。

4.3. 基于小学生计算思维培养的小学 Scratch 程序设计教学流程模型

研究设计了基于小学生计算思维培养的小学 Scratch 程序设计教学流程模型：新课导入-流程设计-指令教学-评价改进-综合创新-分享总结。为一线信息技术教师开展 Scratch 程序设计教学提供了借鉴，也为计算思维的培养与发展的落地提供了经验。

5. 参考文献

王荣良（2012）。计算思维：一种新的学科思维方式。**中国信息技术教育，06**，9-13。

王婷婷、王丹力、路璐、何亮、王宏安和戴国忠（2013）。面向儿童的图形化编程语言和工具。**计算机辅助设计与图形学学报，04**，584-591。

中华人民共和国教育部（2018）。**普通高中信息技术课程标准（2017 年版）**。北京：人民教育出版社 。

李锋和王吉庆（2013）。计算思维：信息技术课程的一种内在价值。**中国电化教育，08**，19-23。

任友群、隋丰蔚和李锋（2016）。数字土著何以可能？——也谈计算思维进入中小学信息技术教育的必要性和可能性。**中国电化教育，01**，2-8。

牟琴、谭良和吴长城（2011）。基于计算思维的网络自主学习模式的研究。**电化教育研究，05**，53-60。

ISTE & CSTA. (2011). *Computational Thinking in K-12 Education: Teacher Resources, Second Edition*. Retrieved November 23, 2015, from http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf,2015-11-23.

Selby, C., & Woollard, J. (2013). Computational Thinking: The Developing Definition. *Proceedings of Special Interest Group on Computer Science Education 2014*. Atlanta.

Perkovic, L., Settle, A., Hwang, S., & Jones, A. (2010). Framework for Computational Thinking across the Curriculum. *Proceedings of 15th Conference on Innovation and Technology in Computer Science Education*. New York: ACM, 123-127.

Computational Thinking and Teacher Development

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Employing Computational Thinking in General Teacher Education

Stefan SEEGERER[1*], Ralf ROMEIKE[2*]
[1] Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
[2] Freie Universität Berlin, Germany
stefan.seegerer@fau.de, ralf.romeike@fu-berlin.de

ABSTRACT
The current political discussion about the digital transformation in Germany's educational context is primarily concerned with the use of digital media in schools. However, all disciplines and their related school subjects are significantly affected by digitalization – as can be seen e.g. with the effects of simulation or data analysis. This results in new topics, methods or strategies that schools must also deal with in the future. In consequence, teachers of any subject require Computational Thinking competencies and Computer Science knowledge, not only for the efficient and effective use of digital technology but also to understand and apply the new topics, methods, and approaches. In this paper, the design and implementation of a new course for teacher education in Germany is presented. With a theme revolving around digital transformation, this course aims at preparing pre-service teachers for teaching in the 21st century. Design principles and content selection are based on an analysis of similar courses and requirements arising from digitalization and its effect on the disciplines. First results show that students have gained a clearer understanding of how digitalization influences their subjects and teaching in general. Additionally, they report feeling more confident in employing aspects of digital education.

KEYWORDS
digitalization, teacher education, computational thinking

1. INTRODUCTION
Digitalization facilitates storing, processing and searching massive amounts of data. This accelerates fundamental changes affecting our daily lives. In addition, all disciplines are significantly affected: Digitization leads to far-reaching changes, for example in the collection of data (scope, quantity, and quality) (Grillenberger & Romeike, 2014) or their mostly automatic processing (simulation, collection, and evaluation of large amounts of data) (Hey, Tansley, & Tolle, 2009). With simulation or data analysis often being considered the third and fourth pillars of science (e.g. Riedel et al., 2008), new approaches to knowledge generation evolved. This results in new topics, methods and strategies for all disciplines (e.g. Hegedus et al., 2017).

These new topics, methods and strategies are becoming increasingly relevant for schools, as well. The application of computing technology across subjects requires an understanding of "how, when and where computers and other digital tools can help us solve problems" (Barr, Harrison & Conery, 2011). To embed this application, teachers require Computer Science competencies and knowledge, not only for the efficient and effective use of digital technology, but also to understand and apply the new topics, methods, and approaches while teaching.

In this paper, the questions to be answered are: which design principles can guide the creation of a course aiming at conveying necessary competences, and what content is relevant for teachers of all subjects. Therefore, various individual research results are outlined and intertwined. Building upon these results, five blended-learning modules for pre-service teachers in Germany are presented. The first iteration is then evaluated with regards to the design principles.

2. COURSE BACKGROUND AND RELATED WORK
Digitalization is having an increasing impact on K12 education. Recently, in political contexts, this has been frequently summarized under the term "digital education". While this term is often associated with the use of technology, there are also approaches that focus more on Computational Thinking. CT describes ways of tackling a problem like a computer scientist would (Wing, 2006). Even though there is no agreed-upon definition, there is a mutual agreement that it includes a set of skills to think about problems and their solutions (Kalelioglu, Gülbahar, & Kukul, 2016). It is about applying skills, such as abstraction or decomposition, so that the solution can be effectively carried out by a computer. Initial approaches to incorporate CT into K12 teaching relate primarily to science education, e.g. by learning science through simulation and modeling (Basu et al, 2013). But the approaches and achievements of the digital humanities show that teaching in other subjects could be heavily influenced by digitalization as well.

Consequently, a number of sources discuss the embedding of CT across the curriculum (cf. Barr & Stephenson, 2011, Kale et al., 2018). Examples include curve fitting or doing a linguistic analysis of sentences (e.g. Barr & Stephenson, 2011). The important role "computer (and possibly its abstraction) can play in enhancing the learning process and improving achievement of K–12 students in STEM and other courses" (Cooper, Pérez & Rainey, 2010) is emphasized in the model of Computational Learning (CL). The model combines theories of learning and the computer's ability to handle complexity and visualize results appropriately to improve understanding as well as learning. While this concept includes the usage of computers, e.g. for simulations and modeling, the authors stress that the model explicitly excludes non-cognitive uses of technology such as clickers or blogs.

Computer science skills and knowledge support teachers in engaging their students in CT and CL. While universities started to offer CS courses for a lot of non-CS student groups, there are still many pre-service teacher trainings focusing mainly on improving information and communication technology (ICT) skills (e.g. Goktas, Yildirim, & Yildirim, 2009). Despite the importance of Computational Thinking in all disciplines and throughout education, existing approaches explicitly embedding CT-related competencies in non-CS teacher training are still rare. Existing CS courses for non-CS teachers tend to focus, for example, on everyday phenomena that they analyze and illuminate from a computer science point of view (Müller, Frommer, & Humbert, 2013), on algorithmic concepts (Yadav et al., 2011), or on emphasizing Computational Thinking in the context of Information & Media Literacy (Dengel & Heuer, 2018).

3. ANALYSIS OF COMPUTATIONAL THINKING SKILLS FOR GENERAL TEACHER EDUCATION

In order to develop a CT curriculum in general teacher training, we have drawn on several sources. This section presents how these individual research results were merged and used as the basis for the course. Building upon the results, this section outlines how digitalization affects subjects and illustrates design principles and selection of content.

3.1. How Digitalization Affects Subjects

While new topics or methods, such as simulation, can be found in the context of science teacher training (Smetana, & Bell, 2012), the discussion of these new methods and topics for other subjects, such as the humanities, is fairly new. In order to understand which CT competencies future teachers need, looking at how digitalization affects the regarding subjects and related disciplines delivers helpful insights. These effects result from a survey among educational researchers for subject-matter teaching and learning and have also been discussed in working groups with the participants (Seegerer & Romeike, 2018c) In the following, we will highlight effects of digitalization in the disciplines and discuss how the effects affect school education.

3.1.1. Tools

Software or hardware tools supporting cognitive processes are an integral part of daily work, e.g. computer algebra systems in mathematics or geoinformation systems in geography. This offers possibilities for teachers as well. For example, the use of specialized databases (e.g. regarding life on Earth) offers great opportunities for biology teaching.

3.1.2. Methods and Ways of Thinking

The effects of digitalization in the disciplines are not limited to the mere use of digital media or tools, but fundamentally change cognitive processes (Berman et al., 2018). The methods used in the disciplines are particularly important for determining Computer Thinking competencies. Methods, like modeling, simulation, or data analysis are becoming increasingly relevant across disciplines (e.g. Ananiadou & McNaught, 2006). These methods also allow for a more student-centric access to topics in school. An example from geography is a lesson in which students use GPS data tracked on their daily ways to school. The resulting data is then evaluated collaboratively regarding the meaningfulness of the data.

3.1.3. Topics

A review of publications in various fields of research shows that not only methods are changing. In addition, the digital transformation also leads to “new” topics (e.g. Bharadwaj, et al, 2013) in the disciplines. The survey with experts in subject-matter teaching and learning shows that this also affects topics taught in school. We refer to a new topic when phenomena or artifacts of the digital world can be explored and explained from concepts, ideas or foundations of the subject. Economics, for example, discuss digital business models in contrast to traditional ones. As students are confronted with the implications of these business models when interacting with digital services, such topics also become relevant for schools.

This has consequences for teacher training. Since most teachers have not received a comprehensive general education in CS, they need a foundation of CS knowledge and CT skills. For example, discussing digital business models requires teachers to have a basic understanding of underlying concepts, such as knowing how to attribute meaning to data. However, it is necessary to address the implications of the effects of digitalization on the subjects: Not only do they need to apply digital tools and computational methods in the classroom, they also need to teach new ways of thinking and problem solving and convey new topics. Although courses such as bioinformatics are becoming more and more popular, they have not yet found their way into the course programs of teachers.

3.2. Design Principles

When CS courses for non-majors are developed, many challenges need to be faced. The students do not necessarily take the course because of an interest in CS, but may take it as a preparation for teaching in a digitalized world. The question is how a course should be designed to give prospective teachers the best possible experience. Based on the effects of the digital transformation on the disciplines, we decided to always embed CS in a theme revolving around digitalization: Topics are discussed in the context of digital transformation to show relevance for the students. Building on the survey results, the constraint of this being the only CT course for nearly all students, and the need to adequately prepare pre-service teachers for a digital age, several key design principles emerged:

3.2.1. Discuss Digital Transformation on a CS Basis

As described in the previous section, Computer science knowledge and skills are necessary to understand the digital transformation and advances in the disciplines. In addition, the digital transformation also leads to new issues becoming relevant for being discussed in class (cf. Brinda & Diethelm, 2017). Due to their history of education, many prospective teachers still do not have a proper foundation in Computer Science. A course must, therefore, be based on fundamental ideas of computer science that underlie the digital transformation and also highlight impacts on society. In order to be a good fit for prospective teachers of all disciplines, a corresponding course must also take the different levels of prior knowledge into account.

3.2.2. Show Relevance to Students of All Disciplines

German teacher education has a strong focus on the individual subjects. Typically, teachers study two subjects in-depth, which they will later teach. As seen in the section before, digitalization leads to new topics, methods and tools in their related disciplines. Thus, it also affects subject teaching. Although the course is generally be designed for students of all subjects and school types, transferability of the topics into the subjects must be ensured. Thus, cross-references to other disciplines should be emphasized repeatedly. This is addressed via different exercises, ideas and examples provided throughout the course. In some cases, students are required to research or discuss additional examples transfer to their subjects. In other cases, we provided a list of ideas they can elaborate on.

3.2.3. *Profit from CT Inside and Outside the Classroom*

CT skills, such as solving a problem using a simulation, help students in their disciplines. As prospective teachers, these students are also in the position to teach CT. By integrating CT and CL into the course, pre-service teachers learn how to profit from computing in their disciplines, as well as inside and outside the classroom. To this end, the course needs corresponding tasks including programming, as it exposes students to CT (Lye, & Koh, 2014). While hands-on activities enable teachers to learn CT, the connection to teaching should be highlighted whenever possible.

3.2.4. *Show the Importance of Skills Like Collaboration or Creativity*

Many experts believe that traditional topics will be less important in the future. Instead, teachers should be enabled to promote creativity, collaboration and critical thinking in the context of digital education (e.g. Bellanca, 2010). These approaches are also part of CT models, such as the CAS Model (Computing at School, 2014). This mindset should also be reflected in course design: Exercises allow students to develop their own ideas or to create personal artifacts. Many tasks are designed to encourage collaboration among students (e.g. via pair programming or collaborative writing). The course material also provides suggestions on how to employ these skills in class. Teachers should be made aware of the importance of these skills so they can foster them in their teaching.

3.3. *Content Selection*

Computer science knowledge and skills are necessary to understand changes and advances in the students' disciplines. To allow for the discussion of this digital transformation on a basis of CS, a well-founded selection of course content is important. When designing a CS course for students who will only take one such course, questions arise about which aspects of Computer Science are essential for everyone. When it comes to selecting the topics, we, as computer science educators, have different possible ways of selecting the content. Course designers rely on idea catalogs, such as the CS Principles (Astrachan & Briggs, 2012). Another idea catalog is the non-exhaustive list of Big Ideas behind K12 Computing Education (Bell, Tymann & Yehudai, 2018), including data representation, algorithms, complexity, comput-ability, digital representations, time-dependent operations, digital systems are designed to serve humans needs, and communication protocols. Others, with a slightly different focus, include the Principles of Computing (Denning, 2013) or the Fundamental Ideas of CS (Schwill, 1994).

These approaches are important to emphasize which topics make up computer science as a subject. But they do not necessarily help to make statements about which aspects are crucial as a basis for every non-CS teacher. Non-CS teachers need very specific knowledge about the basics of digitization and the effects on their subjects in a short amount of time. Therefore, our approach is two-fold. Aspects covered in university level non-major courses can form the basis for analyzing which aspects of CS can support pre-service teachers. In a study we analyzed the most common topics relevant in CS courses for non-majors (Seegerer & Romeike, 2018a, Seegerer & Romeike, 2018b). While the rising number of non-major CS courses may serve as an indicator for identifying key competences of CS considered important in the context of a digital transformation, the effects of digitalization in the subjects need to be considered as well. Accordingly, content was selected not only based on other CS courses for non-majors, but also based on tools, new methods and topics relevant across disciplines identified through literature, and the study with subject-matter teaching and learning experts.

Accordingly, we included basics about the digital representation of data, programming and algorithms. Due to the heterogeneity in prior knowledge of students, we saw a need to include computer systems and networks as well. Cryptography and limits of computing supplement the collection of topics. These topics are interweaved with other topics that gained importance in all related disciplines, namely simulations and different uses of data, such as data analysis.

4. CURRICULUM

The course is designed as a blended learning course. In the beginning, a face-to-face meeting takes place. This meeting is all about tinkering: students explore programming in Snap! using a MakeyMakey Board (Beginner's Mind Collective & Shaw, 2012). They build a banana piano and remix an existing video game to be controlled with the MakeyMakey. Most of the course is then done in an online learning environment. The online learning environment is designed to be engaging for students. Therefore, students often get the chance to gain hands-on experience by using the programming environment Snap!, or multiple interactives, e.g. adapted from the CS Field Guide (csfieldguide.org.nz.) The course ends with a short project phase, where students develop their own ideas in small groups and present them in the last session.

The course consists of 12 modules in total, 5 of them focusing primarily on CT (see Table 1). The CT part includes a module focusing on the impact of digitalization on disciplines, society and education as a whole, one module focusing on more technical aspects exploring the layers of abstraction inside computing systems, one on algorithms and one module concentrating on data analytics and simulations, respectively. Other modules related to CS focus on creativity or information gathering. They discuss different ways of fostering creativity in modern classrooms and emphasize the importance of creative thinking and working.

Table 1. Curriculum Contents and connection to CS.

Title of module	CS content
Fundamentals of Digitalization	Representation of data, programming
Secure use of computer systems and networks in professional teaching and learning	Computer systems, networks, cryptography, limits of computing
Solving subject-specific tasks with algorithms	Algorithms, programming
From data to domain knowledge	Working with data, programming
Modeling and simulations in domain-specific contexts	Modeling, simulations, programming

4.1. Fundamentals of Digitalization

This module marks the introduction into CT and takes place in week two. As the entire course is dedicated to the theme "Teaching in a digitalized world", it starts off by identifying and explaining the differences between digital and analog representations. Photography is used as an example for transforming analog information into digital data. Subsequently, binary numbers are introduced as the basis of digital data storage of a computer. Programming languages are introduced as the language of digitalization. To engage students in problem-solving with programming, students create turtle art in Snap!. They are guided by tutorials introducing sequences and loops. The last task is to remix an existing project, allowing for creative expression.

4.2. Secure Use of Computer Systems and Networks in Professional Teaching and Learning

This module showcases the structure and operating principles of computers and computer networks. Students investigate the layers of abstraction in modern computer systems. They explore package transport via the internet, and discuss why antivirus programs are subject to fundamental limitations. Part of this module was also a section about cryptography, basic techniques, and its importance for society.

4.3. Solving Subject-specific Tasks with Algorithms

The module on algorithms concentrates on how automation is used in, or affects students' subject areas. Students learn how to use conditional statements and apply it to creating a learning app with Snap!. Afterward, the term algorithm is introduced. Students explore what an algorithm is (and what it is not). Finally, students have to implement a certain algorithm. They can choose from different algorithms they already know from their subjects, or that have relevance in their discipline, e.g. calculate square roots, or determine word frequencies.

4.4. From Data to Domain Knowledge

Visualizations can be helpful for the interpretation of data, the generation of hypotheses, or generally for the clarification of correlations. Unfortunately, data is typically not accessible in such a visualized form. Usually, the data is stored as numbers or strings coded in databases, spreadsheets or certain files. One of the most important applications of computers in science is, therefore, the evaluation and analysis of digital data. Accordingly, one module concentrates on analyzing, visualizing, and interpreting data. Teachers engage in sense making of data using representations and learn the importance of being critical when interpreting data, not jumping to conclusions, and always deciding on the basis of the available data. In addition, students are encouraged to evaluate the importance of data analysis in their disciplines. In one exercise, for example, they use Snap! to fathom London's cholera outbreak from 1854.

4.5. Modeling and Simulations in Domain-specific Contexts

The weather, the solar system or particle motion are just some examples of complex science concepts that may require a mental model for learners. Modeling is particularly helpful in such contexts, as it allows us to focus on specific aspects of a real-world phenomenon (Weintrop et al., 2016). This includes the use of abstraction, as unnecessary details are left out. Prospective teachers learn about the importance of modeling for learners. In addition, they learn when and how to incorporate modeling and simulation into teaching. After using existing models that can be manipulated using parameters, the students learn about different aspects of a model itself, and use concept mapping as an easy for applying modeling in the classroom. Afterward, we present them with different exercises where they learn to model and simulate different contexts, including a predator-prey, a market, and an epidemic situation with Snap!.

5. EVALUATION

In the following, insights of an initial qualitative analysis for the first iteration of the modules will be outlined. These were provided in writing by five students at the end of each module. The questions included aspects like perceived personal benefit, personal effort, or perceived relevance. As most of the course was online, we were curious as to how certain topics can be taught. Fortunately, the online format was well received: "Interesting and 'different'. It is more diversified, clearly more casually arranged than other online modules, and loosened up by exercises and practice."

5.1. Discuss Digital Transformation on a CS Basis

As the students should gain a foundation in CS, we were interested in how well the content was received. One of the biggest challenges was the time constraint. Executing a meaningful module combining both typical lecture and lab content within the possible workload for one week proved difficult. Indeed, students have recognized and commented on a heavy workload for some modules. In order to let individual modules not become too extensive, the content of selected modules was streamlined after the first weeks. Still, students rated all content as personally meaningful.

5.2. *Show Relevance to Students of All Disciplines*

The course design aims to teach students how digitalization affects their teaching and their corresponding disciplines. Therefore, we tried to emphasize the relevance to the different subjects and disciplines on as many occasions as possible. Initial reactions show that the course can deliver on the promises. According to the students, they feel more confident to incorporate digital tools and new methods or topics in the classroom after completing the course. "I particularly liked the connection drawn to the subjects, where I've become aware of the specific topics that can be addressed in the classroom using Snap!."

The sections on data analysis and simulations, in particular, did particularly well in this aspect. The biggest potential for improvement is found in the very general section on Computer systems and networks.

5.3. *Profit from CT Inside and Outside the Classroom*

As with most Computer Science courses for non-majors, there are specific aspects that are crucial. Programming tasks in non-major courses are often considered problematic (e.g. (Banerjee & Kawash, 2009)), despite being important for CT. Nevertheless, in many university courses for students of other disciplines, programming is an essential component that is also addressed at a very early stage (e.g. (Garcia, Harvey & Barnes, 2015)). As pre-service teachers of all disciplines should profit from CT inside and outside the classroom, the question arises how an early reference to programming tasks affects the motivation and self-efficacy expectations of the students. We had similar constraints when designing the materials. We found that while students feel overwhelmed by the possibilities, the programming environment Snap! offers, they saw it as a fresh take on the topic and engagement in programming. The first contact with Snap! has been described as: "Websites like Snap! are great to try out things."

Nevertheless, there have been some entry barriers for students. As in most courses for non-CS majors that involve some kind of programming exercises, students face certain difficulties. Most difficulties have been solved by face-to-face meetings or via online communication. Especially the face-to-face meeting after students have worked on the material on their own has been regarded as very helpful. However, coding exercises still represent a hurdle, especially for online courses. We have tried to strike a balance between unrestricted and guided tasks. Students noted missing hints in some places, so they tended to use a trial-and-error approach by trying out different blocks. Regarding students' feedback, for future iterations, the inclusion of additional video material concerning programming is planned.

5.4. *Show the Importance of Skills Like Collaboration or Creativity*

Approaches like tinkering, collaborating and creating, are an important part of CT. Students apply these approaches themselves to learn how to think computationally, and to understand the importance of CT approaches for their own students. Additional readings provided students with the theoretical background and strategies for the application of these approaches in class. Students emphasized their understanding of the importance of these approaches, e.g. creativity, in the evaluation: "I became aware of the relation of creativity and teaching and will pay more attention to foster the creativity of students in my lessons."

6. CONCLUSION AND FUTURE WORK

In summary, we have presented a course for pre-service teachers at a German university, which introduces computing using the background of digital transformation. In the future, teacher training will have to deal increasingly with new topics and the effects of digitalization. Therefore, it will not be a question of whether but of how to incorporate Computer Science and Computational Thinking education into general teacher training. The paper describes design principles that can support the creation of similar courses and provides an example course. Its effort is to form a foundation for all teachers regardless of their subject and/or school type. Building on this foundation, teachers can discuss different aspects of digitalization within their subjects. So, while teachers might compare and debate data usage by different digital business models in economics, they may create or adapt a simulation for science teaching. By combining different aspects of Computational Thinking, we help prepare teachers to incorporate computing inside and outside of the classroom. Computational Thinking and Learning practices can enhance learning for students in all subjects. The course was initially piloted with a small number of students from a limited range of subjects. In the future, it will be opened for further teaching positions. Thus, we plan on integrating additional examples from a diverse range of disciplines. The material is now also adapted for in-service teacher training.

7. REFERENCES

Ananiadou, S., & McNaught, J. (2006). *Text Mining for Biology and Biomedicine*. London: Artech House.

Astrachan, O., & Briggs, A. (2012). The CS Principles Project. *ACM Inroads, 3*(2), 38-42.

Banerjee, S., & Kawash, J. (2009). Re-thinking Computer Literacy in Post-secondary Education. *Proceedings of the 14th Western Canadian Conference on Computing Education.* New York: ACM, 17-21.

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.

Basu, S., Dickes, A., Kinnebrew, J., Sengupta, P., & Biswas, G. (2013). CTSiM: A Computational Thinking Environment for Learning Science through Simulation and Modeling. *Proceedings of the CSEDU.* Aachen, 369-378.

Beginner's Mind Collective, & Shaw, D. (2012). Makey Makey: Improvising Tangible and Nature-based User Interfaces. *Proceedings of TEI '12, Stephen N. Spencer (Ed.).* New York: ACM, 367-370.

Bell, T., Tymann, P., & Yehudai, A. (2018). The Big Ideas in Computer Science for K-12 Curricula. *Bulletin of EATCS, 1*(124).

Bellanca, J. (Ed.). (2010). *21st century skills: Rethinking How Students Learn.* Bloomington: Solution Tree Press.

Berman, F., Rutenbar, R., Hailpern, B., Christensen, H., Davidson, S., Estrin, D., Franklin, M., Martonosi, M., Raghavan, P., Stodden, V., & Szalay, A. (2018). Realizing the Potential of Data Science. *Communications of the ACM, 61*(4), 67-72.

Bharadwaj, A., El Sawy, O., Pavlou, P., & Venkatraman, N. (2013). Digital Business Strategy: Toward a Next Generation of Insights. *MIS Quarterly*, *37*(2), 471-482.

Brinda T., & Diethelm, I. (2017). Education in the Digital Networked World. *Proceedings of WCCE 2017.* Cham: Springer, 653-657.

Computing at School (2014). *Barefoot: Computational Thinking*. Retrieved January 3, 2019, from http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking

Cooper, S., Pérez, L., & Rainey, D. (2010). K-12 Computational Learning. *Communications of the ACM, 53*(11), 27-29.

Denning, P. (2003). Great Principles of Computing. *Communications of the ACM, 46*(11), 15-20.

K-12 Computer Science Framework Steering Committee (2016). *K-12 Computer Science Framework. Technical Report*. New York: ACM.

Garcia, D., Harvey, B., & Barnes, T. (2015). The Beauty and Joy of Computing. *ACM Inroads, 6*(4), 71-79.

Goktas, Y., Yildirim, S., & Yildirim, Z. (2009). Main Barriers and Possible Enablers of ICTs Integration into Pre-service Teacher Education Programs. *Journal of Educational Technology & Society*, *12*(1), 193-204.

Grillenberger, A., & Romeike, R. (2014). Big data – Challenges for Computer Science Education. *Proceedings of International Conference on Informatics in Schools.* Cham: Springer, 29-40.

Hegedus, S., Laborde, C., Brady, C., Dalton, S., Siller, H. S., Tabach, M., & Moreno-Armella, L. (2017). Uses of Technology in Upper Secondary Mathematics Education. *Proceedings of ICME-13*. Cham: Springer, 1-36.

Hey, T., Tansley, S., & Tolle, K. (2009). *The Fourth Paradigm: Data-intensive Scientific Discovery (vol. 1)*. Redmond: Microsoft Research.

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational What? Relating Computational Thinking to Teaching. *TechTrends*, *62*(6), 574-584.

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking based on a Systematic Research Review. *Baltic Journal of Modern Computing, 4*(3), 583.

Lye, S., & Koh, J. (2014). Review on Teaching and Learning of Computational Thinking through Programming: What is Next for K-12? *Computers in Human Behavior, 41*, 51-61.

Müller, D., Frommer, A., & Humbert, L. (2013). Informatik im Alltag– Durchblicken statt Rumklicken [CS in Everyday Life - Understanding Instead of Clicking Around]. *Proceedings of HDI '13*, 98-104.

Riedel, M., Streit, A., Wolf, F., Lippert, T., & Kranzlmüller, D. (2008). Classification of Different Approaches for E-science Applications in Next Generation Computing Infrastructures. *Proceedings of 4th Inter-national Conference on eScience.* US: IEEE, 198-205.

Schwill, A. (1994). Fundamental Ideas of Computer Science. *Bulletin-European Association for Theoretical Computer Science*, *53*, 274-274.

Seegerer, S., & Romeike, R. (2018a). Goals, Topics and Tools of Computer Science for All University or College Courses. *Proceedings of SIGCSE '18, 1090.* New York: ACM.

Seegerer, S., & Romeike, R. (2018b). Was Jeder über Informatik Lernen Sollte – Eine Analyse Von Hochschulkursen Für Studierende Anderer Fachrichtungen [What Everyone Should Learn About CS - An Analysis of University Courses for Students of Other Disciplines]. *Proceedings of HDI '18.* Potsdam: Universitätsverlag.

Seegerer, S., & Romeike, R. (2018c). Computer Science as a Fundamental Competence for Teachers in Other Disciplines. *Proceedings of WiPSCE '18, 149-150.* New York: ACM.

Smetana, L., & Bell, R. (2012). Computer Simulations to Support Science Instruction and Learning: A Critical Review of the Literature. *International Journal of Science Education*, *34*(9), 1337-1370.

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.

Yadav, A. Zhou, N., Mayfield, C. Hambrusch, S., & Korb, J. (2011). Introducing Computational Thinking in Education Courses. *Proceedings of SIGCSE '11.* New York: ACM, 465-470.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Complexity of Teacher Knowledge, Skills and Beliefs about Software Education: Narratives of Korean Teachers

Da-hyeon RYOO[1], Hyo-jeong SO[2*], Dongsim KIM[3]
[1] Department of Educational Technology, Ewha Womans University, Korea
[2] Graduate School of Education, Hanshin University, Korea
rdahyun@ewha.ac.kr, hyojeongso@ewha.ac.kr, southpaw61@hs.ac.kr

ABSTRACT

Under the nation-wide initiative of software (SW) education in Korea, this study attempts to unpack the complexity of knowledge, skills and beliefs that Korean teachers have about SW education in school. Semi-structured interviews were conducted with four types of teachers who were classified by knowledge/skill levels (surplus vs. shortage) and pedagogical beliefs (positive vs. negative). The teacher narratives revealed five main themes: (a) differences in espoused beliefs in SW education, (b) fear about the new knowledge and skill domain, (c) recognition of the need to turn to learner-centered pedagogy, (d) questioning knowledge transfer-centered training and (e) school leadership and support as a catalyst of change. The paper concludes with some suggestions for improving teacher professional development experiences in SW education.

KEYWORDS

software education, teacher professional development, computational thinking (CT)

1. INTRODUCTION

From 2019, software (SW) education (has been mandated in the elementary and middle school curricula in Korea as a way to improve learners' competence demanded in the future society. Consistent with the government policy, this study defines SW education as a curriculum that aims to improve creative problem-solving competence and logical thinking of basic principles of software rather than acquiring coding skills. Since the development of learners' Computational Thinking (CT) is central to SW education, the school curricula have been developed to include various topics and levels of CT-related knowledge and skills, including the understanding of CT, unplugged activities, Educational Programming Languages (EPL), physical computing and text-based programming. Teachers' readiness in both pedagogical and technological aspects is critical for the success of such new initiatives (Rosenlund & Hansen, 2018; Saeli, Perrenet, Jochems, & Zwaneveld, 2011). However, Han (2018) argues that most teachers in Korea have no or little experiences with programming, and hence have faced the difficulty of acquiring necessary knowledge and skills in CT.

It has been widely acknowledged that the introduction of any policy initiatives in education is unlikely to be successful if teachers' epistemological beliefs are not consistent with the initiative's vision and mission (Ertmer, 2005). To understand how Korean teachers perceive the nation-wide SW education initiative, this study attempts to unpack the complexity of teacher's pedagogical beliefs, knowledge and skills in CT through the narratives of four teachers covering the various spectrum of CT knowledge/skills and pedagogical beliefs.

2. THEORETICAL BACKGROUND

2.1. *The Current Status of Teacher Training and Infrastructure of SW Education*

Teacher readiness and supportive infrastructure are essential for SW education to be successfully implemented in Korean schools. Firstly, about the issue of teacher readiness, in 2018, the Korean Ministry of Education began implementing the "SW Education Enhancement Support Project for Teacher Training Universities" to prepare elementary school teachers for the new initiative. However, the current situation in school is not so conducive for teachers to implement SW education. Since most teachers have not received formal training about programing and CT during their pre-service teacher education, the demand for in-service teacher professional development (TPD) is immense. The situation is similar in middle schools. In 2016, a total of 1,354 middle school teachers have been trained in SW education, which is on average less than one teacher per school. Among 15,800 elementary teachers, 2,540 teachers have received offline training and 1,800 teachers have received online training. However, the TPD programs have not reached out to the wider teacher population since only interested teachers participate in the programs, and many teachers lack willingness and motivation for such TPD.

Next, the school infrastructure is still not ready and conducive for teachers to fully implement SW education in classrooms. The percentages of old computers (exceeding 5 years) are 38.6% in elementary schools and 43.7% in middle schools. The reality is that teachers have to implement SW education without sufficient teaching and learning materials to be integrated into lessons, coupled with the lack of wireless networks and computers.

2.2. *Teacher's Knowledge, Belief and Practices*

Teachers should acquire necessary knowledge and skills in computer science to teach CT in the classroom (Saeli et al., 2011). Calderhead (1996) argues that while teacher beliefs generally refer to "suppositions, commitments, and ideologies", knowledge refers to "factual propositions and understandings" (p. 715). Since teacher beliefs are formed based on previous experiences over time, it is difficult to change or reverse belief systems (Ertmer, 2005).

Windschitl (2002) suggested that fundamental changes in learning occur when environmental contexts are considered concomitantly with teachers' knowledge and beliefs. This implies that teacher beliefs are closely related to their knowledge and practices. Belief is a personal identity and a source of motivation (Ertmer, 2005) and provides a guide for teachers to decide what to offer in the classroom context (Biesta, Priestley, & Robinson, 2015). In particular, teacher's belief system, as a filter, influences teachers' acceptance of new knowledge and technology to promote pedagogical practices (Rosenlund & Hansen, 2018).

About teaching with technology, Judson (2006) studied 32 teachers and found that student-centered teaching methods were effective in the technology-integrated class. Pierson (2001) interviewed four teachers according to the level of technology use and teaching abilities (adequate vs. exemplary) and found that Technological Pedagogical Content Knowledge (TPACK) was essential for effective technology integration. It was also found that beliefs about teaching methods were closely related to accepting and practicing new curricula. Rosenlund & Hansen (2018) proposed teacher competence to teach in the networked world, emphasizing teachers' beliefs and knowledge about using technology and distributed resources.

2.3. Computational Thinking (CT) and Teachers

Recently, there have been attempts to provide guidelines for teachers to improve students' CT. For example, CSTA & ISTE (2014) gathered opinions from about 700 computer science teachers, researchers, and practitioners and provided an operational definition of CT and the CT Leadership Toolkit 2.0 for teachers. Angeli et al. (2016) proposed a curriculum framework that allows teachers to teach CT in K-6, which include five components of CT: abstraction, generalization, decomposition, algorithms (sequencing, flow of control), and debugging. The framework was used to design courses that provide problem-solving tasks based on real-life issues from the simple to the complex level. Students can develop CT in relation to their daily life and authentic situations through a holistic design approach that deals with the complexity and interconnections across separate elements.

Recent studies show that teachers must employ innovative practices with technology to produce positive learning outcomes. For instance, Angeli et al. (2016) noted the role of TPACK for CT because technology plays an important role in teaching CT practices. Some studies pointed out that teachers' expertise cannot be developed solely by acquiring technical skills. So & Kim (2009) found the conflict between in-use TPACK and espoused TPACK. In their study, teachers had difficulty with designing pedagogically-sound technology-integrated lessons (in-use TPACK) although they had acquired necessary knowledge and strong beliefs about the use of technology in teaching and learning (espoused TPACK).

Such a conflict between in-use TPACK and espoused TPACK is generally caused when teachers cannot translate tacit knowledge into explicit knowledge in classroom practices. Tacit knowledge is a certain type of knowledge that [one] cannot tell (Polanyi, 1966). One way to overcome this conflict is to change teacher beliefs while acquiring explicit knowledge (So & Kim, 2009). Beliefs are one of the important sources of teacher changes, and belief systems are formed gradually with experiences over time (Rosenlund & Hansen, 2018). Similarly, teachers construct new knowledge and understanding based on their prior knowledge and belief. In particular, technology integration is closely related to pedagogical beliefs, methods and content knowledge (Angeli et al., 2016). Therefore, teachers need to develop relevant knowledge, skills, and beliefs in order to innovate teaching with new technology (Mishra & Koehler, 2006). Despite this, little is known about the complexity of teacher beliefs, knowledge, and skills in CT.

3. METHOD

3.1. Participants

To examine Korean teachers' beliefs, knowledge and skills about CT under the nation-wide movement of SW education, this qualitative study adapted the matrix of teacher profiles (see Table 3) proposed by Rosenlund & Hansen (2018) to identify the various spectrum of teachers' CT-related knowledge, skills and pedagogical beliefs. We used a snow-balling method to recruit study participants through the teacher recommendation in the teacher community of SW education. Based on data from the pre-study interviews, we intentionally selected four teachers, one for each cell of the matrix. Table 1 presents the demographic information about four teachers (three in elementary, one in middle school) who participated in this study.

Table 1. Demographic information

Name*	Gender / Age	School Level	Experience with CT	Teaching experience
JW (Type I)	Male 36	Elementary	Selected as an excellent SW teacher	14 years
CH (Type II)	Male 44	Middle	Selected as an excellent SW teacher	20 years
SM (Type III)	Male 35	Elementary	3 years of information technology related work	3 years
YM (Type IV)	Female 44	Elementary	No experience	17 years

※ pseudonyms are used for data anonymity

3.2. Data Collection

This study used interviews as a main source of data. The researchers (authors of this paper) conducted in-depth face-to-face interviews with individual participants in July 2018. Each participant was interviewed approximately for 90-120 minutes. Before the interview, the researcher informed the participants about the purpose of the research and the audio recordings for data collection. With the consent of the participants, the interviews were recorded and transcribed for analysis.

The researchers conducted the semi-structured interviews with the guiding questions (Table 2) that examine various domains of teacher experiences along the trajectory that affected the development of their CT competency, including distributed knowledge resources used for personal learning and related knowledge acquisition, teaching experiences, and personal beliefs about the policy initiative.

Table 2. Guiding interview questions

Domain	Questions
Initial experience	What was the main factor that affected your decision to become a teacher?
Teacher training	Have you voluntarily participated in various training courses related to SW education?
Knowledge resources	Where do you usually find the necessary information about SW education?
Teaching experience	Have you ever conducted a SW class in school? If yes, how?
Interest in SW education	Do you have any particular interest in SW education? Why or why not?
Beliefs about SW education policy	Why do you think it is necessary for SW education to be taught in school?

3.3. Analytical Framework and Method

We analyzed the interview narratives to identify the complexity of teachers' CT-related knowledge, skills and beliefs. In this study, teacher's pedagogical beliefs that determine good teaching practices in CT are viewed as an important factor in determining the quality of SW education. Previous studies suggest that teachers' pedagogical beliefs and technology are closely related. Belief systems provide teachers with a professional guide to make decisions in class (Chen, Looi, & Chen, 2009). In particular, Tondeur, Van Braak, Ertmer, & Ottenbreit-Leftwich (2017) reported that teachers with traditional teacher-centered beliefs tended to use technology less frequently in class.

Concerning teachers' beliefs, knowledge, and skills in teaching, Rosenlund & Hansen (2018) proposed the matrix of teacher profiles, which classifies teachers according to ICT skills (surplus vs. shortage) and approach to use ICT in school (positive vs. negative). Based on their framework, we constructed the matrix of teacher profiles in CT with two factors: (a) the level of teachers' pedagogical beliefs about SW education (positive vs. negative); (b) the level of knowledge and skills to teach SW education (surplus vs. shortage). As shown in Table 3, each participant was mapped to the matrix, based on their levels of knowledge/skills and pedagogical belief revealed in the pre-interview. This matrix leads to four types of teacher profiles:

- *Type Ⅰ - The innovative teacher (surplus of knowledge/skills, positive belief)*: A teacher who shows high quality teaching with good knowledge/skills and deep understanding about the necessity of SW.
- *Type Ⅱ - The serious teacher (surplus of knowledge/skills, negative belief)*: A teacher who has sufficient knowledge/skills for SW education but is not fully aware of the necessity of SW education.
- *Type Ⅲ - The insecure teacher (shortage of knowledge/skills, positive belief)*: A teacher who has a positive view about SW education, but lacks technical skills and pedagogical knowledge.
- *Type Ⅳ - The confused teacher (shortage of knowledge/skills, negative belief)*: A teacher who lacks knowledge and skills with a low self-confidence about using technology and does not fully understand the necessity of SW education.

Table 3. Matrix of teacher profiles

	Pedagogical belief with SW education	
	Positive	Negative
Surplus of knowledge & skills	JW (Type I) The innovative teacher	CH (Type II) The serious teacher
Shortage of knowledge & skills	SM (Type Ⅲ) The insecure teacher	YM (Type Ⅳ) The confused teacher

4. RESULTS

The analysis of interview narratives revealed that the participants had different knowledge/skill levels, perceptions, and experiences depending on the type. We present the main findings according to the five themes emerged.

4.1. Theme 1: Differences in Espoused Beliefs about SW Education

While SW education in school has been mandated by the government, not all participants appeared to readily accept the necessity of this initiative. Regardless of CT skills, there were some differences in the degree of espoused beliefs according to the teacher type. Type I and III teachers with positive beliefs argued that education should be expanded with SW to promote the national growth of talented people. On the other hand, Type II and IV teachers, who both have negative pedagogical beliefs, questioned the necessity of SW education, particularly the argument that CT is necessary for all students.

Researcher: *Why do you think SW education is necessary?*

JW (Type I): *I think that in the future society technology will be in all aspects of everyday life, like mobiles that are deeply used in our lives now. The idea underlying SW education is that children learn the language that communicates with a computer. Just as you cannot communicate with foreigners if you do not speak English, I think if you do not have CT, you will not be able to get a job in the future society.*

CH (Type II): *I personally question that SW education is required for all children. Not everyone needs to be a computer programmer. Each person has a different personality and ability. In Korea, there are vocational high schools for cultivating technical experts. So I think it would be enough if a SW curriculum is provided to the students in vocational high schools. In fact, I have been* *selected as an excellent SW teacher and have been running SW education as instructed by the Ministry of Education. That's not to say, I think SW education is necessary for all students.*

YM (Type IV): *I think children should be happy. From elementary schools, Korean students have to learn too many curriculum contents. As the SW education is mandated, I feel that the academic burden is increasing with the private tutoring.*

4.2. Theme 2: Fear about the New Knowledge and Skill Domain

While the participants are in their 30s and 40s, they felt that CT concepts are new and showed some degree of concern about learning the new knowledge and skill domain (i.e., computer science). In particular, such fears and concerns were obvious in Type III and IV teachers who lack relevant technical skills and experience of CT. Since both types of teachers were more interested in other subject areas, the mandated teacher PD programs were a burden rather than a learning opportunity to develop an expertise in CT:

Researcher: *In what way, was the SW education class difficult to teach?*

SM (Type III): *When I took the SW education teacher training, it was too difficult to understand the concepts. I am interested in Korean language and physical education, so I am involved in related field training. Because SW education is difficult to understand, it is a burden for me to participate in the teacher PD programs.*

YM (Type IV): *I am very interested in cultural education and student life guidance and am also involved in the related training. SW education with digital devices and computer programming is a burden for me.*

Such fears often led to the unwillingness of implementing CT classes. Type III and IV teachers shared that some students are better than them in terms of technical skills, and were afraid of answering student questions.

Researcher: *If you need to teach a SW class now, would you mind?*

SM (Type III): *I am afraid that students will ask questions that I do not know. How can I teach a class without knowing about CT? I'd rather have a professional teacher who can teach better.*

YM (Type IV): *I will not take it. Kids are better than me. I think I will lead the class introduction, but the rest will be done by the students.*

4.3. Theme 3: Turn to the Learner-centered Pedagogy

While the participants commonly showed some degree of fears and concerns about SW education, it was interesting to find that they also sought solutions to overcome such difficulties through accepting the learner-centered pedagogy. They recognized that since the present levels of many teachers' knowledge and skills are limited, it is important for teachers to co-teach with or receive help from experts, even engaging students with good levels of CT in teaching. Hence, the positive side is that it also provided them with opportunities to realize and accept the changing role of teachers as a facilitator, designer, and critical investigator (Laurillard, 2012), as shown in the following narratives:

Researcher: *If you need to teach a SW class now, what could you do?*

CH (Type II): *How do I know everything? I think the same is true for the world where students grow up. If you have something new like CT, you have to cope with it and learn it. I am just looking for an expert who can help me.*

YM (Type IV): *Students can help me out in class. And I think this can give them a chance to grow even more. I do not think there are any teachers who just want to pass on knowledge.*

4.4. Theme 4: Questioning Knowledge Transfer-centered Training

Even Type I and Type II teachers who were selected as excellent SW teachers expressed the concern about the efficacy of the current teacher professional development in SW education. Similarly, Type II and IV teachers who lack CT knowledge/skills questioned the efficacy of knowledge transfer-centered training since the present CT training tends to be theory-heavy rather than practice-oriented, and to be imposed as a top-down requirement to be completed within a short timeframe. Such negative perceptions about the efficacy of the current teacher PD programs appeared to be associated with the lack of teachers' empathy about the necessity and value of SW education in school:

Researcher: *Are there enough opportunities for teacher training as SW education will be introduced from the next year?*

JW (Type I): *I would be more helpful if you let me know how to proceed with actual lessons rather than delivering the theory to the center.*

CH (Type II): *As an excellent SW teacher, I participated in SW training programs as a lecturer. Teachers do not listen well because the topic is difficult.*

4.5. Theme 5: School Leadership and Support as a Catalyst of Change

The participants shared different stories about how the school leadership and support affected their perception and experiences with CT. The Type I teacher was in charge of after-school programs in SW education and was engaged in various activities to help students develop CT through the linkage of industry-academia partnership programs. He cited the support and leadership from the school principal as a main driving force for his efforts in SW education.

Researcher: *What is the driving force of your success in SW education?*

JW (Type I): *Our principal gives a full support if you say yes. Last time, I thought I would not have a wireless Internet connection due to the budget. The principal brought budget as much as possible, so I was able to set up the Internet connection with one of my colleagues.*

On the other hand, the Type II teacher did not receive sufficient support from the school principal, coupled with the budget constraint to build the learning environment necessary for the implementation of SW education.

> Researcher: *If you need to teach a SW class now, would you mind?*
>
> CH (Type II): *I kept telling the principal, but he could not be convinced! I borrowed hamster robots from the neighbor school. Last year, there was no budget, so I could not buy any tablets. I still had more than half of old computers in class, so it was hard to teach SW education. I did not have any related budget this year, so I had no idea how to operate the SW education program.*

Currently, most schools in Korea do not have school-wide wireless networks. The participants commonly pointed out that building the wireless Internet environment in school is essential for SW education. This also necessitates the relevant administrative and financial support to build SW education-conducive learning environments.

5. DISCUSSION AND CONCLUSION

In this section, we discuss the implications of the main findings and propose some suggestions for teacher PD related to CT in SW education.

First, Themes 1 and 2 indicate that teachers tend to experience conflicts in espoused beliefs about CT, and such conflicts are often caused by the psychological factor, which is the fear and concern about entering a new knowledge domain. Teachers tend to perceive that SW education and CT are highly skilled domains that teachers cannot easily enter or reach the basic mastery of the domain. Lowering or minimizing such psychological fears through emotional support needs to be considered prior to delivering training about cognitive knowledge and technological skills (Shin, Kim, & Jeong, 2019). One promising approach for emotional support is to engage teachers to learn from each other in a community-based setting. For instance, this community-based approach has been implemented in UK where computing education is mandatory for students in elementary and secondary schools. Teachers are connected with master teachers in computer science. The nationwide training network helps leading schools with specialized staff in computer science to help teachers in nearby schools (Oliver & Venville, 2011).

Second, it was encouraging to see that emotional conflicts function as an impetus to accept the changing role of teachers and students and even to modify teaching practices (Theme 3). Teachers tend to show the belief that teaching CT and SW requires acknowledging that teachers are not a sole source of knowledge, but a facilitator in collaborative knowledge building (Rosenlund & Hansen, 2018). By necessity, teachers turn to the learner-centered pedagogy. This story indicates that obstacles eventually become an opportunity to change.

However, we also want to emphasize that such a positive impetus to the learner-centered pedagogy is unlikely to be sustainable if supporting structures are not provided (Theme 5). The existing literature on teachers and technology integration suggests that changing teachers' pedagogical beliefs are the most important and challenging task (Ertmer, 2005; So & Kim, 2009). In particular, Type IV teacher sought ways to overcome weaknesses through self-reflection. She understood the characteristics of digital native students who are highly receptive with new technology and provided activities to engage students to find and explore topics among peers. In addition, teachers need to receive support from the school leadership in both infrastructural challenges (e.g., the purchase and deployment of software and hardware equipment) and pedagogical innovations (e.g., initiating new teaching methods).

Ertmer (2005) argues that teachers tend to face first-order barriers such as issues with resources, time, training and support when they attempt to integrate technology into teaching. She also suggests that teachers face second-order barriers that are associated with teachers' beliefs, typically rooted in their underlying beliefs about how teaching and learning should work. While the existing literature suggests that the second-order barriers are more important and critical than the first-order barriers, we argue that for CT and SW education, attempts should be made to tackle both types of barriers concurrently. SW education as a subject matter requires a certain level of technological infrastructure to be established. That is, unlike other subject matters where the use of technology is optional, technology in SW education is both a goal and a tool (Angeli et al., 2016). This was evident in Types III and IV teachers who did not develop enough technological knowledge during pre-service teacher education, thereby having fears about SW education. Understanding that achieving technology integration is a multi-faceted effort for teachers, this study suggests that policy support to overcome first-order barriers need to be systematically developed, in parallel with the redesign and provision of teacher PD programs that aim to tackle second-order barriers.

Third, related to the design of teacher PD (Theme 4), for teachers to grow into Type I with positive beliefs and good knowledge/skills, it is essential to provide systematic supports where teachers can understand and empathize the vision and goal underlying the national initiative of SW education. In particular, most teachers in Korea do not have sufficient prior experiences with SW education and their learning opportunities are limited to the government-led training programs. However, current teacher training programs are mostly conducted in the form of formal lectures in 1-2 hours. Such short-term training formats are unlikely to meet the needs of teachers, especially Types III and IV teachers. Teacher training needs to be redesigned considering the level of school environments and teachers' readiness. Some recent studies on TPACK suggest the potential of combining CT and design thinking through problem-solving activities in real contexts (Shin, Kim, & Jeong, 2019). In addition, teacher PD programs need to help teachers better understand the core concept of CT, to show various applications in class, and to provide systematic support that enables teachers to apply and develop competencies in teaching CT (Liu, 2011).

The limitations of this study are as follows. First, this study was conducted in the transition year prior to the nation-wide implementation of SW education. It is important for future research to continuously examine teacher perceptions, challenges and developmental trajectory from the implementation year of 2019. Second, since this study used the convenience sampling method to identify the participants and only four teachers were examined in the study, the generalizability of the research findings is limited and needs a further validation through other sources of data and larger samples of teachers. Despite these limitations, we believe that this study contributes to the existing CT literature by unpacking the perceptions and challenges faced by teachers under the situation where SW education is mandated.

6. REFERENCES

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.

Biesta, G., Priestley, M., & Robinson, S. (2015). The Role of Beliefs in Teacher Agency. *Teachers & Teaching*, *21*(6), 624-640.

Calderhead, J. (1996). Teachers: Beliefs and Knowledge. In *Handbook of Educational Psychology*. New York: Macmillan Library Reference, 709-725.

Chen, F. H., Looi, C. K., & Chen, W. (2009). Integrating Technology in the Classroom: A Visual Conceptualization of Teachers' Knowledge, Goals and Beliefs. *Journal of Computer Assisted Learning, 25*(5), 470-488.

Ertmer, P. A. (2005). Teacher Pedagogical Beliefs: The Final Frontier in Our Quest for Technology Integration? *Educational Technology Research & Development*, *53*(4), 25-39.

Han, Y. (2018). Analysis of Effectiveness of Programming Learning for Non-science Major Preliminary Teachers' Development of Computational Thinking. *Journal of the Korean Association of Information Education, 22*(1), 41-52.

Judson, E. (2006). How Teachers Integrate Technology and their Beliefs about Learning: Is There a Connection? *Journal of Technology & Teacher Education*, *14*(3), 581-597.

Laurillard, D. (2012). *Teaching as a Design Science: Building Pedagogical Patterns for Learning and Technology*. New York, NY: Routledge.

Liu, S. H. (2011). Factors Related to Pedagogical Beliefs of Teachers and Technology Integration. *Computers & Education*, *56*(4), 1012-1022.

Mishra, P., & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record, 108*(6), 1017-1054.

Oliver, M., & Venville, G. (2011). An Exploratory Case Study of Olympiad Students' Attitudes Towards and Passion for Science. *International Journal of Science Education, 33*, 2295-2322.

Pierson, M. E. (2001). Technology Integration Practice as a Function of Pedagogical Expertise. *Journal of Research on Computing in Education*, *33*(4), 413-430.

Polanyi, M. (1966). *The Tacit Dimension*. New York, NY: Doubleday.

Rosenlund, L. T., & Hansen, J. J. (2018). Teaching in a Networked World-skills, Knowledge and Beliefs. In *Designing for Learning in a Networked World* (pp. 81-101). London: Routledge.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education*, *10*(1), 73-88.

Shin, S. B., Kim, C., & Jeong, Y. S. (2018). Teacher Training Strategies for Improvement Technological Pedagogy Knowledge (TPK) Connected with Problem Solving. *Journal of the Korean Association of Information Education. 22*(1), 23-32.

So, H. J., & Kim, B. (2009). Learning about Problem Based Learning: Student Teachers Integrating Technology, Pedagogy and Content Knowledge. *Australasian Journal of Educational Technology, 25*(1), 101-116.

Tondeur, J., Van Braak, J., Ertmer, P. A., & Ottenbreit-Leftwich, A. (2017). Understanding the Relationship between Teachers' Pedagogical Beliefs and Technology Use in Education: A Systematic Review of Qualitative Evidence. *Educational Technology Research & Development, 65*(3), 555-575.

Windschitl, M. (2002). Framing Constructivism in Practice as the Negotiation of Dilemmas: An Analysis of the Conceptual, Pedagogical, Cultural, and Political Challenges Facing Teachers. *Review of Educational Research*, *72*(2), 131-175.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Model for Readiness Analysis of Schools Conducting Computational Education

Yu-lan HUANG[1], Ting-chia HSU[2*]
[12] National Taiwan Normal University, Taiwan
60771020h@ntnu.edu.tw, ckhsu@ntnu.edu.tw

ABSTRACT
The computational thinking education in K-12 has become an important issue so this study arranged a technology leadership lectures of computational thinking courses for the principals in primary and secondary schools. After the principals finished the computational thinking courses and completed the training, the principals needed to fill out a questionnaire to evaluate their schools' readiness in every aspect about conducing computational thinking education. Through the scale, this study developed a model analysis for the readiness of schools conducting computational thinking education, to explore the path relation of TPACK and its three aspects—technological knowledge (TK), pedagogical knowledge (PK) and content knowledge (CK). The other three parts of readiness are the hardware preparations of national compulsory education schools in the domain of technology education, the teaching resources management and preparation, and supports of leadership. Through this path analysis, this study systematically established the software and hardware preparation of computational thinking.

KEYWORDS
computational thinking, readiness, TPACK, path analysis

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

運算思維之學校準備度模型分析

黃友嵐[1]，許庭嘉[2*]
[12] 科技應用與人力資源發展學系，國立臺灣師範大學，臺灣
60771020h@ntnu.edu.tw，ckhsu@ntnu.edu.tw

摘要

K-12 運算思維教育已經成為一個重要議題，本研究舉辦校長科技領導講座講授運算思維課程，在校長經歷了運算思維課程並完成了培訓後，請校長填寫問卷來評估他們的學校各方面準備度，進一步探討落實運算思維教育時，各校在各層面所需要準備的工作進度。本研究最後透過量表，發展一個運算思維教學準備度模型，探討 TPACK 與其三個面向—科技知識（TK）、教學法知識（PK）以及內容知識（CK），與科技領域之國民義務教育的學校之硬體物件準備、教學資源管理準備與領導支持之間的路徑關係，希藉此路徑分析，以有系統的建立運算思維的軟硬體準備。

關鍵字

運算思維；準備度；TPACK；路徑分析

1. 研究背景

資訊科技是世界基礎建設的根本。在此社會背景，教育與生產或服務產業一樣，受到技術的影響（García-Peñalvo, 2018）。而運算思維（Computational thinking，CT）是能夠解決複雜且技術導向之社會問題的一種關鍵技能，學校可以幫助學生培養運算思維能力，進而導致教師將 CT 融入教學中的必要性（Kale et al., 2018）。運算思維作為 21 世紀學生的關鍵技能，也促成許多以運算思維為重點的課程計劃並將其嵌入 K-12 課綱中（Yadav, Hong, & Stephenson, 2016）。但在 K-12 課程中增加學生對運算思維的認知需要系統性的改變、教師參與和關鍵資源的開發與計算機科學教育界的合作（Barr & Stephenson, 2011），整體而言，運算思維的加入為 K-12 教育提供了新的方向（Kafai, 2016）。

CT 不僅適用於教師培訓，也適用於校長培訓，校長都知道 CT 是什麼，以及如何將其融入課程，以及這些課程的要求。自 2011 年以來，台灣的許多縣市都要求新任校長參加與技術和領導相關的培訓課程。預期校長會了解學校的設施和教師對執行技術相關的教學、管理和服務，並期望在最近的兩年中，讓他們了解與 CT 有關的相關問題。2015 年以色列將 CT 應用於教師教育，以克服教師在「計算機科學概論」課程中獲得專業知識的障礙。 K-12 教師意識到資源不足的學生可能遇到的困難。透過對職前教師或新任校長的教師教育，可以知道如何為他們的教師提供支持和幫助，以加強 CT 教育（Israel, Pearson, Tapia, Wherfel, & Reese, 2015）。

本研究以 Hsu（2018）發表之臺灣實施運算思維教育的準備情況研究量表作討論，該研究結合移動學習的準備情況調查表（Yu, Liu, & Huang, 2016）與 TPACK 的模型做修定，協助校長評估與描述技術領域的學校教師以及學校設備與教學內容等準備程度。

2. 文獻探討

2.1. 運算思維

運算思維是指用於解決計算機科學領域問題的基本概念和過程（Wing, 2006）。分解問題是 CT 的歷程之一，以便將大問題分成幾個較小的子問題，這稱為「問題分解」階段。然後，第二階段是識別數據表示或數據結構中的模式。換句話說，如果學生觀察到任何重複的數據或方法的呈現，他們可以識別他們的相似性，規律性或共通性。因此，當他們寫出解決方案步驟時，他們不需要花時間重複工作。另外一個重要的階段是通則化（Generalization）或抽象原則（Abstraction），使其成為一個公式或規則。學生必須嘗試對他們在上一步中找到的模式進行建模。在測試之後，他們識別並抽象出表示模型的關鍵或關鍵因素，以便在此步驟中解決問題。最後，他們在第四階段設計算法，確保它們包含所有步驟系統地解決問題。儘管 CT 不等於程式設計，但是視覺化的程式設計語言（如 Scratch，Blockly，mBlock，App Inventor 等）是輔助開發學生 CT 能力的好工具。因此， CT 被定義為「制定問題及其解決方案所涉及的思維過程，以便解決方案以可由信息處理代理有效執行的形式表示」（Cuny, Snyder, & Wing, 2010）。

2.2. 校長領導

教師領導力是學校進步的核心（Szeto & Cheng, 2018）。校長變革策略的實踐對學校的發展、改革與紮根會產生重大影響（Soini, Pietarinen, & Pyhältö, 2016）。變革型領導是教學領導的必要條件（Marks & Printy, 2003）。了解在學校中實現和維持教育成果的校長的領導角色和行為，為有抱負和服務的校長提供了更多差異化、上下文敏感的培訓和發展方面的支持（Cheon, Lee, Crooks, & Song, 2012）。轉型校長能夠緩解環境不確定性對學校組織健康的負面影響（Hameiri & Nir, 2016）。

2.3. 教學準備度

科技技術使知識的積累和相互作用日益累積，影響大部分社會的知識獲取、交流和傳播過程。與此同時，評估教育系統成功引入和實施電子學習計劃的能力—即電子學習準備，對於實現教育目標至關重要（Darab & Montazer, 2011）。在馬來西亞，為了讓大學生使用手機學習，開發基本準備度、技能準備度、心理上的準備度以及預算準備度等四個行動學習準備度的面向做探討（Hussin, Manap, Amir, & Krish, 2012）。而有效整合行動學習資源並在執行行動學習前完成充分準備，是學校創新並達到成功的第一步，行動學習準備度是作為學校實施行動學習前的自我檢核評估工具，並依此分析中小學行動學習準備度現況（Yu et al., 2016）。

2.4. TPACK

大量文獻表明，科技及教育是使用技術和課堂整合的決定因素。而教師的科技內容教學知識（Technological Pedagogical Content Knowledge, TPACK）有助於將科技應用於教育目的（Scherer, Tondeur, Siddiq & Baran, 2018）。TPACK 已被教師作為用技術有效教學的知識庫框架。該框架源於特定教育背景下的技術整合得益於內容、教學法和技術潛力的精心調整，因此希望將技術整合到教學實踐中的教師需要能夠勝任所有三個領域（Voogt, Fisser, Pareja Roblin, Tondeur & Van Braak, 2013）。技術知識、教學知識和內容知識都是職前教師 TPACK 的重要預測因素（Chai, Koh, & Tsai, 2010）。各機構應為跨學科的教師教育者提供有力的支持，並為課程採用連貫的技術框架（Nelson, Voithofer, & Cheng, 2019）。

3. 研究目的與問題

3.1. 研究目的

本研究旨在透過校長針對學校之自我評估，使用 Hsu（2018）量表作衡量，其中有硬體物件準備（Object readiness，OR）、教學資源管理準備（Instructional material resource readiness，IMRR）、領導支持（Leadership support，LS）、技術知識（Knowledge of technology，TK）、教學法知識（Knowledge of pedagogy，PK）、內容知識（Knowledge of content，CK）與 TPACK 衡量校長對運算思維課程的各項資源準備度，本研究將此命名為運算思維之學校教學準備模型。

3.2. 研究假設

教師在教育課程中納入 TPACK 通常是一個持續的變革過程。但要實現變革，必須考慮領導者在變革創新中的作用。領導者、院長和部門主管必須是這過程的成員。而創新、變革和教育領導者面臨的挑戰是將教師準備計劃轉變為可實踐的 TPACK 環境，並確定提供充沛的學習機會與支持，以激勵學校領導和教師接受變革過程。雖然挑戰來自將教師準備計劃轉變為實踐的 TPACK 環境，但領導力成為開發新方法的關鍵，這個問題必須解決包含內容、教學和技術在內的核心知識。為了完成這項任務，教師們面臨著將這些想法融入教育課程中，並不斷變化的過程相結合。但首先必須考量領導階層在進行變革的關鍵作用（Thomas, Herring, Redmond, & Smaldino, 2013）。而缺乏技術領導和技術整合計劃是在學校有效使用技術的重要障礙。Vatanartiran & Karadeniz（2015）研究 K12 教師在將技術整合到課堂中時所面臨的挑戰和需求。結果表明，教師將技術融入教學有三個主要問題：執行、基礎設施和教學。執行問題主要與管理和財務方面的挑戰有關；基礎設施問題包括技術和硬體方面的挑戰，而硬體的挑戰如電腦資訊設備、各式技術教室、桌遊或是機器人教材以及軟硬體解決方案顧問（Hsu, 2018）；教學問題包括與教學材料、學生準備情況和教師能力相關的挑戰，如國家編審之教科書應用能力、為運算思維課程編撰教材的能力以及籌畫課程教學計劃之競賽，且有信心教授技術領域的教材（Hsu, 2018）。綜合以上提出以下假設：

H1：領導支持（LS）會正向影響硬體物件準備（OR）與教學資源管理準備（IMRR）

隨著當代技術對教育系統的需求，教師應具備將資訊與通信科技（Information and Communication Technology， ICT）整合到未來教學的能力和技能。研究評估了整合課堂 ICT 機會以及與 TPACK 和 SAMR（Substitute, Augmentation, Modification, and Redefinition），發現的大多數挑戰都與缺乏基礎設施、改變的準備以及缺乏教學 ICT 應用的能力有關。而進一步的工作應側重於開展實驗研究設計，以解開現有的 ICT 使用限制（Kihoza, Zlotnikova, Bada & Kalegele, 2016）。

H2：硬體物件準備（OR）會正向影響技術知識（TK）、教學知識（PK）與內容知識（CK）

Agustin & Liliasari（2017）研究發現，職前科學教師的 TPACK 仍然很低，不足以在初中教授科學。此外，需要一些協助計劃來幫助職前科學教師準備在科技教學的能力。在此情況下，更複雜的同伴教學計劃和更好的購買是提高職前科學教師 TPACK 能力的必要條件。引入技術教學內容知識（TPCK）作為教學中有效技術整合的必要條件（Mishra & Koehler, 2006）。

H3：教學資源管理準備（IMRR）會正向影響技術知識（TK）、教學知識（PK）與內容知識（CK）

Koehler & Mishra（2009）提出一種技術整合的教師知識框架，稱為科技教學內容知識（Technological Pedagogical Content Knowledge，TPACK）。Koehler 以 Shulman（1986）的教學內容知識（Pedagogical Content Knowledge，PCK）構建為基礎，描述了教師知識的 TPACK 框架，為三個知識體系之間的複雜互動：內容，教育學和技術。故本研究的第四個假設為技術知識、教學知識與內容知識作為 TPACK 之基礎，會正向影響 TPACK 之運行。

H4：技術知識（TK）、教學知識（PK）與內容知識（CK）會正向影響 TPACK

4. 研究方法

4.1. 參與者

本研究之受試者來自台灣 65 所國小、國中校長。男性佔 37 名（56.9%）、女性 28 名（43.1%）；任職於國小的校長有 42 名（64.6%）、國中有 23 名（35.4%）；年資以 31 至 35 年 28 名（43.1%）最多，年資 26 至 30 年 16 名（24.6%）次之。

4.2. 測量工具

衡量方法是參考 Hsu（2018）之臺灣實施運算思維教育的準備情況研究量表進行修改與衡量。各題項衡量標準使用李克特五點量表，範圍從 1（非常不同意）到 5（非常同意）。

參與者必須回答他們學校 CT 教育準備情況的調查問卷，並表達他們在教師的 TPACK 中所感知的情況。有八個問卷中的量表。前四個是從行動學習的準備情況調查表（Yu et al., 2016）進行修訂，其中提到了折衷的電子學習準備，包括對象準備，軟件準備和領導支持（Darab & Montazer, 2011），以及提到基於計劃行為理論的高等教育行動學習準備模型（Cheon et al., 2012）。因此，對象準備，教學資源管理準備和領導支持是評估在學校實施某些內容的準備的重要尺度，例如電子學習，行動學習或 CT。因此，本研究使用準備問卷，修訂問卷中每個量表的 Cronbach 可靠性係數為.75，對象準備就緒，.84 為教學資源管理準備，.90 為領導支持。

在 TPACK 模型的框架中明確指出了教師的技術，教學和內容知識之間的關係（Mishra & Koehler, 2006）。因此，大量研究採用這種模型來評估教師的專業性或教師教育的有效性（Chai et al., 2010; Koehler, Mishra, & Yahya, 2007）。這個模型也被引入另一項研究中，供教師進行自我評估（Schmidt et al., 2009）。目前的研究還採用 TPACK 模型（Chai et al., 2010）為校長描述技術領域的學校教師。修訂後問卷中每個量表的 Cronbach 可靠性係數對於技術知識為.92，對於教育學知識為.93，對於內容知識為.95，對於整體 TPACK 為.98。

4.3. 實驗程序

舉辦為期兩天的國中小校長科技領導講座，該培訓共 18 個小時，主要目的為讓與會的國中小校長可以了解科技領域的需求，包括硬體的需求以及師資的需求，使校長可藉由研習比對目前學校的情況。本研究花費了 9 個小時透過可視化編程工具體驗與數學結合的 CT 課程，並在剩餘的 9 個小時，帶領校長參訪基礎設施已經建立的學校，且透過培訓課程，介紹 K-12 在技術領域的要求。在校長們經歷了 CT 課程並完成了培訓後，填寫調查問卷以評估他們的學校。問卷的一部分是根據行動學習的準備情況進行修訂的，另一部分是 TPACK（即技術，教學和內容知識）模型。

4.4. 數據分析

本研究針對修定後的量表進行信度分析，針對準備度模型做 PLS-SEM 檢定分析。而偏最小平方法（Partial Least Square，PLS）使用於結構方程模型（Structural equation modeling，SEM），稱為 PLS-SEM。Ringle, Wende, and Becker（2015）改良 SmartPLS 2.0 的缺點並增加許多新功能，發展了 SmartPLS 3.0。本研究使用 SmartPLS 3.2.7 作為分析工具，並參考蕭文龍（2018）之統計書籍做為操作依據。

本研究係針對各項測量模型之參數進行估計，以檢定各個變項與構念的信度與效度。在收斂效度方面，J. Hair, Anderson, Tatham, and Black （1998）提出必須考量個別項目的信度（Individual Item Reliability）、潛在變項組成信度（Composite Reliability，CR）與潛在變項的平均變異萃取（Average Variance Extracted，AVE）等三項指標，若此三項指標均符合，方能表示本研究具收斂效度。Hulland（1999）建議因素負荷量應該都在 0.7 以上，方能表示測量指標具有良好信度，但題項 OR4、OR5 與 IMRR1 之因素負荷量分別為 0.598、0.635 與 0.667 小於 0.7，為避免信度問題，故將此三個變數刪除。Bagozzi & Yi（1988）建議 CR 值大於 0.8，方能表示構面具有良好的內部一致性。而 Fornell & Larcker（1981）建議 0.5 為 AVE 值之臨界標準做為收斂效度之衡量基準，若 AVE 值皆大於 0.5 表示具收斂效度，但原始量表之人力資源準備（Human resources readiness，HRR）的 AVE 值 0. 485 <0.5，不具收斂效度（Discriminate validity），故將此構念刪除。Cronbach's alpha（α）標準的下限為 0.7，若 α 值大於 0.7 則具有信度（Hair, Black, Babin, & Anderson, 2010）。顯著性部分，在 PLS-SEM 模式中，當 t 值>1.96，表示已達到 α 值為 0.05 的顯著水準以*表示；當 t 值>2.58，表示已達到 α 值為 0.01 的顯著水準以**表示；當 t 值>3.29，表示已達到 α 值為 0.001 的顯著水準以***表示（Hair, Ringle, & Sarstedt, 2011）。

5. 研究結果

SmartPLS 提供使用 PLS-SEM 檢驗模型契合度（Model Fit）的方法，而標準化均方根殘差值（Standardized Root Mean Square Residual, SRMR）定義為觀察到的相關性與模型隱含相關矩陣之間的差異，小於 0.10 或 0.08 的值（Hu & Bentler, 1998）被認為是一個很好的選擇。Henseler（2014）引入 SRMR 作為 PLS-SEM 的適配度衡量標準，可避免模型錯誤。當 SRMR 值愈小，表示模型配適度愈好，Hu & Bentler（1999）認為數值低於 0.08 就算是模式配適度佳。本研究 Saturated Model 之 SRMR 值為 0.067 < 0.08，具有良好的模式適配度。

5.1. 收斂效度

本研究有硬體物件準備（OR）、教學資源管理準備（IMRR）、領導支持（LS）、技術知識（TK）、教學法知識（PK）、內容知識（CK）與 TPACK 等七個潛在變數，我們整理輸出報表結果如表 1。

表 1 因素負荷量, T-value, Cronbach’s Alpha, rho_A, CR, AVE

構面	問項	因素負荷量	T-value	Cronbach's α	rho_A	CR	AVE
OR	OR1	0.808	13.128	0.762	0.780	0.863	0.678
	OR2	0.881	30.378				
	OR3	0.778	9.915				
IMRR	IMRR1	0.867	25.225	0.838	0.879	0.901	0.753
	IMRR2	0.819	10.991				
	IMRR3	0.914	43.190				
LS	LS1	0.818	14.085	0.904	0.909	0.928	0.721
	LS2	0.818	19.639				
	LS3	0.874	22.498				
	LS4	0.858	24.859				
	LS5	0.876	27.817				
TK	TK1	0.839	14.011	0.920	0.928	0.943	0.807
	TK2	0.909	27.118				
	TK3	0.930	39.658				
	TK4	0.913	37.041				
PK	PK1	0.856	10.200	0.932	0.941	0.952	0.831
	PK2	0.882	13.210				
	PK3	0.953	48.995				
	PK4	0.953	57.42				
CK	CK1	0.945	31.802	0.951	0.954	0.968	0.910
	CK2	0.946	24.638				
	CK3	0.972	64.099				
TPACK	TPACK1	0.961	44.550	0.979	0.980	0.984	0.940
	TPACK2	0.981	86.912				
	TPACK3	0.960	34.830				
	TPACK4	0.976	63.014				

由表 1 可知本研究所有觀察變項之因素負荷值皆大於 0.7，表示測量指標具有良好信度。CR 值皆大於 0.8，表示構面具有良好的內部一致性。而本研究七個構面的 AVE 值皆大於 0.5，表示具收斂效度。

5.2. 區別效度

由表 2 可知各構面 AVE 值皆大於構念間共項變異值，表示本研究構面潛在變項的平均變異數抽取量之平方根值大於相關係數值，故顯示各構念應為不同的構念，具有區別效度。

表 2 Fronell-Larcker Criterion

	1	2	3	4	5	6	7
CK	0.954						
LS	0.548	0.849					
OR	0.470	0.692	0.824				
PK	0.882	0.587	0.485	0.912			
TK	0.520	0.588	0.540	0.499	0.898		
IMRR	0.386	0.642	0.549	0.488	0.537	0.868	
TPACK	0.757	0.469	0.433	0.775	0.466	0.437	0.970

5.3. 路徑係數與解釋力因果關係圖

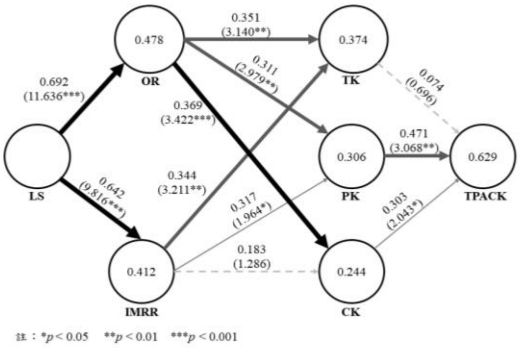


圖 1 路徑係數與解釋力因果關係圖

由研究模式的因果關係圖可知，領導支持對硬體物件準備（t = 11.636, p < 0.001）與教學資源管理準備（t = 9.816, p < 0.001），兩者皆呈顯著正相關，而估計值分別為 0.692 與 0.642，故假設一（H1）成立；硬體物件準備對技術知識（t = 3.140, p < 0.01）、教學知識（t = 2.979, p < 0.01）與內容知識（t = 3.422, p < 0.001），三者皆呈顯著正相關，估計值分別為 0.351、0.311 與 0.369，故假設二（H2）成立；教學資源管理準備對技術知識（t = 3.211, p < 0.01）、教學知識（t = 1.964, p < 0.05）、內容知識（t = 1.286, p > 0.05），其中內容知識不顯著而技術知識與教學知識呈顯著正相關，估計值分別為 0.344、0.317 與 0.183，故假設三（H3）部分成立；TPACK 對技術知識（t = 0.696, p > 0.05）、教學知識（t = 3.068, p < 0.01）與內容知識（t = 2.043, p < 0.05），教學知識與內容知識對 TPACK 呈顯著正相關，但技術知識對 TPACK 不顯著，估計值分別為 0.074、0.471 與 0.303，故假設四部份成立。

解釋力可由研究模式的因果關係圖得知，領導支持對硬體物件準備的解釋力為 47.8%，領導支持對教學資源管理準備的解釋力為 41.2%，硬體物件準備與教學資源管理準備對技術知識、教學知識與內容知識的解釋力分別為 37.4%、30.6%與 24.4%，而整體而言各潛在變項對 TPACK 的解釋力為 62.9%，顯示模式解釋潛在變項的程度良好。

6. 討論

6.1. 研究結果討論

本研究旨在提供 CT 學校準備度評估模型，使學校領導者能夠藉此模型了解現任學校目前之硬體、軟體等 CT 教學準備度。由圖 1 的路徑係數與解釋力因果關係圖可知，領導支持會正向影響硬體物件準備與教學資源管理準備，林明地（2000）提出校長教學領導的具體作為可以歸納為六大領域，包括巡視教室與校園、協助教師在職進修或專業成長與發展、表達較高的期望，提高師生表現的標準、了解學生學習情形、實踐行政支持教學的理念、建立良好的教學環境，以及建立楷模等。而當學校管理層支持和鼓勵教學以及技術領域學習的願景、政策或計劃，並且建立在技術方面具有優異教學表現的學生獎勵制度，重視參與 DIY 活動和比賽的學生以及鼓勵教師和學生參加機器人或程式設計比賽的背景下，學校會投入更多的資訊設備和創客及科技等教室，與邀請熟悉硬體和軟體的技術專家，並堅強學校的基礎設施，使科技領域教師能在課程中順暢使用教科書，並且有能力編制課程的教材，進而促使教師願意參加課程教學計劃的競賽，以將科技領域的 12 年國民教育付諸實踐。

所有教育工作者都知道，教學是一項交織著多種專業知識的實踐，教學需要教師應用不同的複雜知識結構案例、背景和方法來協助（Mishra, Spiro, & Feltovich, 1996; Spiro & Jehng, 1990）。而 TPACK 各個因子的相互作用，使得無論是在理論上還是在實踐中，都成功產生了用於教學的整合技術類型之靈活知識（Koehler, Mishra, & Cain, 2013）。當科技教師熟悉各項教學資源時，即可使用這些設備與資源輕鬆學習新技術並且解決技術問題。而教師擁有技術技能並適當使用這些技術時，就能同時培養教師使用運算思維工具或軟體來解決問題。本研究預期透過這樣學習歷程之教師可培養出適性的教學方法，根據學生目前的學習或不理解的方式調整教學，並且知道如何評估學生在課堂上的表現。項目準備（OR）也會使教師有各種方式和策略來發展他們對計算思維的理解，讓教師對運算思維有足夠的了解，並可以像專業運算思維專家一樣思考，充實教師的內容知識，但教學資源管理準備則無法產生此功效，是因為內容知識比較多是跟教師本身的背景有關係，不會受到教學資源是否豐厚的影響他對內容知識的認識，Matherson（2014）提到，現今仍有許多資深教師，即在 2005 年之前畢業的老師，沒有足夠的技術知識、技能和經驗來教學生，因為他們沒有沉浸在技術語言中，也沒有教過技術。1999 年，Kent 和 McNergney 報告說，儘管對技術的重視程度越來越高，但只有 15％的美國教師接受專業技術發展培訓。在幾年後的研究，儘管現有技術有所增加，但資深教師接受技術培訓的比例並沒有顯著增加，仍然低於 24％（Sawchuk, 2010）。所以假設三（H3）教學資源管理準備至內容知識這條線是不顯著的。

科技內容教學知識是由技術知識、教學知識以及內容知識所建構而成（Koehler & Mishra, 2009），教學知識（PK）是關於教學和學習的過程和實踐或方法的深刻

知識，以及它如何包括整體教育目的、價值觀和目標等。這是一種通用的知識形式，涉及學生學習、課堂管理、課程計劃制定和實施等所有問題。它包括有關在課堂上使用的技術或方法的知識、目標受眾的性質以及評估學生理解的策略。內容知識（CK）是關於要學習或教授的實際主題的知識。包括對特定領域內的中心事實、概念、理論和程序的了解，組織和聯繫思想的解釋框架的知識以及證據和證據規則的知識（Shulman, 1986）。技術知識（TK）就技術而言，這包括作業系統和電腦硬體的知識，以及使用文書處理軟體（如文字處理程序、電子表單、瀏覽器和電子郵件）的能力。但 TPACK 中的技術知識的含義缺乏明確性的定義（Graham, 2011）。Koehler 和 Mishra 沒有區分傳統與現在技術類型包含的知識（Mishra & Koehler, 2006；Technology, 2008）。定義和限制技術知識如何被感知的範圍對於框架的清晰度是重要的（Graham, 2011）。故，相較於通則性的教學知識、內容知識，本研究並未給予技術知識明確的定義，以致模式顯示不顯著。若後續明確定義之後應可優化此模型。

6.2. 研究限制與建議

本研究之樣本限制有二，其一為研究對象皆為國中、小學校領導人，而非實際任課教師。但是這符合科技領導講座研習的需求，因為研習初衷是為讓校長了解，不論是硬體的準備度、老師的準備度或是教材的準備度對學校要實施科技領域教育都具重要性，這與本研究之研究目的相符。另一個限制為本研究樣本數量較小，在使用拔靴法統計時容易產生偏誤（Tong & Brennan, 2007）。

臺灣從 2019 年 8 月開始，所有中學學生都需要接觸運算思維能力。但本研究之調查時間在新的國民教育上路前一至兩年實施，具有時間限制，故本研究模型僅能作為近幾年或是其他尚未引入運算思維教育之國家參考。

本研究模型之數據來源為臺灣運算思維課程教育準備現況，未必適合已運行運算思維學程之國家，或是同樣正在籌劃運算思維學程但不同文化之國家。

7. 致謝

本研究感謝科技部研究計畫編號: MOST 105-2628-S-003-002-MY3 與 MOST 107-2511-H-003-031 補助。

8. 參考文獻

蕭文龍（2018）。**統計分析入門與應用--SPSS 中文版+SmartPLS 3（PLS_SEM）第二版**。臺北：碁峰資訊。

林明地（2000）。校長教學領導實際：一所國小的參與觀察。**教育研究集刊，44**，143-172。

Agustin, R. R., & Liliasari, L. (2017). Pre-service Science Teachers’Readiness to Integrate Technology (An Exploration Toward Tpack in Preliminary Practical Context). *Jurnal Pengajaran MIPA, 21*(2), 191-196.

Bagozzi, R. P., & Yi, Y. (1988). On the Evaluation of Structural Equation Models. *Academy of Marketing Science. Journal, 16*(1), 74-94.

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.

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.

Cuny, J., Snyder, L., & Wing, J. M. (2010). *Demystifying Computational Thinking for Non-computer Scientists*. Retrieved November 17, 2010, from http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.

Darab, B., & Montazer, G. A. (2011). An Eclectic Model for Assessing E-learning Readiness in the Iranian Universities. *Computers & Education, 56*(3), 900-910.

Fornell, C. G., & Larcker, D. F. (1981). Evaluating Structural Equation Models with Unobservable Variables and Measurement Error. *Journal of Marketing Research, 18*(1), 39-50.

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

Graham, C. R. (2011). Theoretical Considerations for Understanding Technological Pedagogical Content Knowledge (TPACK). *Computers & Education, 57*(3), 1953-1960.

Hair, J., Anderson, R., Tatham, R., & Black, W. (1998). *Multivariate Data Analysis*. New York: Macmillan.

Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2010). *Multivariate Data Analysis* (Vol. (7th ed.)). Englewood Cliffs: Prentice Hall.

Hair, J. F., Ringle, C. M., & Sarstedt, M. (2011). PLS-SEM: Indeed a Silver Bullet. *Journal of Marketing Theory and Practice, 19*(2), 139-151.

Hameiri, L., & Nir, A. (2016). Perceived Uncertainty and Organizational Health in Public Schools: The Mediating Effect of School Principals’ Transformational Leadership Style. *International Journal of Educational Management, 30*(6), 771-790.

Henseler, J., Dijkstra, T. K., Sarstedt, M., Ringle, C. M., Diamantopoulos, A., Straub, D. W., Ketchen, D. J., Hair, J. F., Hult, G. T. M., & Calantone, R. J. (2014). Common Beliefs and Reality about Partial Least Squares: Comments on Rönkkö & Evermann (2013), *Organizational Research Methods, 17*(2), 182-209.

Hsu, T. C. (2018). *The Readiness of Computational Thinking Education in Taiwan: Perspectives from the K-12 Principals in 2017.* Paper Presented at the International Conference on Computational Thinking Education 2018, Hong Kong, China.

Hu, L. T., & Bentler, P. M. (1999). Cutoff Criteria for Fit Indexes in Covariance Structure Analysis: Conventional Criteria Versus New Alternatives. *Structural Equation Modeling: A Multidisciplinary Journal, 6*(1), 1-55.

Hu, L. T., & Bentler, P. M. (1998). Fit Indices in Covariance Structure Modeling: Sensitivity to Underparameterized Model Misspecification. *Psychological Methods, 3*(4), 424-453.

Hulland, J. (1999). Use of Partial Least Squares (PLS) in Strategic Management Research: A Review of Four Recent Studies. *Strategic Management Journal, 20*(2), 195-204.

Hussin, S., Manap, M. R., Amir, Z., & Krish, P. (2012). Mobile Learning Readiness among Malaysian Students at Higher Learning Institutes. *Asian Social Science, 8*(12), 276.

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.

Kafai, Y. B. (2016). From Computational Thinking to Computational Participation in K-12 Education. *Communications of the ACM, 59*(8), 26-27.

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational What? Relating Computational Thinking to Teaching. *TechTrends*, 1-11.

Kent,T. W, Sc McNergney, R. F. (1999). *Will Technology Really Change Education?: From Blackboard to Web. Thousand Oaks.* CA: Corwin Press.

Kihoza, P., Zlotnikova, I., Bada, J., & Kalegele, K. (2016). Classroom ICT Integration in Tanzania: Opportunities and Challenges from the Perspectives of TPACK and SAMR models. *International Journal of Education and Development Using Information and Communication Technology, 12*(1), 107-128.

Koehler, M., & Mishra, P. (2009). What is Technological Pedagogical Content Knowledge (TPACK)? *Contemporary Issues in Technology and Teacher Education, 9*(1), 60-70.

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.

Koehler, M. J., Mishra, P., & Cain, W. (2013). What is Technological Pedagogical Content Knowledge (TPACK)?. *Journal of Education, 193*(3), 13-19.

Marks, H. M., & Printy, S. M. (2003). Principal Leadership and School Performance: An Integration of Transformational and Instructional Leadership. *Educational Administration Quarterly, 39*(3), 370-397.

Matherson, L. H., Wilson, E. K., & Wright, V. H. (2014). Need TPACK? Embrace Sustained Professional Development. *Delta Kappa Gamma Bulletin, 81*(1), 45.

Mishra, P., & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record, 108*(6), 1017.

Mishra, P., Spiro, R. J., & Feltovich, P. J. (1996). Technology, Representation, and Cognition: The Prefiguring of Knowledge in Cognitive Flexibility Hypertexts. In H. van Oostendorp & A. de Mul (Eds.), *Cognitive Aspects of Electronic Text Processing*, 287-305. Norwood, NJ: Ablex.

Nelson, M. J., Voithofer, R., & Cheng, S.L. (2019). Mediating Factors that Influence the Technology Integration Practices of Teacher Educators. *Computers & Education*, *128*, 330-344.

Ringle, C. M., Wende, S., & Becker, J.M. (2015). *SmartPLS 3. Boenningstedt: SmartPLS GmbH*. Retrieved December 29, 2015, from http://www.smartpls.com.

Sawchuk, S. (2010). *Professional Development for Teachers at Crossroads.* Education Week. Retrieved November 10, 2010, from https://www.edweek.org/ew/articles/2010/11/10/11pd_overview.h30.html.

Scherer, R., Tondeur, J., Siddiq, F., & Baran, E. (2018). The Importance of Attitudes Toward Technology for Pre-service Teachers' Technological, Pedagogical, and Content Knowledge: Comparing Structural Equation Modeling Approaches. *Computers in Human Behavior, 80*, 67-80.

Schmidt, D. A., Baran, E., Thompson, A. D., Mishra, P., Koehler, M. J., & Shin, T. S. (2009). Technological Pedagogical Content Knowledge (TPACK) the Development and Validation of an Assessment Instrument for Preservice Teachers. *Journal of Research on Technology in Education, 42*(2), 123-149.

Shulman, L. S. (1986). Those Who Understand: Knowledge Growth in Teaching. *Educational Researcher, 15*(2), 4-14.

Soini, T., Pietarinen, J., & Pyhältö, K. (2016). Leading a School through Change–principals' Hands-on Leadership Strategies in School Reform. *School Leadership & Management*, *36*(4), 452-469.

Spiro, R. J., & Jehng, J.C. (1990). Cognitive Flexibility and Hypertext: Theory and Technology for the Nonlinear and Multidimensional Traversal of Complex Subject Matter. In D. Nix & R. Spiro (Eds.), *Cognition, Education, and Multimedia: Exploring Ideas in High Technology*, 163-204. Hillsdale, NJ: Lawrence Erlbaum Associates.

Szeto, E., & Cheng, A. Y. N. (2018). Principal–teacher Interactions and Teacher Leadership Development: Beginning Teachers' Perspectives. *International Journal of Leadership in Education, 21*(3), 363-379.

American Association of Colleges for Teacher Education. Committee on Technology. (2008). *Handbook of Technological Pedagogical Content Knowledge (TPCK) for Educators.* Routledge.

Thomas, T., Herring, M., Redmond, P., & Smaldino, S. (2013). Leading Change and Innovation in Teacher Preparation: A Blueprint for Developing TPACK Ready Teacher Candidates. *TechTrends*, *57*(5), 55-63.

Tong, Y., & Brennan, R. L. (2007). Bootstrap Estimates of Standard Errors in Generalizability heory. *Educational and Psychological Measurement, 67*(5), 804-817.

Vatanartiran, S., & Karadeniz, S. (2015). A Needs Analysis for Technology Integration Plan: Challenges and Needs of Teachers. *Contemporary Educational Technology, 6*(3), 206-220.

Voogt, J., Fisser, P., Pareja Roblin, N., Tondeur, J., & Van Braak, J. (2013). Technological Pedagogical Content Knowledge–A Review of the Literature. *Journal of Computer Assisted Learning, 29*(2), 109-121.

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

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.

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.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking in Finnish Pre-Service Teacher Education

Kati H. MÄKITALO[1*], Matti TEDRE[2], Jari LARU[3], Teemu VALTONEN[4]
[13] University of Oulu, Finland
[24] University of Eastern Finland, Finland
kati.makitalo@oulu.fi, matti.tedre@uef.fi, jari.laru@oulu.fi, teemu.valtonen@uef.fi

ABSTRACT

The pressures of digitalization and computerization mount on our educational systems, but progress is slowed down by a weak research base on integration of the necessary skills into the school subjects. We aim to fill widely recognized gaps in knowledge on how to integrate computational thinking (CT) into school curricula. We merge two popular frameworks - CT from the field of computing and TPACK from the field of education in order to test the new CTPACK framework with teacher students as well as in-service teachers over a large number of subjects and topics. Through design research, educational interventions, and case study research, we produce 1) an extended pedagogical CTPACK framework; 2) a model for seamlessly integrating CT, subject matter, pedagogy, and technology (TPACK); 3) curriculum guidelines for CT integration; and 4) apps and frameworks for evaluating CT skill progression.

KEYWORDS

computational thinking, curriculum, pre-service teacher education, TPACK

1. INTRODUCTION

Successfully living and working in a digitalized society requires skills and competences that are often referred to as 21st century skills. One of those new skills, computational thinking (CT), has recently become a catchphrase for a broad variety of efforts to bring computing skills and knowledge into school curriculum—not only as a part of computer science courses, but as a part of all school subjects (Guzdial, 2015; Tedre & Denning, 2016; Mouza et al., 2017; Lockwood & Mooney, 2017; Denning & Tedre, 2019). Broadly speaking, computational thinking refers to the skills and competences necessary for understanding, controlling, and automating information processes. Introduction of CT to schools, curriculum guidelines, and frameworks have been prepared by major organizations, including CSTA in the US, CAS in the UK, ACARA in Australia, as well a large number of other national bodies around the world. Also, the Finnish national K–9 and K–12 curricula (The National Core Curriculum 2014) now include computing—although not as a separate subject to be taught but as a cross-curricular topic that should be integrated in all subjects—while in reality computing in Finnish schools is almost solely about basic programming concepts. After a long history in the academia and the school system (Denning & Tedre, 2019), the latest wave of CT for K–12 started in 2006 when Jeannette Wing leveraged her position at the US National Science Foundation (NSF) to campaign her CT vision to the public, funding bodies, and academic audiences (Wing, 2006). Wing's rallying cry persuaded decision-makers to intensify CT efforts in schools in a large number of countries, starting with the US. The public interest, combined with focused and sustained research funding for CT under Wing's NSF directorship, led to some impressive achievements, such as a new advanced placement (AP) program in the US, the Obama administration's $4 billion investment in CS for All (NSF, 2016), the training of 10,000 computing teachers in the US, and a surge of popular initiatives like code.org, k12cs.org, and Google for Education. European countries have their own initiatives and movements, too, albeit they are fragmented between languages and countries. The latest literature surveys list hundreds of recent articles, textbooks, and journal special issues (Lockwood & Mooney, 2017). Universities like CMU, Harvard, and MIT have recently set up CT research centers or research programs. There are a number of early models and tools for testing for CT skills, such as the MCT (Mobile CT) model (Sherman & Martin, 2015), the PECT (Progression of Early CT) model (Seiter & Foreman, 2013) and the REACT (Real-Time Evaluation and Assessment of CT) tool (Koh et al, 2014). Valentina Dagienė's CT-related "Bebras Challenge" has been done by more than two million students worldwide (bebraschallenge.org). Despite all this, there is a consensus in the "CT for K–12" community that there are notable gaps in the state-of-the-art knowledge about CT in K–12 schools. The research literature has repeatedly highlighted three especially critical problems.

The first problem, which Denning called one of the "remaining trouble spots" with CT education, is "How do we measure students' computational abilities?" (Denning, 2017). For example, Mark Guzdial's (2015) quintessential textbook on CT barely touches the topic of how to evaluate CT skills. A 2017 literature review of more than 200 CT studies and initiatives noted that despite some models for testing for CT skills, "work in testing for CT is in its infancy," and the existing models are "in the early stages of development" (Lockwood & Mooney, 2017). A large working group report identified a number of pioneering articles, each of which urged researchers to turn their attention to how to measure CT skill development (Mannila et al., 2014). Those problems are exacerbated by a lack of consensus over what skills does CT consist of, and over how does skill progression in CT look like.

The second critical problem is "How do we integrate CT in school subjects?" CT is typically taught in specialized computing courses, instead of being integrated in school subjects. While there is less empirical research on whether integrating CT into school subjects promotes students' analytical skills or widens their understanding about CT (Guzdial, 2015), integrating CT in school subjects is crucial for understanding the ways in which digitalization and computerization have changed all sciences and areas of life. In line with the educational vision of teaching disciplinary ways of thinking and practicing, the knowledge and skills of CT fit well the formal learning goals of other subjects.

However, due to fewer concrete examples of how to integrate CT into school subjects, there is a need for empirical evidence on how CT integration influences pupils' learning outcomes. But promoting CT in primary education is challenging due to its specialized nature and broad applicability: Few primary school teachers have the requisite knowledge and skills on CT as a concept or on its integration in other subjects, and while computer scientists have knowledge and skills on CT in computing, they lack the knowledge and skills for CT integration in education (Mouza et al., 2017).

The third critical problem has to do with lack of understanding on "How do teachers learn to synthesize CT with existing content and pedagogical strategies?" (cf. Mouza et al., 2017). In other STEAM fields (science, technology, engineering, arts, and mathematics), a popular TPACK framework (technological pedagogical content knowledge) offers a tool for understanding teachers' technological knowledge and skills needed for integrating technology and technological resources effectively in classrooms. TPACK is an actively used theoretical framework for studying how (pre-service) teachers combine the areas of technology and pedagogy with content taught (Koehler & Mishra, 2009). Valtonen et al. (2017) have shown that despite of intensive integration of technology, content, and pedagogy, pre-service teachers experience technological knowledge as a separate domain. Studies show that integrating CT in teacher education courses enhances pre-service teachers' knowledge and understanding about CT, but when CT is taught separate from the teachers' own discipline, that understanding remains at an abstract level which is not applied in teaching (Yadav, Stephenson, & Hong, 2017). TPACK framework offers a practical model for integrating CT within those subject matters and pedagogical approaches that pre-service teachers are expected to teach in future classrooms (Yadav et al., 2017). However, CT literature has pointed out a dire need for more research on understanding teacher learning and how to support their learning in best way in pre- and in-service teacher education (Guzdial, 2015; Mouza et al., 2017).

We address the three critical research problems above by

1. Expanding our current knowledge on how to improve pupils' CT knowledge and skills as well as pre- and in-service teachers' TPACK and CT curriculum integration knowledge and skills,

2. Investigating how to combine CT and TPACK knowledge and skills with subject matter content and pedagogy—we call that new framework CTPACK,

3. Defining and testing different models for integrating CT into curriculum both in schools as well as teacher education, and

4. Designing evaluation frameworks and tools for assessing CT skill progression.

Through design research, educational interventions, and case study research, we produce 1) an extended pedagogical CTPACK framework; 2) a model for seamlessly integrating CT, subject matter, pedagogy, and technology (TPACK); 3) curriculum guidelines for CT integration; and 4) apps and frameworks for evaluating CT skill progression.

2. RESEARCH OBJECTIVES

In terms of research objectives, the study aims are divided into four objectives:

The concept of "computational thinking" has a variety of definitions (Tedre & Denning, 2016; Denning & Tedre, 2019) that provide different starting points for applying CT for education. Together with CT pioneers we will analyze, synthetize, and conceptualize a CT skill and competence progression scheme based on a systematic review of theoretical and empirical work. In order to establish a baseline for improving higher education, we study how pre-service teachers understand the principles of computational thinking, how they define the concept, what kinds of personal epistemologies do they create, and how they would apply the principles of CT in education.

While TPACK has become a standard feature of education research, its central constituents—technology, pedagogy, and content—do not directly address the main driver of digitalization: CT (Angeli et al., 2016; Mouza et al., 2017). We will synthesize TPACK and CT into a new CTPACK framework that establishes CT as part of pedagogical–technological–curricular thinking and as a target of learning, too. We investigate pupils' CT knowledge and skills as well as school teachers' CTPACK knowledge and skills through interventions at school with our pre-service teachers and computer science students. Our new CTPACK pedagogical framework is accompanied by example learning objects and empirical results on pupils' CT development and teachers' CTPACK knowledge and skills.

For facilitating the uptake of our newly developed CT and CTPACK frameworks, our project will study the concretization of those frameworks in a collaboration between in-service teachers and pre-service teachers. We will study how the comprehension of CT can be supported through technology and classroom practices in the context of STEAM topics, and what leeway does CT give to STEAM classes. We will outline interventions within teacher education and computer science context to design best pedagogical practices, assessment practices and triggering case elements.

Seamless integration of technology is essential for enabling the technological elements of TPACK. This study develops and tests technological principles for designing CT tools for classroom integration. In collaboration between teacher education students we will design, implement, and test apps for triggering and supporting pupils' and teachers' innovative thinking for designing ways to apply principles of CT in various learning contexts. We will also design and implement interactive apps for evaluating pupils' skill and competence progression in CT.

3. PARTICIPANTS AND METHODS

Participants of this case study, which is a part of wider research programme, are 28 teacher education (TE) students (pre-service primary school teachers) and grade 1-6 pupils in primary schools.

Quantitative and qualitative methods are used for collecting and analyzing the data. The survey includes open-ended questions, which we ask all participants to complete at the beginning and at the end of the interventions. Open ended questions are aimed at clarifying what participants think that the term computational thinking means, how they see technologies could be used to support the development of pupils' CT skills, and how students' personal epistemologies develop over time. We collect all the material produced by pre-service teachers, for example, case reports, feedback, products and materials designed for pupils' use.

This presented case study is done in the context of a pre-service teacher educational technology course. A project-based learning approach will bring together pre-service teachers to work in groups. Their task is to design learning objects (series of short lessons) for pupils' activities where their CT knowledge and skills are enhanced and integrated into the recent curriculum. Learning objects will be designed for the school's e-learning platform and researchers and CS students will develop external educational apps, which fit into curricula and are available for everyone. The platform also provides teachers the means to share their own lesson plans and experiences and give feedback on how these series of lessons work in practice.

TE students enter the schools to implement CT-supplemented STEAM lessons in the school context and monitor the project at the implementation level. TE students and researchers engage in research-based activities during the implementation, mainly observation. They will write case reports where they focus on describing the implementation as well as case reflection. Main research data from this phase contains observations and case reports. We will continuously revise the instructions, learning environment, and the instruments for improving the courses.

4. DISCUSSION

Overall, this research programme combines CT and design of information technology with TPACK and technology-enhanced learning and teaching. That combination goes beyond the current global state-of-the-art research in CT education due to its generalizability and solid empirical evidence base as well as its grounding in the pedagogically sound TPACK and conceptually solid CT frameworks. By bringing together new global state-of-the-art knowledge in CT and top Finnish education expertise, this project creates a new horizon for CT/TPACK education research. It also broadens the current understanding of the 21st century teacher education by producing empirical results on the effect of project- and integration-based interventions on teachers' CTPACK knowledge and skills.

In the conference, we present preliminary results about pre-service teachers' perception about CT and their pilots' related to CT into curriculum. As the project is work in progress, the CTE2019 conference provides a great opportunity for us to discuss with the leading experts in the CT field in order to gain the best results of this study with regarding to the theoretical framework, methods and practice.

5. REFERENCES

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. *Educational Technology & Society*, *19*(3), 47-57.

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

Denning, P. J., & Tedre, M. (2019). *Computational Thinking*. Cambridge, MA: The MIT Press.

Guzdial, M. (2015). *Learner-Centered Design of Computing Education: Research on Computing for Everyone. Synthesis Lectures on Human-Centered Informatics*. San Rafael, CA: Morgan & Claypool.

Koehler, M., & Mishra, P. (2009). What is Technological Pedagogical Content Knowledge (TPACK)? *Contemporary Issues in Technology and Teacher Education*, *9*(1), 60-70.

Koh, K. H., Basawapatna, A., Nickerson, H., & Repenning, A. (2014). Real Time Assessment of Computational Thinking. *In 2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 49-52.

Lockwood, J., & Mooney, A. (2017). *Computational Thinking in Education: Where Does It Fit? A Systematic Literary Review*. Technical Report, National University of Ireland Maynooth.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. *In Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*. ACM, 1-29.

Mouza, C., Yang, H., Pan, Y. C., Yilmaz Ozden, S., & 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), 61-76.

The National Core Curriculum. (2014). *The Finnish National Board of Education*. Retrieved September 9, 2018, from http://www.oph.fi/download/163777_perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

NSF [National Science Foundation]. (2016). *CS for All*. Retrieved September 9, 2018 from https://www.nsf.gov/news/special_reports/csed/csforall.jsp

Seiter, L., & Foreman, B. (2013). Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. *In Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research, ICER '13*. ACM, 59-66.

Sherman, M., & Martin, F. (2015). The Assessment of Mobile Computational Thinking. *Journal of Computing Sciences in Colleges*, *30*(6), 53–59.

Tedre, M., & Denning, P. J. (2016). The Long Quest for Computational Thinking. *In Proceedings of the 16th Koli Calling International Conference on Computing Education Research, Koli Calling '16*. ACM, 120-129.

Valtonen, T., Sointu, E., Kukkonen, J., Kontkanen, S., Lambert, M. C., & Mäkitalo-Siegl, K. (2017). TPACK Updated to Measure Pre-service Teachers' Twenty-first Century Skills. *Australasian Journal of Educational Technology*, *33*(3), 15-31.

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

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational Thinking for Teacher Education. *Communications of the ACM*, *60*(4), 55-62.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking Education for In-Service Elementary Swedish Teachers: Their Perceptions and Implications for Competence Development

Dan KOHEN-VACS[1*], Marcelo MILRAD[2*]
[1] Department of Computer Science and Media Technology, Linnaeus University, Sweden
[1 2] Faculty of Instructional Technologies, Holon Institute of Technology, Israel
dan.kohen@lnu.se, marcelo.milrad@lnu.se

ABSTRACT

Many countries and governments have laid down a national strategy for digitalization in schools with the intent to strengthen both pupils' and teachers' digital competences. Computational Thinking (CT) can be regarded as a central ingredient of teaching and learning in the XXI century and should be part of these competences. The Swedish government has recently launched a national strategy for digitalization in schools stating that CT and programming should be integrated in many school subjects. Thus, one particular challenge that needs to be addressed is the one related to teachers' competencies and skills related to programming. Since 2016, The National Agency of Education has been working together with several Swedish universities on developing academic courses in the field of CT and programming for in-service teachers. One challenge we are exploring in this paper is the one related to teachers' perceptions and competencies related to programming. As part of our on-going efforts in the course *“Introduction to Programming for Elementary Teachers”* offered to in-service teachers we have analyzed data we collected representing teachers´ perceptions on CT and programming before the start of the course. We applied sentiment mining and word counting on 127 texts generated by the teachers. We examine their perceptions and cluster them accordingly in order to understand how teachers perceived different aspects related to CT education. By doing so, we lay down the foundations on how to tailor competence development efforts in this field so that they fit teachers´ need. We expect these efforts will lead to the development of a model enabling to identify teachers´ current perceptions in order to plan future actions that can increase their Technological Pedagogical Content Knowledge.

KEYWORDS

computational thinking (CT), teacher education, visual programming, CT across subjects

1. INTRODUCTION

Many countries and governments have laid down a national strategy for digitalization in schools with the intent to strengthen both pupils' and teachers' digital competences and skills. Computational Thinking (CT) can be regarded as a central ingredient of teaching and learning in the XXI century and should be part of these skills and competences (Voogt et al., 2015; García-Peñalvo et al., 2016).

Even though the concept of CT has gained a lot of attention in the context of education and schools in the last decade, the core ideas and concepts behind it are not new (Yadav et al., 2017). Actually, Denning (2017) claims that the idea of CT exists since the 1960s and can be referred also as *algorithmic thinking* or Traditional CT. The current understanding of CT can be defined as higher order thinking processes and skills involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out (Wing, 2006). According to Denning (2017), there are some differences between the traditional CT and the newer and more recent CT, as elaborated by Wing (2006). These two views on CT are not the same. One of the distinct differences is that in the traditional CT view *programming ability* may produce CT, while in the *new* CT perspective learning and exploring certain concepts in a variety of subjects may lead to *programming ability*. As quoted by Denning (2017), *“The direction of causality is reversed”.*

As implied from the discussion above, these shifts in the interpretation of CT have major implications for the introduction of different aspects of computational thinking in the educational system at various levels and subjects. These among others include; design of proper curriculum, pedagogical strategies involved in its implementation, competence development actions for in service teachers, as well as practical issues related to the implementation of these actions (Bean et al., 2015). Novel pedagogical approaches for introducing CT into the classroom bring teachers to design complex and open learning activities that offer students challenging educational opportunities for fostering reasoning and creative problem solving (Chang, 2016; Malizia et al., 2017; Wing, 2014). The implementation of such efforts aims to prepare students for their future civic lives while emphasizing on understanding how digitalization affects the individual and society's development (Grover & Pea, 2018).

These trends in modern education awakes a myriad of challenges and questions related to the integration of computational thinking in school teaching concerning a wide variety of subjects and levels. Traditionally, STEM (Science, Technology, Engineering and Math) related subjects tend to be “naturally” and frequently selected by educators as topics to be associated and integrated with programming lessons (Grover & Pea, 2018). In recent years, governmental agencies and educational institutions are demanding to integrate different CT related aspects also in the humanities, social sciences and the arts. Subject domains that are taken into consideration include language, literature, history, music and arts to mention some of them. In such cases, new challenges arise as the integration of CT and programming into these subjects seems to be less intuitive if compared to STEM related topics (Yadav et al., 2017).

The Swedish government has recently launched a national strategy (https://bit.ly/2QoHjbq) for digitalization in schools. The basic idea is that the Swedish society, oriented towards building a sustainable future supported by Information and Communication Technologies (ICT), should build on cultivating innovative competences for all citizens (Bocconi et al., 2018). The application of this strategy has brought changes to the national school curriculum for primary and secondary levels, starting July 1st, 2018. One important action point as part of this strategy deals with the fact that computational thinking and programming should be now integrated in many school subjects (not just mathematics and technology) and not be only taught as a subject by its own (Malyn-Smith et al., 2018). However, the Swedish strategy does not provide a prescribed plan on how to approach these changes.

Thus, one particular challenge that needs to be addressed is the one related to teachers' competencies and skills related to computational thinking and programming. Since 2016, The National Agency of Education (Skolverket) has been working together with a number of Swedish universities on the development of academic courses in the field of computational thinking and programming for in-service teachers. Since the fall 2016, the authors of this paper have been systematically working in different activities (seminars, hands-on workshops, academic courses) to support knowledge building and competence development for teachers. In this paper, we present the results of our efforts related to the development, and implementation of a six months long academic course on *CT and programming for elementary school teachers*. Our view on programming is that it is not just about writing code. It is also about creative problem solving, logical thinking and structured working methods. We understand programming as a technique, a medium for self-expression and an entry point for developing new ways of thinking (Grover & Pea, 2018). One particular challenge we are exploring in this paper is the one related to teachers' perceptions, competencies and skills related to programming. Thus, the main research question that guided our efforts has been formulated as following: *What are the initial teacher´s perceptions and understanding in relation to the introduction of programming and computational thinking in the classroom?*

The remaining of the paper is organized as follows. In the next section, we present the educational context and research settings that served as a ground for the exploration of our research question. We then proceed by presenting the elaboration of our results followed by a discussion section together with a brief description about the directions of our future work.

2. EDUCATIONAL CONTEXT & RESEARCH SETTINGS

In this section, we present detailed information about the course we have developed, its content and teaching and learning approach. We briefly explain which data we collected (to be further elaborated in section 3) from the participants followed by some background information about the participants of the course.

2.1. *Content of the course, structure & forms of delivery*

This course is called *"Introduction to Programming for Elementary Teachers"* and is offered to in-service teachers across the entire country. The department of Computer Science and Media Technology at Linnaeus university (LNU), Sweden has been responsible for this course. In terms of content, the course gives students the fundamentals of programming combined with techniques and approaches on how to integrate those in the classroom.

Upon completion students, should be able to know how programming can be used as a powerful tool for different forms of expression in elementary schools. Moreover, they should be able to master different programming techniques to break down, analyze and interpret subject maters in novel ways. Visual programming was used as the programming metaphor combined also with physical computing devices and sensors.

The course has been offer during the period November 2017- June 2018 and has been given in a blended form including mostly on-line meetings, video-recordings combined with 3 compulsory meetings on site. Previous to the start of the course, students were asked to complete an on-line survey where they were asked a number of questions about their professional experience, areas of expertise, previous knowledge and perceptions about programming and expectations about the course. In terms of content and efforts required from the students, 100 hours of the course have been used to teach, exercise and test the fundamentals of programming while 100 hours were used to validate these ideas in the classroom. The students were required to complete five different assignments related to fundamental of programming and CT concepts. Additional, teachers were asked to conduct a pilot project related to programming in their classrooms. This activity included design, test, validation and assessment of learning activities related to the introduction of programming in the classroom. A written scientific report (10 pages long) and a 3-5 minutes short video film about their pilot projects were part of the expected outcomes.

2.2. Participants

In terms of participants attending the course, we had 127 teachers representing 51 local municipalities from all 290 existing in Sweden. Based on the demographic and professional data we collected we could gain some information about the years of teaching experience they have and the areas in which they teach. The teachers had an overall average of 14.33 years of experiences as practitioners. Many of them (N=79) were teachers in subjects such as mathematics and/or technology. Approximately 25% of them (N=31) were teachers in combined areas such social sciences, mathematics and technology. Nine of those teachers specialized in ICT while other four were language teachers. The remaining four teachers were active in other domains. Participants teachers did represent different grades including the following distribution: 31% of teach in 1st to 3rd grade. About 27% teach in 4th to 6th grade and another 34% of them teach in 7th

to 9th grade. The remaining 11% of the teachers teach at other levels (secondary school/special education).

3. METHODOLOGICAL APPROACH AND RESULTS

In this section, we discuss how we analyzed the different set of data we have collected and the approaches and techniques used to interpret them. In the rest of the section we elaborate on those aspects and present our results.

3.1. Methodological Approach

In section 1 of this paper we stated one question that guides the core of this research and focuses on examining the initial teachers' perceptions related to the topics of programming and CT before the start of the course. One of our lines of reasoning for exploring this particular aspect is the fact that it may be a gap between what the National Agency of Education and the new revised curriculum expects and what Swedish teachers understand about those changes. Research has been scarce showing how teachers´ perceptions and understanding of core concepts of CT and programming should be taken into consideration when implementing these new changes in the classroom. This particular perspective has not be addressed neither discussed in recent research exploring different issues in connection to teachers 'education (Grover & Pea, 2018; Voogt et el., 2015 & Yadav et al., 2017). Moreover, these authors have identified the need for more research exploring the integration of CT in education and in particular how CT can be developed in students and in disciplines beyond Computer Science. Specifically, we intend to examine and understand teachers´ perceptions in order to adapt and develop competence development efforts and strategies so that they are carefully implemented. In order to explore teachers´ perceptions regarding aspects of CT and programming we apply sentiment mining (Leong et al., 2012) for analyzing the pieces of text (200 words) generated by the teachers based on the data we collected before starting the course (as described in section 2). We wrote a piece of software and used *TextBlob*, a Python library for processing textual data, in order to perform the sentiment mining (Vijayarani & Janani, 2016). Additionally, we also processed teachers' texts while using a tag-cloud generator (Roe, 2018) that allowed us also to identify the frequency of words used in these texts. We used this tool in order to identify prominent and less prominent terms, ideas and concepts as perceived by the teachers. In the next subsection, we continue and present results.

3.2. Sentiment Analysis of Teachers´ Perceptions on CT

As mentioned previously, we applied sentiment mining on 127 pieces of texts generated by the teachers describing their thoughts and attitudes towards CT and programming in the classroom. We examine teachers' ideas and perceptions and cluster them according to the years of experience they have as in service teachers. We then checked for the polarity of all texts as it can be on the range of -1 (for negative sentiment) and +1 (for positive sentiment). Thus, we aim on checking whether the series of examined sentences reflect positive, negative or neutral standpoints. The overall computed polarity resulted on a kind of neutral with a slight tendency for positivity. We divided the entire class into 7 groups classified by years of teaching experience. We found the highest mark of polarity among the group of the teachers having 16 to 20 years of experience (N=18; p=0.304). Additionally, we checked the correlation between the years of experience and the polarity. We spotted such correlation only in the group corresponded to teachers having between 12 to 16 years of experience. In this one, we found moderate, positive and meaningful correlation (r=0.399, n=28, p=0.028). The analysis of these results point out to a kind of neutral attitude and perceptions towards topics related to programing and CT. We did complement these efforts by examining also the most frequent words (terms and concepts) used by the teachers in their texts. Table 1 depicts the most 3 frequent words that have been identified in their texts and present them in relation to the years of experience for each group of teachers.

Table 1. Frequencies of words within the different groups

Years of experience	1st	2nd	3rd
0 - 4	programming (6)	foundations (3)	class (3)
4 - 8	programming (43)	advice (15)	knowledge (11)
8 - 12	programming (26)	school (7)	knowledge (6)
12 - 16	programming (24)	learn (10)	students (9)
16 - 20	programming (17)	students (13)	class (8)
20 - 24	programming (37)	school (15)	learn (11)
above 24	programming (12)	students (5)	subjects (4)

The most frequent word for all groups has been *"programming"* and it has been mentioned 165 times. The words *"students",* "*school*" as well as "*class*" were mentioned 60 times. In addition, those words connected to teaching and learning ("*learn", "knowledge"* and "*advice*") were cited 53 times. We notice in this analysis that teachers addressed the tool (programming) and various organizational aspects related to education and learning but refers less to issues and processes connecting to subject matter and its relation to programming. Furthermore, teachers did not address directly to CT or related terms in their texts.

4. DISCUSSIONS AND CONCLUDING REMARKS

Integrating computational thinking education concepts and ideas into in-service elementary teacher practices is a very challenging task. In the light of paradigms shifts and changes in the national curriculum teachers are required to gain new knowledge and skills related to how content, pedagogical strategies and ICT tools need to be combined to introduce CT into their educational practices in meaningful ways. The ultimate objective of those actions is to allow learners to apply these ideas to solve domain-specific and interdisciplinary challenges and problems. In the previous section, we presented and analyzed the ideas and perceptions coming from more than 100 Swedish elementary teachers at the beginning of the course. We have analyzed these data as we consider their perceptions crucial for creating and developing a framework for teachers´ competence development in this field. The framework proposed by

Darling-Hammond and Bransford (2005) could be modified and adapted in order to prepare teachers to gain new knowledge and skills related to different aspects of CT as well as methods and tools that teachers should acquire for integrating these ideas into subject matter and curriculum goals. By doing so, we have collected some data that can give some answers to the RQ presented in section I. Our initial analysis points out to the fact that teachers may have different perceptions regarding issues related to programming and computational thinking in the classroom depending on their years of experience and pedagogical knowledge. One salient point is the fact that most of the teachers have problems to see how CTE and its relation to teaching and learning about a specific subject matter can be pedagogically and instrumentally integrated in the classroom. We also acknowledge that policies and curriculum changes (including their practical implementation) in the field of CTE, like those underway in Sweden (and even in other countries), are very complex processes. First, there are many different stakeholders (ministries, schools, universities, politicians and even the industrial sector) involved. Secondly, these changes generate new challenges as preserving a balance between legislation and the implementation of these changes in becomes a real challenge as teachers and students need new knowledge and skills to cope with those on a daily basis. Consequently, teachers seem to be caught between, top-down political decisions and the realities of everyday classroom practices. It is important also to recognize that the current lack of an agreed-upon, exclusive definition of the elements of computational thinking education makes it a challenge to develop clear pathways for in-service teachers to be computationally thinking educated. The analysis and elaboration of the results we have presented in section III, combined with the knowledge and experience we gained during this course have helped us to identify a kind of categorization describing teachers' perceptions and actions addressing the incorporation of CT concepts and programming tools into the curricula and across domains and levels (Kjällander et al, 2018).

In our coming efforts we will further elaborate on the ideas presented in this paper in order to develop a model that could help researchers, policy makers and educators to identify teachers´ current perceptions and previous knowledge in order to plan and implement future actions and competence development activities and how their Technological Pedagogical Content Knowledge (TPACK) (Mishra & Koehler, 2006) should progress accordingly. In this way, the TPACK framework could be used as a model for integrating CT education where core ideas are closely weave within the subject matter and pedagogical approaches that will assist in-service teachers that need to introduce them into their classrooms and everyday practices.

5. REFERENCES

Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). Starting from Scratch: Developing a Pre-service Teacher Training Program in CT. *Proceedings of Frontiers in Education Conference (FIE).* IEEE, 1-8.

Bocconi, S., Chioccariello, A., & Earp, J. (2018). The Nordic Approach to Introducing Computational Thinking and Programming in Compulsory Education. *Report prepared for the Nordic@ BETT2018 Steering Group.*

Chang, C. (2016). Using Computational Thinking Patterns to Scaffold Program Design in Introductory Programming Course. *Proceedings of 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI).* IEEE, 397-400.

Darling-Hammond, L. and Bransford, J., Eds. (2005). *Preparing Teachers for a Changing World: What Teachers Should Learn and Be Able to Do*. San Francisco, CA: Jossey-Bass.

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

Grover, S., & Pea, R. (2018). Computational Thinking: A Competency Whose Time has Come. *Computer Science Education: Perspectives on Teaching and Learning in School*, 19.

Malizia, A., Fogli, D., Danesi, F., Turchi, T., & Bell, D. (2017). TAPASPlay: A Game-based Learning Approach to Foster Computation Thinking Skills. *Proceedings of 2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 345-346.

Malyn-Smith, J., Lee, I. A., Martin, F., Grover, S., Evans, M., & Pillai, S. (2018). Developing a Framework for Computational Thinking from a Disciplinary Perspective. *Proceedings of International Conference on Computational Thinking Education 2018.* Hong Kong: The Education University of Hong Kong, 182-184.

Mishra, P. & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record, 108,* 1017-1054.

Leong, C. K., Lee, Y. H., & Mak, W. K. (2012). Mining Sentiments in SMS Texts for Teaching Evaluation. *Expert Systems with Applications*, *39*(3), 2584-2589.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). *An Overview of the Most Relevant Literature on Coding and Computational Thinking with Emphasis on the Relevant Issues for Teachers*. Belgium: TACCLE3 Consortium.

Roe, A. (2018). Generating Word Clouds. *The School Librarian*, *66*(1), 19-19.

Vijayarani, S., & Janani, R. (2016). Text Mining: Open Source Tokenization Tools–An Analysis. *Advanced Computational Intelligence*, *3*(1), 37-47.

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

Wing, J. (2014). Computational Thinking Benefits Society. *40th Anniversary Blog of Social Issues in Computing*.

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

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational Thinking in Teacher Education. *In Emerging Research, Practice, and Policy on Computational Thinking*. Cham: Springer, 205-220.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Technology Acceptance and Teacher Attitude for Computational Thinking in the Netherlands

Marcus SPECHT[1*], Robert Jan JOOSSE[2]
[1] Delft University of Technology, Netherlands
[12] Open University Netherlands, Netherlands
m.m.specht@tudelft.nl, rj.joosse@studie.ou.nl

ABSTRACT
This paper presents a study with 106 teachers from primary and secondary education in the Netherlands investigating their attitude and acceptance factors for enabling the implementation of Computational Thinking in their classrooms. The results show that performance expectancy, effort expectancy and attitude are important predictors of behavioural intention. Facilitating conditions have a more limited correlation with the behavioural intention. For the attitude it appears that this is mainly influenced by performance expectancy and effort expectancy. Social influence does not appear to have a significant connection with behavioural intention. Nor have there been any indications that gender, age and experience have moderating effects on performance expectancy, effort expectancy, social influence and facilitating conditions.

KEYWORDS
coding skills, computational thinking, attitude, UTAUT model, ICT acceptance of teacher attitudes

1. INTRODUCTION
An international debate is taking place about the importance of coding skills within education in relation to future professions. Several countries have already implemented coding skills in education, but in the Netherlands this is not yet the case. Teachers have a key role in educational innovation and therefore it is necessary that they are positive about coding skills. The UTAUT model (Venkatesh et al., 2003) is often used for research into ICT acceptance. In this model, performance expectancy, effort expectancy, social influence and facilitating conditions are direct predictors of the intention to use a system. This behavioural intention is decisive for the actual use of a system. Furthermore, research shows that attitude plays a central role in teachers' use of new systems (Teo, 2011). With this research, the believes and attitudes of teachers were mapped and tested using the UTAUT model. The results contribute to the research that takes place around ICT acceptance. In addition, the results give direction to policy issues concerning coding skills in education.

There are various models to determine whether or not a new technology will be used in the future. A well-known basic model for this is the Technology Acceptance Model (TAM) from Davis (1989) (see Figure 1). This model assumes that in particular the two variables, perceived usefulness (PU) and perceived ease of use (PEU) determine whether or not a new technology will be used.

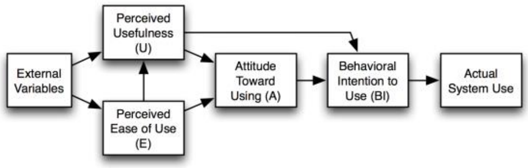


Figure 1. Technology Acceptance Model (Davis, 1989).

The Unified Theory of Acceptance and Use of Technology (UTAUT) model (Venkatesh, Morris, Davis, & Davis, 2003) was developed from these models. UTAUT explains, according to the authors, 70% of the variance in behavioral intention and exceeds the models on which these are based.

The central question in this study is: "To what extent do the four independent variables performance expectancy, effort expectancy, social influence and facilitating conditions have a relation with the use intent of teachers in primary and secondary education regarding computational thinking and specific coding skills."

2. Technology Acceptance, Teacher Attitude and Computational Thinking in the classroom
The research carried out is a cross-sectional study, in which the population consists of the teachers of primary education (PO) and secondary education (VO) of the Netherlands. 106 people completed the questionnaire. Of these 62 teachers are working in the PO and 43 teachers in the VO. As a starting point of our survey, a questionnaire was used that was adapted from a study to measure the views of teachers on coding skills in Finland, China and Singapore. In the first instance, an inventory was made of which questions are suitable for measuring constructs from the conceptual UTAUT model. The research proposal was approved by the the Ethical Research Committee (cETO) of the Open Universiteit.

The quality of the questionnaire was evaluated using a Cronbachs Alpha (α) analysis. The hypotheses from the research model were investigated by means of correlation analysis and by means of multiple regression analyses. To determine the mediating effect of attitude, various regression models have been drawn up.

3. RESULTS
A total of 106 respondents completed the questionnaire. Of these, 62.3% (N = 66) of the female gender and 29.2% (N = 31) of the male gender. The average age of the respondents

is 42.85 years (SD = 11.903). The averages of the variables show that in particular performance expectancy (M = 4.29, SD = 0.46) has a high score with regard to computational thinking, followed by effort expectancy (M = 3.76, SD = 0.75). Facilitating conditions (M = 3.19, SD = 0.90) and social influence (M = 3.26, SD = 0.89) have a lower score. Of the dependent variables, attitude (M = 4.07, SD = 0.52) has a higher average than the behavioral intention (3.43, SD = 0.87). Table 1 shows the averages of these values, with a distinction being made between teachers in primary education and secondary education.

Table 1. Means and Standard Deviations.

	Totaal		PO		VO	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
Performance Expectancy	4,29	0,46	4,36	0,42	4,19	0,50
Effort Expectancy	3,76	0,75	3,90	0,74	3,56	0,72
Social Influence	3,26	0,89	3,37	0,89	3,10	0,86
Facilitating conditions	3,19	0,90	3,23	0,87	3,13	0,95
Attitude	4,07	0,52	4,18	0,42	3,90	0,60
Behavioural Intention	3,43	0,87	3,46	0,86	3,38	0,90

With an independent t-test it was investigated whether there are significant differences between the above variables with respect to gender. No significant differences were found. With the help of an ANOVA it was checked whether there are significant differences between these variables and the type of school where the teachers work. Here there is a significant difference between teachers who teach in the lower part of the PO (M = 4.25, SD = 0.38), the higher part of the PO (M = 4.16, SD = 0.43), the lower part of the PO (M = 3.78, SD = 0.67) and the higher part of the VO (M = 3.96, SD = 0.58) and the attitude, F (3.102) = 3.18, p = 0.027. Primary education in particular has a more positive attitude towards computational thinking and coding skills. With respect to the other variables, no significant differences were found between the types of education.

Table 2 shows the averages, the standard deviations and the Pearson correlations between the different variables. The significant correlations are bold in this. Using a multiple regression analysis, it was also examined whether attitude can be predicted by performance expectancy and effort expectancy. This analysis shows that this regression model is significant, F (2.103) = 67.93, p <0.001. The model can thus be used to predict attitude and is strong in strength: 57% of attitude is predicted by PE, EE (R2 = 0.57). Performance expectancy, b * = 0.57, t = 7.38, p <0.001, 95% CI [0.41, 0.72], effort expectancy, b * = 0.29, t = 6.20, p <0.001, 95% CI [0.20, 0.38] have a significant correlation with the attitude.

Table 2. Correlation of Variables.

	Mean	S.D.	PE	EE	RC	SI	EX	BI
Performance Expectancy	4,29	0,46	1					
Effort Expectancy	3,76	0,75	,320**	1				
Facilitating Conditions	3,19	0,90	,275**	0,110	1			
Social Influence	3,26	0,89	,348**	0,156	,767**	1		
Experience	3,21	0,74	,348**	,654**	,225*	0,181	1	
Behavioural Intention	3,43	0,87	,508**	,580**	,220*	0,185	,592**	1
Attitude	4,07	0,52	,639**	,584**	0,095	,230*	,545**	,490**

4. DISCUSSION AND CONCLUSION

In this study, the teacher's views on teaching CT skills were investigated. This research has shown that the UTAUT model is a good starting point when measuring the intention of teachers to implement coding skills in the curriculum. In addition, it has emerged that attitude plays an important role in the behavioral intention with regard to the provision of coding skills in education. Although attitude in the original UTAUT model does not play a role (Venkatesh et al., 2003), the main focus is on the affective aspects of attitude, while in this research the focus is more on the cognitive aspects of attitude. The results obtained show that lecturers score high on performance expectancy and attitude. Although the research shows that facilitating conditions affect the use intention to a lesser extent than the other variables, it is recommended to take a critical look at this. The reason for this is that the qualitative data show that teachers experience insufficient facility support, which may affect the usage intent. As already mentioned, facilitating conditions is seen as a predictor of the actual use and the lower score, together with the qualitative data, implies that this threshold is for the actual offering of coding skills. Schools should therefore invest more in ICT infrastructure and methods to offer programming skills. In order to get a good picture of what is required to increase the facilitating conditions, it is advisable to obtain deeper insights into the role of facilitating conditions in the follow-up research with regard to the actual provision of coding skills. This can be done by conducting in-depth interviews with teachers.

5. REFERENCES

Davis, F. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 319-340.

Teo, T. (2011). Factors Influencing Teachers' Intention to Use Technology: Model Development and Test. *Computers and Education, 57*(4), 2432-2440.

Venkatesh, V., & Morris, M. G. (2000). Why Don't Men Ever Stop to Ask for Directions ? Gender, Social Influence, and their Role in Technology Acceptance and Usage Behavior. *MIS Quarterly, 24*(1), 115-139.

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly, 27*(3), 425-478.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and STEM/STEAM Education

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

An Empirical Study on STEM Learning Satisfaction and Tendency for Creativity of Chinese Secondary School Students

Wangwei LI[1*], Chun CHEN[2]
[1 2]South China Normal University, China
et-lww@qq.com, 929158652@qq.com

ABSTRACT

Based on the STEM learning practice in the secondary school, the study uses the method of investigation to explore the satisfaction of STEM learning in junior high school students at Guangzhou of China. Then the researcher analyzes the workshop-based STEM learning process and validates the effectiveness of workshop-based STEM learning to enhance students' creative thinking and ability. The result shows that the Chinese students' STEM satisfaction is generally at a high level, and there is no significant gender difference. Taking the workshop-based STEM learning as an example, we find that the context design by teacher, the student's independent inquiry, collaborative learning and the integrated practice of technology enhanced have helped students to improve their thinking and ability; Moreover, the student's self-efficacy as a mediator of students' participation in STEM learning and practice catalyzes the realization of students' workshop-based STEM learning process and the goal of improving creativity.

KEYWORDS

STEM learning, satisfaction, tendency for creativity, self-efficacy

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

中国中学生 STEM 学习满意度与创新力倾向的实证研究

李王伟[1*]，陈醇[2]
[12]华南师范大学教育信息技术学院，中国
et-lww@qq.com，929158652@qq.com

摘要

本研究立足中学基于STEM教学实际，采用调查研究法，调查广州某区初中生STEM学习满意度，解析基于工作坊的STEM学习过程，验证基于工作坊的STEM学习于提升学生创新思维和能力的有效性。研究表明，中国学生的STEM满意度整体处于较高水平，且男女生并无显著差异；以基于工作坊的STEM学习为例，教师情境设计、学生自主探究、合作学习和技术增强的综合实践促成了学生创新思维和能力提高；自我效能感作为学生参与 STEM 学习与实践的中介要素，促进了学生STEM—工作坊学习过程和创新力提升目标的实现。

关键字

STEM 学习；满意度；创新力倾向；自我效能感

1. 研究缘起

随着全球教育从"教育创新"向"创新教育"转变，缘起于提升国家竞争力和培养科技人才目标的STEM教育因在培养STEM领域人才方面的卓越成效受到国内外教育政策制定者、一线教师和家长的欢迎（Marginson et al, 2013）。各国和地区因其教育理念、教学环境、人才目标和文化实践的不同，其STEM教育方式和途径呈现出丰富多样、各具特色的多元化形式，但清晰、可供参考的 STEM 教育理念、政策和实施普遍缺乏。

1.1. 促进创新力培养的 STEM 学习方式

近来，涉及跨领域、多学科及主动学习特征的STEM教育显现出跨学科学习的优势，学生通过科学、技术、工程和数学等多元学科媒介，寻求现实问题的解决方式，进行创新合作设计与实践，激发其创造性学习意识、创新能力及素养（Boy, 2013）。研究者基于美国八大 STEAM 教育案例的目标、内涵、实施路径及评价方式分析，洞察了学业成就与创新力培养并重的 STE（A）M 教育新方式。肯尼迪小学通过社区 STEM 空间营造跨学科情境，设计社区STEM创作活动，培养学生计算思维（CT）和批判性与创造性思想家（李王伟等，2018）。教育部于 2017 年 3 月颁布《义务教育小学科学课标准》，明确了中小学STEM学习于培养学生创新能力与素养的核心价值。基于众多理论论证与实践已经证明，STEM 作为一种创新力培养的新学习方式有其独特优势。美国科学教师协会会长 Juliana Texley 教授认为STEM教育是一种与生活紧密联系、注重工程实践与创新能力培养的跨学科学习方式（姜新杰，2017）。

1.2. 基于工作坊的 STEM 学习路径

研究者统整STEM学习方式的知识、能力、思维、创新和价值目标，借鉴工作坊式"学习社区"优势，构建基于工作坊的STEM学习路径，提供学生真实的STEM学习体验，达成创新能力及素养培育目的（李王伟等，2018）。其活动过程如下图 1 所示。

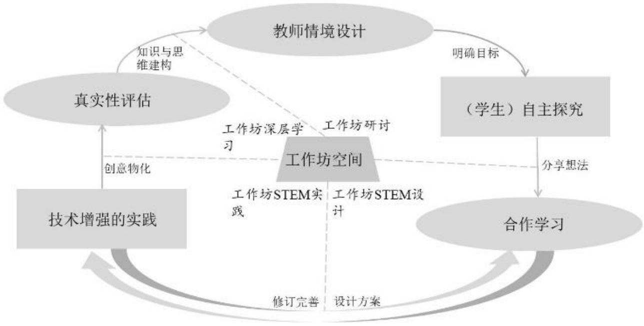


图 1 基于工作坊的 STEM 学习活动过程

然而，基于工作坊的STEM学习五要素于学生创新力培养的有效性，STEM 学习成效的关键要素有哪些？管窥当前国内外STEM教育学界，其并无明确的界定和研究论据，研究者需要基于中国中小学教学实际展开探索和实践，找寻STEM学习作为一种提升学生创新力的学习方式的实证逻辑。

1.3. 研究问题

本研究立足广州地区中学 STEM 教学实际，分析 STEM 学习方式组成要素，实施基于工作坊的STEM学习模式。为深入分析 STEM—工作坊学习过程于培养学生实践与创新能力的有效性，本研究主要聚焦以下几个 关键问题：

（1）学生对基于工作坊的 STEM 学习满意度和创新力倾向提升如何？

（2）基于工作坊的 STEM 学习中哪些过程元素会影响学生创新力培养的关键成效，它们之间的关系如何？

（3）国内中学生 STEM 学习是否存在性别差异，男生 STEM 学习创新力倾向与女生有无显著差异？

2. 研究设计

作基于先前研究，研究者从STEM教育理念与方式，以及目标、过程和评价维度，设计了基于工作坊的 STEM 学习路径模式，包括教师情境设计—自主探究—合作学习—技术增强综合实践—真实性评估五过程（如图 1 所示）。

2.1. 研究假设

研究者通过 STEM 学习评估初步证实学生创新力倾向提升成效，为进一步探讨工作坊学习与学生创新力培养实证关系，在预调查学生 STEM 学习满意度和创新力倾向基础上，针对本研究问题作出假设：

（1）H_1：学生对基于工作坊的 STEM 学习整体满意度较高，表现出较为明显的实践能力与创新力倾向提升成效。

（2）H_2：基于工作坊的 STEM 学习的核心要素：教师情境设计、自主探究、合作学习、技术增强实践及真实性评估与学生的自我效能感和实践与创新能力的提升显著相关，其中自我效能感作为学生工作坊实践和创新力倾向的中介变量显著影响学生 STEM 学习满意度与创新力倾向提升，各要素关系如图 2 所示：

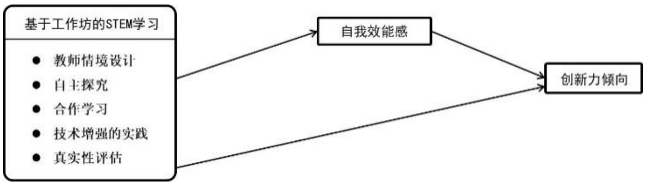


图 2 STEM—工作坊学习要素关系假设

（3）H_3：国内中学生 STEM 学习满意度和创新力倾向并无明显性别差异，男女生 STEM 学习成效无显著相异性。

2.2. 研究方法

本研究选取广州市某区六所学习基础学情无显著差异的初级中学，开展为期两学期的基于工作坊的 STEM 学习实践。研究重点关注六所中学 STEM—工作坊学习评估过程和学生满意度，并就 STEM 学习满意度与教师、学生进行初步访谈，预设 STEM 学习目标倾向和调查开展条件，促成教师和学生参与 STEM 满意度及创新力倾向调查意愿。根据基于工作坊的 STEM 学习一般过程和六所学校科学教师 STEM 教学实践，研究者采用里克特五等级量表方式，设计 STEM 学习满意度问卷，并改编威廉斯创造力倾向量表（威廉斯，2003）。在对其中两所学校 17 名学生进行满意度问卷和量表预调查后，研究者确定了包括教师情境设计、自主探究、合作学习、技术增强实践、真实性评估及自我效能感等六大项的 STEM 学习满意度问卷和威廉斯创新力倾向量表，每个维度包含 3-4 个条目，以概括学生 STEM 学习过程。

研究者与各学校 STEM 教师一起采用线上与线下相结合的方式向全部 547 名学生进行调查。所有问卷均得到全部回收，剔除无效问卷 19 份和量表 17 份，为保证学生 STEM 学习与创新力倾向测量对应，研究者继续剔除无效问卷中其他学生的威廉斯创新力倾向量表，共得 528 份 STEM 学习满意度问卷和威廉斯创新力倾向量表。调查对象全部来自六所初级中学，其中男生 261 名，占比 49.4%，女生 267 名，占 50.6%。先前预调查学生满意度问卷的总体 α 信度系数值为 0.976>0.9，因而说明研究数据信度质量较高。教师情境设计、自主探究、合作学习、技术增强的实践、真实性评估及自我效能感的 α 信度系数分别为 0.930，0.945，0.950，0.951，0.939，0.942，其均大于 0.8 临界值，表明 STEM—工作坊学习满意问卷信度较高。此外，基于工作坊的 STEM 学习满意度效度检验的 KMO 值为 0.978>0.6，且问卷 Bartlett 球形检验值 P<0，05，表明学生 STEM 学习满意度问卷效度较高，威廉斯创新力倾向量表信效度与此相似。

本研究采用多变量统计分析、显著性差异检验与中介效应建模法，关注学生 STEM—工作坊学习过程满意度及其与学生创新力倾向提升关系。问卷数据和量表分析主要从多元变量统计分析、探索性因子分析（EFA）和验证性因子分析（CFA）三方面展开（Kline, 2011），分析条目包括学生基本信息和满意度 23 项及创新力倾向量表 32 项，将满意度问卷变量归纳为性别、教师情境设计、自主探究、合作学习、技术增强实践、真实性评估及自我效能感等七类条目，威廉斯创新力倾向量表得分归为一类条目。

研究聚焦学生 STEM 学习满意度各要素变量分析，运用探索性因子分析 STEM 学习各要素满意度的显著性差异及其与学生创新力倾向解释关系，利用验证性因子分析建立 STEM—工作坊学习要素与学生创新力倾向提升结构关系模型，验证已有研究假设中自我效能感的潜变量关系。

另外，对国内中学男女生 STEM 学习有无显著性差异问题，研究者采用多组路径分析法，从创新力倾向提升到学习满意度差异进行全面测定。采用 1 和 0 分别标记问卷男、女生变量，以探测学生 STEM—工作坊学习及态度的性别差异。

3. 研究结果

研究者将所得 528 份 STEM 学习满意度问卷数据按各要素变量录入 SPSS22.0 软件，统计与之相对应的学生创新力倾向量表得分，并将其单独作为一维度纳入 STEM 学习满意度 SPSS 数据分析集中，进行各要素效应量和因素统计分析，阐释学生 STEM—工作坊学习满意度及创新力倾向提升情况。

3.1. 描述性分析

研究首先对 STEM 学习满意度的六个要素变量及创新力倾向得分进行描述性统计，采用五等级里克特量表对学生的满意度进行编码，分别将“完全同意，同意，部分同意，不同意和完全不同意”对应赋值“5 分，4 分，3 分，2 分和 1 分”，并将全部 528 名学生数据录入 SPSS22.0 软件，利用其描述性统计分析功能测定结果如下表 1 所示。

表 1 描述性统计分析

	平均值	**标准差**	**计算偏度系数**	**峰度系数**
教师情境设计	4.32	0.71	1.840	2.460
自主探究	4.54	0.76	1.460	1.975
合作学习	4.44	0.79	1.480	2.019
技术增强的实践	4.41	0.82	1.479	2.041
真实性评估	4.43	0.79	1.449	2.032
自我效能感	4.47	0.87	1.451	1.943
创新力倾向	143.50	1.67	2.247	2.576

3.2. 结构方程模型建立

为验证 STEM—工作坊学习过程要素与学生创新力倾向提升关系的研究假设，研究者运用验证性因子分析检验假设模型（如图 2）的拟合度，通过 Mplus7.0 工具的结构方程模型（SEM）分析基于工作坊的STEM学习过程与创新力倾向关系假设模型，观察“自我效能感”与“创新思维”的潜在变量效应，利用 SPSS22.0 软件分析 STEM 学习过程各要素交叉协方差及相关矩阵，以确定结构方程模型的拟合度，如下表 2 所示。

表 2 基于工作坊的 STEM 学习要素复合变量矩阵

	教师情境设计	自主探究	合作学习	技术增强的实践	真实性评估	自我效能感
教师情境设计	—					
自主探究	0.137**	—				
合作学习	0.128**	0.173**	—			
技术增强的实践	0.176**	0.247**	0.126**	—		
真实性评估	0.324**	0.129**	0.217**	0.347**	—	
自我效能感	0.197**	0.215**	0.133**	0.143**	0.217**	—
创新力倾向	0.043*	0.021*	0.098**	0.040*	0.132**	0.08**

注：**p < .01; *p < .05.

如表 2 所示，STEM 学习要素变量中教师情境设计、合作学习与自我效能感之间呈统计学上的显著相关；创新力倾向与（学生）自主探究、真实性评估及自我效能感显著相关，但教师情境设计和合作学习及技术增强的实践与学生创新力倾向无显著相关关系。表明基于STEM学习初始效应量建立的关系模型与统计数据拟合度良好。需要明确的是，除教师情境设计和技术增强的实践与学生创新力倾向交叉影响外，所有变量的路径相关系数都呈显著相关。此模型拟合指数为 $\chi^2/df = 3.130/2$， CFI 值为 0.994（CFI>0.90），为可接受范围，进一步证明当前模型很好地符合数据关系。

此外，基于深度学习和创新能力培养成效研究发现，学生的创新思维是创新力激发和提升的重要机制。本研究后续将学生工作坊学习作品从创意与思维创新视角由六位 STEM 教师合作进行评分（5 等级评分制），并将所得分纳入学生 STEM 学习满意度“创新思维”维度进行关系验证。学习理论表明，将自我效能感与创新思维作为学生创新力倾向提升的中介变量与 STEM—工作坊学习要素及成效关系模型具有潜在一致性。研究者通过 Mplus7.0 中的 Delta 方法检验了自我效能感和创新思维作为 STEM—工作坊学习中介效应量的显著性，如下表 3 所示。

表 3 STEM—工作坊学习中介效应量检验

STEM 学习与创新力倾向路径效应分析	参数估计值
从教师情境设计至学生创新力倾向	
教师情境设计—自我效能感—创新力倾向	0.019
教师情境设计—创新力倾向	-0.132
从合作学习至创新力倾向	
合作学习—自我效能感—创新力倾向	0.027
合作学习—创新力倾向	0.120
从自主探究至创新力倾向	
自主探究—创新力倾向	0.021
从技术增强的实践至创新力倾向	
技术增强的实践—创新思维—创新力倾向	0.050
技术增强的实践—创新力倾向	0.201
从真实性评估至创新力倾向	
真实性评估—创新思维—创新力倾向	0.031
真实性评估—创新力倾向	0.113
从自我效能感至创新力倾向	
自我效能感—创新力倾向	0.014

注：**p < .05;

如表 3 所示，STEM—工作坊学习与创新力倾向相关的直接效应变量均显著相关，包括从教师情境设计、合作学习、自主探究、技术增强的实践，以及真实性评估均与学生创新力倾向紧密相关，其效应量分别是-0.132，0.120，0.021，0.201 和 0.113。进一步检验发现，自我效能感、创新思维作为 STEM—工作坊学习的中介效应量与学生创新力倾向显著相关，二者与 STEM—工作坊学习要素及创新力倾向的双向交叉分析同样显著关联，其参数估计值小于直接效应量值，证明自我效能感与创新思维作为 STEM—工作坊学习效应量与创新力倾向提升显著影响因素的合理性。此外，中介效应检验证明，自我效能感与创新力倾向呈直接显著关系，其路径效应量为 0.014（**p < .05），表明其为 STEM—工作坊学习效应中介变量的合理性。

同时，结构方程模型拟合优度的卡方值 P 为 0.052，表明融入中介效应的结构关系模型拟合优度更好。研究最终采取融入中介效应变量的路径结构模型为 STEM—工作坊学习与创新力倾向关系模型，如下图 3 所示。

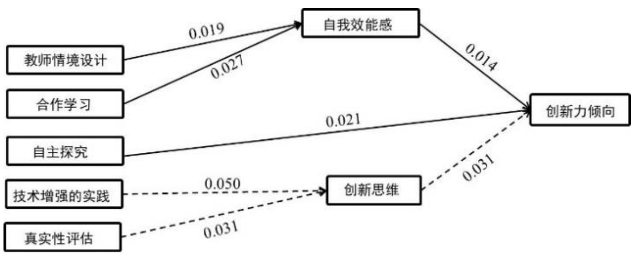


图 3 STEM—工作坊学习要素与创新力倾向关系模型

3.3. 男女生性别差异的多组路径分析

本研究关注国内中学是否存在如国外普遍报告的男女生 STEM 教育成效差异问题。研究采用多组路径分析法，

对学生STEM学习满意度问卷设定男女生性别标记，比较不同性别学生总体STEM学习满意度和创新力倾向差异及STEM—工作坊学习各要素差异。先前调查结果显示，参与STEM学习的男女生数量相近，无显著差异，其中男生261、女生267名，为后续不同性别间的多组路径分析提供前提条件。研究者利用 SPSS22.0 软件的方差分析和卡方检验及 Mplus7.0 的潜变量路径分析比较男女生STEM—工作坊学习过程及创新力倾向提升差异，结果如下表4所示。

表 4 男女生STEM学习满意度与创新力倾向比较

不同性别的变量比较	卡方检验（χ^2）	方差分析（P）
STEM 学习总体满意度	—	0.245
STEM 学习创新力倾向	—	0.136
STEM 学习各要素满意度	1.481/5	0.157
STEM 学习各路径效应	1.089/5	0.104

注：$^{**}p < .01$; $^{*}p < .05$

如表4所示，以男女生性别因素为自变量，研究者探析了学生STEM学习总体满意度和创新力倾向差异，方差分析结果分别为 P=0.245>0.05、P=0.136>0.05，表明男女生在STEM学习满意度及创新力倾向总体得分上无显著差异；在基于工作坊的STEM学习满意度调查中，男女生得分数据卡方检验 χ^2值为 0.370，P值>0.05，说明男女生STEM—工作坊学习过程满意度不存在显著差异；男女生STEM学习各路径效应（包括从教师情境设计到创新力倾向的直接效应至融入自我效能感与创新思维的中介效应路径）卡方检验平均值为 0.272，P>0.05，表明男女生STEM学习与创新力倾向路径效应不存在显著差异。

4. 研究结论

本研究为探测国内中学生STEM学习满意度和创新力倾向提升成效，基于广州某区中学教师和学生访谈及STEM作品评价，设计了基于工作坊的STEM学习满意度问卷，改编了威廉斯创造力测验表以测定学生STEM—工作坊学习创新力倾向提升成效。研究证实了基于工作坊的STEM学习对学生满意度与创新力倾向提升的有效性，并从STEM学习过程变量交叉分析角度验证了学生的自我效能感和创新思维与学生创新力倾向提升的显著相关性。

4.1. 学生 STEM 学习满意度及创新力倾向的显著提升

研究者将所得满意度问卷数据及创新力倾向量表得分纳入 SPSS22.0，完成描述性统计分析，结果如表 1 所示。学生在基于工作坊的STEM学习中教师情境设计、自主探究、合作学习、技术增强的实践、真实性评估及自我效能感满意度平均值均高于 4 分，对应问卷“满意”态度，标准差均在0.8左右浮动，且峰度系数与偏度系数均在合理范围内，排除显著变异项影响。数据表明，学生对STEM—工作坊学习过程呈总体满意。研究初步证实STEM—工作坊学习六要素对于学习成效的积极影响，尤其是学生的创新思维、实践能力和团队合作精神得到显著提高。改编后威廉斯创新力倾向量表学生得分统计显示，学生创新力倾向得分平均值为143.5/160 分，显著高于威廉斯创造力测试的常态值，表明STEM—工作坊学习过程对中学生创新力倾向提升有积极影响，即STEM学习过程与学习满意度呈正相关，对学生创新力倾向提升有积极影响，其表现为教师情境设计、合作学习与技术增强的实践等STEM核心过程满意度与创新力倾向的同步提升。

4.2. 促进创新力倾向提升的 STEM 要素关系模型建立

本研究结合已有STEM课程过程分析和Han等人对学生项目学习过程变量的调查，初步构建出STEM—工作坊学习过程与创新力倾向关系模型（Han, 2017）。随后，研究者将“自我效能感”作为中介变量对STEM学习过程六要素及创新力倾向进行变量交叉检验，建构STEM学习过程要素复合变量矩阵和STEM—工作坊学习要素与创新力倾向关系模型（如图3所示）。结果表明，自主探究、技术增强的实践及真实性评估等STEM过程要素均与创新力倾向显著相关；教师情境设计和合作学习与创新力倾向关系显著性低于其他要素，但对自我效能感有积极作用；同时自我效能感与创新力倾向的交叉检验表明，自我效能感对创新力倾向有显著影响。根据皮亚杰认知发展论和布鲁纳发现学习论，研究者进一步假定STEM学习中教师情境设计和合作学习通过增强学生的自我效能感进而提升创新力，并运用结构方程模型拟合法，进行STEM—工作坊学习与创新力倾向路径交叉分析。研究证明，自我效能感与创新思维作为STEM—工作坊学习与创新力倾向的中介效应量，对学生创新力及价值的提升具有积极影响；

4.3. 男女生 STEM 学习满意度及创新力倾向的性别差异

基于国外 K12 教育普遍存在的 STEM 学习性别差异，研究者就STEM—工作坊学习满意度及创新力倾向，调查了广州某区六所中学参与STEM学习的男女生，并运用多组路径分析法，借助 SPSS22.0 和 Mplus7.0 软件的交叉经验和方差分析功能，测定了中学男女生STEM—工作坊学习的总体满意度和创新力倾向差异、STEM学习创新力倾向提升路径及总体效应（如表4所示）。调查表明，男女生STEM学习总体满意度和各要素满意度均未出现明显差异，但男女生STEM学习路径效应存在小幅差异。即男女生在STEM满意度和创新力倾向上总体无显著差异，但男女生在达成STEM共同核心目标—创新力培养路径上存在细微差异，这部分是由于域内中学生班级集体授课与合作学习造成的，男女生总体上能达成共同的学习目标，但各自认识与实践优势不同，导致上述STEM学习路径的微小差异。

5. 未来研究与展望

本研究致力于国内中学生STEM学习过程及成效进行满意度调查和创新力倾向测定，利用 SPSS22.0 统计与交

叉分析与 Mplus7.0 结构方程模型方法探讨了基于工作坊的 STEM 学习过程要素与创新力倾向关系及男女生 STEM 学习的性别差异，为国内中小学 STEM 有效教学模式建构和教学原则应用提供示范。然而，本研究依然存有局限，也是未来研究努力致于的方向。一是本研究选取的调查样本集中在广州市，样本分布没有兼顾国内其他 STEM 教育实践地区，样本代表性不够充分，未来本研究将选取上海、陕西、长春、深圳等中小学 STEM 教育实践地区展开研究，进行地区间差异对比，以为区域 STEM 教育发展提供建议。二是研究着力于学生创新力倾向的提升，对学生创新力实践及价值促进的解释力并非完全充分，未来研究将关注学生创新力倾向转化创新思维及实践程度，以比较两者结果的差异，为 STEM 学习设计与创新力培养目标的实现提供充足证据。

6. 参考文獻

李王伟和徐晓东（2018）。统整艺术与 STEM 实践的创新力培养——来自美国八大 STEAM 教育案例的启示。**外国中小学教育**，**12**，9-17。

姜新杰（2017）。国际视野下的 STEM 发展新动向。**上海教育**，**34**，56-59。

李王伟和徐晓东（2018）。综合学习视野下的 STEM 教育:工作坊学习。**基础教育参考**，**18**，35-38。

威廉斯（2003）。威廉斯创造力倾向测量表。**中国新时代**，**2**，89-90。

Boy, G. A. (2013). From STEM to STEAM: Toward a Human-centred Education, Creativity & Learning Thinking. *Proceedings of the 31st European Conference on Cognitive Ergonomics*. New York: ACM.

Han, S. (2017). Korean Students' Attitudes toward STEM Project-based Learning and Major Selection. *Educational Sciences: Theory & Practice*, *17*, 529-548.

Kline, R. B. (2011). *Principles and Practice of Structural Equation Modeling (3rd ed.)*. New York: Guilford Press.

Marginson, Tytler, & Russell, et al. (2013). STEM: Country Comparisons. Retrieved January 6, 2019, from http://dro.deakin.edu.au/view/DU:30059041

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Using a 6E Model Approach to Improve Students Learning Motivation and Performance about Computational Thinking

Hsien-sheng HSIAO[1], Jyun-chen CHEN[2*], Yi-wei LIN[3], Hung-wei TSAI[4]
[134] Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taiwan
[1]Institute for Research Excellence in Learning Sciences, National Taiwan Normal University, Taiwan
[1]Chinese Language and Technology Center, National Taiwan Normal University, Taiwan
[2]Research Center for Testing and Assessment, National Academy for Educational Research, Taiwan
etlab.paper@gmail.com, cv999999999@yahoo.com.tw, gn02294997@gmil.com, vincent0965@gmail.com

ABSTRACT
6E learning model has used in a learner-centered teaching model that enhances students' motivation and learning outcomes. Robot education combines a lot of the knowledge and skills of a variety of learning areas to enhance student understanding and validation through the 6E learning model, and allows students to learn computational thinking through robotics. This research is aimed at 70 senior students in an elementary school. The Arduino robot is designed and assembled through practical curriculum. This study used experimental design with two different groups: 6E learning mode and general learning mode. The research found that using the 6E learning mode would enhance students’ motivation and computational thinking performance in the robotic practice curriculum, and enhance the use of learners in practical operations.

KEYWORDS
6E model, robot education, performance curriculum, computational thinking

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

6E 學習模式結合機器人教育對學習動機與運算思維學習成效之影響

蕭顯勝 [1]，陳俊臣 [2*]，林奕維 [3]，蔡宏為 [4]
[134]科技應用與人力資源發展學系，臺灣師範大學，臺灣
[1]學習科學跨國頂尖研究中心，臺灣師範大學，臺灣
[1]華語文與科技研究中心，臺灣師範大學，臺灣
[2]測驗及評量研究中心，國家教育研究院，臺灣
etlab.paper@gmail.com，cv999999999@yahoo.com.tw，gn02294997@gmil.com，vincent0965@gmail.com

摘要

6E（Engage、Explore、Explain、Engineer、Enrich、Evaluate）學習模式時常被使用在以學習者為中心的教學模式，可以提升學生的學習動機以及學習成效。而機器人教育結合多種不同學習領域的知識和技能，透過 6E 學習模式增進學生的理解以及驗證，並讓學生以機器人教學課程學習運算思維。本研究針對 70 位國小高年級的學生，透過實作課程設計組裝 Arduino 機器人，採準實驗設計，分別使用：6E 教學模式、及傳統教學模式。研究結果發現使用 6E 教學模式結合機器人教育，會增進學生在實作課程的學習動機及運算思維學習成效，並提升學習者的實作能力。

關鍵字

6E 模式；機器人教育；實作課程；運算思維

1. 前言

現今全球資訊科技蓬勃發展的趨勢之下，許多新興科技應運而生，機器人技術可以幫助學生將原本抽象的概念化成現實（Heiner, 2018）。讓學生更投入參與課程，增進他們的好奇心，並培養學生的批判性思考。為此，從中小學階段開始加入機器人課程能讓學生與世界潮流接軌，而機器人所結合的各項科學領域亦有助於學生學習和生活，增加學生解決問題的能力（Grubbs & Strimel, 2018）。

近年來運算思維受到非常多的重視，凡舉資料分析、程式語言、演算法之設計等皆需要運用運算思維之能力（許庭嘉、陳子潔、施詠恬、邱于軒、王韻茹、諸恩琳、張韶宸，2017）。運算思維可以使學生學習如程式一樣的邏輯思考方式，並將日常生活中的問題應用於此。在國中小階段的生活科技主要是希望學生透過實作課程來引發學生的學習興趣。臺灣的科技教育主要強調「做中學」，也是全球科技教育的核心理念（張玉山、張雅富和陳冠吟，2016）。

透過動手實作可以進一步的提升國小學生的手眼協調能力、設計創意能力、學習動機與興趣。把適合用在實作課程的機器人教育及 6E 教學模式的搭配，以學生當作學習的中心主角，可以將各領域的知識加以整合（Connor, Karmokar, & Whittington, 2015）。因此，本研究結合國小生活科技以及機器人的實作課程，以 6E 教學模式為核心，藉由老師帶領學生們學習科學知識、操作機械器具、設計組裝機器人以及增進學習興趣。希望透過此課程增進學生對機器人學習的學習動機及運算思維學習成效。

2. 文獻探討

2.1. 6E 教學模式

6E 教學模式在 2013 年由美國國際科技與工程教師學會（International Technology and Engineering Educators Association, ITEEA）提出，透過 6E 模式可以培養學生的設計與探究能力（Barry, 2014；Burke, 2014）。6E 教學模式的階段包含：參與、探索、解釋、實作、深化、評量，6E 模式之定義與各項說明如表 1。透過教師的引導，幫助學生善加利用課堂上的概念、實務經驗，融入設計以及探索式教育等理念。讓學生能夠將知識進行自我構築並內化成為自己的知識。

表 1 6E 模式說明

6E 模式	具體說明
參與（Engage）	激發出學生的學習興趣，引起學生的好奇、興趣進而投入。
探索（Explore）	讓學生建構自己對於面對教學活動的理解，依據教學建立架構。
解釋（Explain）	讓學生學習解釋與定義他們所學，並思考先前所學的意義及價值。
實作（Engineer）	讓學生運用概念、實務經驗以及發展，對實作活動主題的深度理解所學知識應用，以獲得深層的理解。
深化（Enrich）	讓學生有機會運用教學活動的範例，更進一步應用所學解決複雜有難度的問題，並藉此深化學習經驗讓學生做更深入的學習。
評量（Evaluate）	提供教師和學生確認在學習歷程中，讓師生瞭解學習的效果。

2.2. 實作課程

動手實作長期以來被認為能夠促使學生學習的關鍵因素之一（Satterthwait, 2010）。杜威認為「所有的教育都由經驗中產生」（Dewey, 1938）。實作課程可以建構出學習的架構與觀念，透過充滿娛樂性的課程可引人入勝（Giannakos, Divitini, & Iversen, 2017）。學習者親身體驗生活，並在實作的過程中，得以嘗試錯誤而學習到系統性思考。動手做的過程有助於增進學習動機、學習成效（Arangala, 2013）。

而在程式設計相關的實作課程部分，許多程式語言的初學者時常對程式語言的設計與編寫感到困惑（Denny, Luxton-Reilly, Tempero, & Hendrickx, 2011）。也有部分

學者指出，最讓初學者困惑的多半並非程式語法上的問題，事實上最讓程式學習感到困難的是如何設計整個結構與流程（Tan, Ting, & Ling, 2009）。

2.3. 機器人教育

機器人教育整合了電機、機械、資訊、通訊、電子、能源、材料及創意內容等領域的知識與技術應用（陳怡靜、張基成，2015），在機器人教育中，擁有一個良好的媒介來結合教育與機器人領域是非常重要的，而這也是現行教育體系中較缺乏的部分。在過去的幾年當中，影響學生學習資訊教育的因素非常多，而且也會影響學生在 STEM 學科中的行為（Pappas, Aalberg, Giannakos, Jaccheri, Mikalef, & Sindre, 2016）。

許多歐美國家皆認為學習程式設計不僅對個人發展很重要，連帶的也影響國家競爭力。程式設計的課程並不是期望讓小學生成為程式設計工程師，而是培養運算思維以及解決問題的能力。在 21 世紀，不只是更加強 STEM 領域的教學，也希望可以針對 STEM 領域上較不感興趣的學生，吸引這些學生對 STEM 領域的興趣（Mavroudi, Giannakos, Divitini, & Jaccheri, 2016）。學習程式設計可以讓學生透過分析、計畫、選擇和執行的過程中形成正確的邏輯思考過程，帶動學生提高思考層次，以更加清晰、正確、可行的方式將問題拆解並運用在真實世界當中（Feng & Chen, 2014）。

2.4. 運算思維

運算思為的概念最早被提出是以模組化合將問題分解成許多小問題來解決的方式（Wing, 2006）。比起程式教學，我們更應該教導學生以更有效率的方式解決問題（García-Peñalvo, 2018）。我國在近年也強調培養學生的邏輯能力，希望學生可以透過動手實作、有效運用運算思維之概念來培養系統性思考之概念與過程（林育慈和吳正已，2016）。

運算思維近幾年在許多國家的教育策略中越來越重要，教學上可以使用模組化的課程設計，建立學生對運算思為的學習經歷（陳怡芬、林育慈和翁禎苑，2018）。運算思維所著重的要點都是要讓學習者可以學習如何更好的發現並解決問題（Swaid, 2015）。最近許多關於運算思維的教學上，許多教育者結合許多不同類型的學習材料，例如遊戲、機器人、應用程式或是其他可穿戴式裝置（Kafai & Burke, 2014）。

3. 研究方法

3.1. 研究架構

本研究使用 6E 教學模式結合 Arduino 的機器人課程，探討學習動機與學習成效。內容包含電子元件的認識、感測器的應用、創意設計以及工具操作能力，本課程將程式設計以及 Arduino 進行機電整合教學，分別使用 6E 教學模式以及傳統教學模式融入課程。研究架構圖如圖 1 所示。

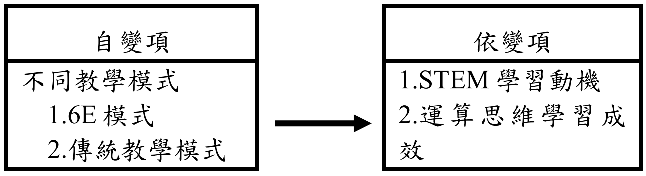


圖 1 研究架構圖

3.2. 研究對象

本研究對象為程式語言初學者，具有基礎資訊科技操作能力，但是沒有程式語言學習經驗之學習者。年齡介於 12 至 13 歲之間共 70 人，分為實驗組兩班、對照組兩班，接受為期 16 週，每次 80 分鐘的課程。

3.3. 研究設計與實施

選取新北市八里區某國民小學高年級學生，共四個班級，70 位學生，分為實驗組兩班、對照組兩班，進行學習動機和實作知識成效前測。研究設流程構如圖 2 所示。本研究實施步驟如下：

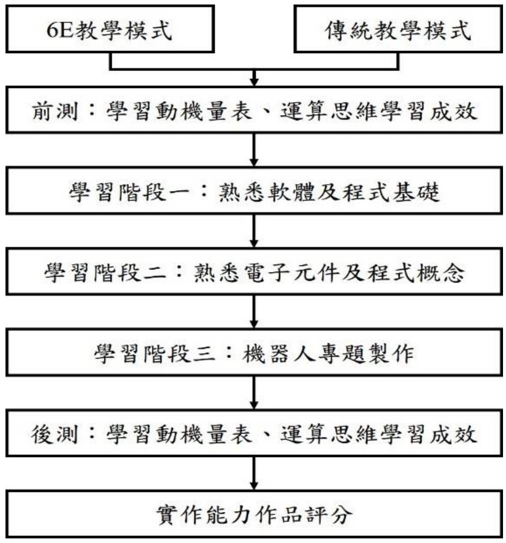


圖 2 研究流程圖

（1） 課程學習階段一的學習目標是熟悉 Scratch for Arduino 軟體操作和程式基礎，包含循序結構、迴圈結構、條件結構和變數，每週 80 分鐘，共 5 週。

（2） 課程學習階段二的教學活動，學習目標是認識和熟悉 Scratch for Arduino 擴充板的電子元件以及如何應用電子元件的程式基礎，每週 80 分鐘，共 6 週。

（3） 課程學習階段三「螃蟹機器人」的設計製作，學生必須應用前面所學，透過主題創意發想、腳本設計、撰寫程式和組裝硬體、測試與修改設計出：Scratch for Arduino 程式、電子元件的應用、螃蟹外觀造型。

（4） 後測階段，進行作品的成果發表以及學習動機和實作知識成效後測，測驗時間為一節課，共

40 分鐘；接著繼續進行運算思維成效後測，測驗時間為一節課，共 40 分鐘。

3.4. 6E 教學模式活動設計

本研究之實驗組每個教學活動皆採用 6E 模式進行機器人教學活動設計，學生將配合教師的教學及搭配學習單的引導輔助學習。實驗學校位於淡水河口，擁有豐富的生態環境，學生必須應用前面所學自行發想主題設計出：（1）Scratch for Arduino 程式；（2）感測器應用；（3）螃蟹的外觀造型，並將學習單發下引導學生發想。學生在此階段必須連結過去所學螃蟹生態的知識，發想有關螃蟹之主題因此以「螃蟹過馬路」為例說明學習階段一與學習階段二 6E 模式的教學。

3.5. 傳統教學模式活動設計

本研究之對照組每個教學活動皆以傳統教學模式進行教學，教師逐步講述，學習者將依照教師示範進行機器人實作課程的學習，學習者屬於被動的接受訊息。在學習階段一與學習階段二的每個教學活動中，教師會先執行作品成果並說明教學目標與教學內容，教師講解完成品的做法，接著學生會自行實作教師剛剛講解的內容，此時教師會下去觀察學生實作的情況，若學生遇到困難，教師會重複地講解並提醒學生常做錯的地方。最後，學生必須執行作品成果，若有錯誤則持續進行修正。在學習階段三時，提供螃蟹機器人製作之學習單，與上述 6E 模式一樣依循著主題發想、腳本設計、故事說明步驟進行遊戲的設計，在發想設計的過程中，教師不會參與學生的遊戲設計討論，讓學生自由發揮，接者學生將依照上述所寫的設計內容，將其遊戲實作完成。

3.6. 研究工具

3.6.1. 學習動機量表

Printrich、Smith 與 McKeachie 在 1989 年題所編之「激勵的學習策略量表（Motivated Strategies for Learning Questionnaire, MSLQ）」衡量學生的學習動機。本研究採用國內學者吳靜吉和程炳林（1992）曾對 MSLQ 量表進行翻譯，使該測驗適用於國內使用，後續許多研究亦證明其具有良好的信度及效度。MSLQ 量表題項包含價值成分、期望成分和情感成分，本研究將以價值成分、期望成分為主衡量學習動機。其中價值成分分為「內在目標導向」、「外在目標導向」及「工作價值」三個向度；期望成分又分為「控制信念」、「自我效能」及「期望成功」三個向度。問卷開發經過 75 位六年級學生進行預試，問卷進度之 Cronbach’s α = .87。

3.6.2. 創意作品評分矩陣

創意作品評分矩陣（Creative Product Assessment Matrix, CPAM）的創新性（Novelity）、解決方案（Resolution）及製作與統合（Elaboration & Synthesis）三大指標及九個次指標。最後，學生發表作品成果，說明作品創作的歷程，教師則根據實作作品、學習單內容與口頭報告內容，以 CPAM 標準評量學生作品，定義且規劃適用於本研究之作品評分指標如表 2。

表 2 CPAM 向度與評分標準

向度	指標	評分標準
1.創新性（Novelity）	1.1 原創性（Original）	作品與現有的產品不同
	1.2 驚奇性（Surprise）	作品呈現出意想不到的資訊或效果
2.解決方案（Resolution）	2.1 價值性（Valuable）	作品符合需求
	2.2 邏輯性（Logical）	產品依循正確的規則
	2.3 有用性（Useful）	作品具有明顯的實際應用
	2.4 可理解性（Understandable）	作品可自我解釋並易於理解
3.製作與統合（Elaboration & Synthesis）	3.1 基本品質（Organic）	作品完整且能運行良好
	3.2 精緻程度（Elegance）	作品在各方面的細緻程度
	3.3 良好手藝（Well-crafted）	作品在經過調教後所達到的最佳程度

3.6.3. 運算思維學習成效測驗卷

測驗主要目的是評量學生對整體課程內容知識的了解，包括程式設計、Arduino、電子元件的應用。依認知層次低至高分成知識記憶、知識理解與知識應用，共 25 題，每題 1 分滿分為 25 分。測驗卷內容與有機器人教學經驗的老師討論修改而成，具有專家信度與效度，測驗卷進度之 Cronbach’s α = .70。

4. 研究結果

4.1. 實作知識成效

本研究欲了解不同教學模式間實作知識成效提升之差異，因此採用共變數分析進行比較。依據共變數分析的基本假設，首先需進行迴歸係數同質性考驗。依據同質性考驗結果可知，組內迴歸係數同質性考驗結果未達顯著水準（$F = 1.60$，$p = .21 > 0.05$），表示兩組迴歸線的斜率相同，即表示共變項（實作知識成效前測分數）與依變項（實作知識成效後測分數）之間的關係不會因為自變項各處理水準的不同而有所差異，符合共變數組內迴歸係數同質性假定，可繼續進行共變數分析。

共變數分析結果如表 3，排除前測分數對後測分數之影響後，各組之間的差異達到顯著水準（$F = 9.85$，$p = 0.003 < 0.01$），顯示不同教學模式之實作知識學習成效提升分數具顯著差異。本研究進一步計算效果量（Effect size）且依據 Cohen（1988）及 Cohen（1994）的建議，$.01 < \eta^2 \leq .06$ 時為低度效果量，$.06 < \eta^2 \leq .14$ 時

為中度效果量，$\eta^2 > .14$ 為高度效果量，本研究計算所得之 $\eta^2 = .13$ 介於.06 與.14，為中度效果量。

表 3 不同教學模式在實作知識成效之共變數分析

變異來源	平方和	自由度	平均平方和	F	p	η^2
組別	85.70	1	85.70	9.85^{**}	.003	.13
誤差	582.98	67	8.70			

$^{**}p < .01$

4.2. *運算思維學習成效*

本研究欲了解不同教學模式間運算思維成效提升之差異，因此採用共變數分析（ANCOVA）進行比較。依據共變數分析的基本假設，首先需進行迴歸係數同質性考驗。依據同質性考驗結果可知，組內迴歸係數同質性考驗結果未達顯著水準（$F = 2.17$，$p = .15 > .05$），表示兩組迴歸線的斜率相同，即表示共變項（運算思維成效前測分數）與依變項（運算思維成效後測分數）之間的關係不會因為自變項各處理水準的不同而有所差異，符合共變數組內迴歸係數同質性假定，可繼續進行共變數分析。

共變數結果如表 4 所示，排除前測分數對後測分數之影響後，各組之間的差異達到顯著水準（$F = 14.75$，$p < .001$），顯示不同教學模式之運算思維學習成效提升分數具顯著差異。本研究進一步計算效果量（Effect size）且依據 Cohen（1988）及 Cohen（1994）的建議，$.01 < \eta^2 \leq .06$ 時為低度效果量，$.06 < \eta^2 \leq .14$ 時為中度效果量，$\eta^2 > .14$ 為高度效果量，本研究計算所得之 $\eta^2 = .18$ 大於.14，為高度效果量。

表 4 不同教學模式在運算思維成效之共變數分析

變異來源	平方和	自由度	平均平方和	F	p	η^2
組別	87.87	1	87.87	14.75^{***}	< .001	.18
誤差	399.30	67	5.96			

$^{***}p < .001$

4.3. *實作知識成就*

本研究欲了解不同教學模式間實作能力及各分項表現差異，因此採用變異數分析進行比較。評分由研究者及實驗學校資訊教師採用創意作品評分矩陣（Creative Product Assessment Matrix, CPAM）進行學生作品的評分，進行資料分析並解釋結果，撰寫研究報告。

表 5 實作能力變異數分析檢定表

實作能力		平方和	自由度	平均平方和	F	p	η^2
總分	組間	101.20	1	101.20	5.34*	.02	.55
	組內	1288.67	68	18.95			
	全體	1389.87	69				

$^{*}p < .05$

依據表 5 之變異數統計結果可知，兩組學生在實作能力總分上有顯著差異。本研究進一步計算效果量（Effect size），根據 Cohen（1988）之研究，因實作能力分析為兩母群體平均數之比較，效果量大小則由 η^2 值進行效果量判斷。$.20 < d \leq .50$ 時為低度效果量，$.50 < d \leq .80$ 時為中度效果量，$d > .80$ 為高度效果量（Cohen, 1988; Cohen, 1994），本研究在實作表現總分（$d = .55$）為中度效果量，因此進一步進行上述實作能力的事後比較。

5. 結論與建議

本研究旨在探討不同教學模式（6E 模式、傳統教學模式）對國小高年級學生在機器人實作課程對學習者學習動機、學習成效及實作能力之影響。透過教學實驗發現，機器人課程中比起一般的資訊課能帶給學生有較多的學習內容，不只有軟體操作，還有程式設計與電子元件的知識。除此之外，實作活動的加入也為課程內容更豐富，顯示機器人實作課程能提升學生的學習動機。而 6E 模式以建構主義的學習觀為主，提供一個以學習者為中心的課程模式，並強調真實情境的呈現及實作設計的核心概念。學習者接受 6E 模式進行學習活動，經過自身思考後，會將課程的內容程式概念與實作知識內化成個人認知，在運算思維的測驗時，學生能有效的將思考邏輯應用至情境式題型。機器人教育被證實可增加學生在運算思維上的能力，相較於傳統式的課程，透過機器人等跨領域課程，能夠讓學生學習更高階的邏輯思考能力。因此未來能透過機器人教育，在國小階段開始培養學生運算思考的能力，了解程式語言以及機械原理是如何正確執行指令並解決問題。打破學科之間的隔閡以及建立起共通的橋樑，讓學生在學習相關知識時可以更加得心應手。未來能夠透過類似問題導向式學習法或主題式學習法，來加深學生對於應用學科的靈活性以及實用性。

6. 致謝

感謝臺灣教育部高等教育深耕計畫補助臺灣師範大學（NTNU）學習科學跨國研究中心、華語文與科技研究中心；臺灣科技部專題研究計畫，MOST106-2511-S-003-019-MY3 ， MOST106-2622-S-003-002-CC2 ， MOST106-2511-S-003-049-MY3 ， MOST107-2622-S-003-001-CC2 ， MOST107-2511-H-003-046-MY3 ， MOST108-2511-H-656-001-MY2，支持本研究發展。

7. 參考文獻

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.

張玉山、張雅富和陳冠吟（2016）。Kolb 經驗學習理論於國中機器人活動之教學應用。**科技與人力教育季刊**，**2**（**4**），1-16。

許庭嘉、陳子潔、施詠恬、邱于軒、王韻茹、諸恩琳和張韶宸（2017）。結合擴增實境與運算思維之雷切教具於排序演算法的學習成效分析。 **科技與人力教育季刊**，**4**（**1**），1-14。

陳怡芬、林育慈和翁禎苑（2018）。運算思維導向程式設計教學-以[動手玩音樂]模組化程式設計為例。**中等教育**，**69（2）**，127-141。

陳怡靜和張基成（2015）。兩岸機器人教育的現況與發展。**中等教育**，**66（3）**，37-59。

Arangala, C. (2013). Developing Curiosity in Science with Service. J. *Civic Commit*, *20*, 1-10.

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.

Barry, N. (2014). The ITEEA 6E Learning byDeSIGN™ Model. *The Technology and Engineering Teacher*, *73*(6), 14-19.

Burke, B. N. (2014). The ITEEA 6E Learning ByDesign™ Model: Maximizing Informed Design and Inquiry in the Integrative STEM Classroom. *Technology and Engineering Teacher*, *73*(6), 14-19.

Connor, A. M., Karmokar, S., & Whittington, C. (2015). From STEM to STEAM: Strategies for Enhancing Engineering & Technology Education. *International Journal of Engineering Pedagogy (iJEP)*, *5*(2), 37-47.

Denny, P., Luxton-Reilly, A., Tempero, E., & Hendrickx, J. (2011). *Understanding the Syntax Barrier for Novices. Proceedings of Conference on Innovation and Technology in Computer Science Education*. New York: ACM, 208-212.

Dewey, J. (1938). *Experience and Education.* New York: Macmillan.

Feng, C. Y., & Chen, M. P. (2014). The Effects of Goal Specificity and Scaffolding on Programming Performance and Self-Regulation in Game Design. *British Journal of Educational Technology*, *45*(2), 285-302.

García-Peñalvo, F. J. (2018). Computational Thinking. *IEEE RITA*, *13*(1), 17-19.

Giannakos, M. N., Divitini, M., & Iversen, O. S. (2017). Entertainment, Engagement, and Education: Foundations and Developments in Digital and Physical Spaces to Support Learning through Making. *Entertainment Computing*, *21*, 77-81.

Grubbs, M. E., Strimel, G. J., & Huffman, T. (2018). Engineering Education: A Clear Content Base for Standards. *Technology and Engineering Teacher*, *77*(7), 32-38.

Heiner, C. (2018). A Robotics Experience for All the Students in an Elementary School. *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* ACM, 729-734.

Kafai, Y. B., & Burke, Q. (2014). *Connected Code: Why Children Need to Learn Programming*. Cambridge: MIT Press.

Mavroudi, A., Giannakos, M., Divitini, M., & Jaccheri, L. (2016). *Arenas for Innovative STEM Education: Scenarios from the Norwegian University of Science and Technology*. Retrieved January 2, 2019, from http://stemeducation.upatras.gr/histem2016/assets/files/histem2016_submissions/histem2016_paper_39.pdf

Pappas, I. O., Aalberg, T., Giannakos, M. N., Jaccheri, L., Mikalef, P., & Sindre, G. (2016). Gender Differences in Computer Science Education: Lessons Learnt from an Empirical Study at NTNU. *Proceedings of NIK 2016 conference*.

Satterthwait, D. (2010). Why are 'Hands-on' Science Activities so Effective for Student Learning? *Teaching Science: The Journal of the Australian Science Teachers Association*, *56*(2), 7-10.

Swaid, S. I. (2015). Bringing Computational Thinking to STEM Education. *Procedia Manufacturing*, *3*, 3657-3662.

Tan, P. H., Ting, C. Y., & Ling, S.W. (2009). Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception. *Proceedings of International Conference on Computer Technologies and Development*. IEEE, 42-46.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Robotic Course Designed with CT 3D Model

Feng-shen HE[1], Xiao-qing GU[2*], Yu-he YI[3], Yong OU[4]
[123] Department of Educational Information Technology, East China Normal University, China
[123] Shanghai Engineering Research Center of Digital Education Equipment, China
[4] College of Teacher Education, East China Normal University, China
736523119@qq.com, xqgu@ses.ecnu.edu.cn, 313699389@qq.com, 1042941425@qq.com

ABSTRACT

Based on the three-dimensional model of computational thinking including three dimensions of computational concept, computational practice and computational concept, this study designed a STEM course for robot learning, which was implemented in a robotics interest class of a junior middle school in Shanghai. Computational Thinking test (CTt) was used before and after the course, and the test results were analyzed and discussed. It is found that the robot course designed by the three-dimensional model of computational thinking plays a significant role in the development of students' computational thinking, but there are significant differences in the test results between the high-performance group and the low-performance group. This paper analyzes and discusses this problem.

KEYWORDS

computational thinking, course design, the three-dimensional framework, robotic course

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

基于运算思维三维模型重新设计机器人课程

何沣燊[1]，顾小清[2*]，易玉何[3]，欧勇[4]
[123]华东师范大学教育信息技术学系，中國
[123]上海数字化教育装备工程技术研究中心，中國
[4]华东师范大学教师教育学院 学科教学（物理），中國
736523119@qq.com，xqgu@ses.ecnu.edu.cn，313699389@qq.com，1042941425@qq.com

摘要

本研究基于包含运算概念、运算实践和运算观念三个维度的运算思维三维模型，设计了项目式学习的机器人STEM课程，并于上海市一个初中机器人兴趣班中进行实施。在课程前后使用运算思维测试（CTt）进行前后测，并对测试结果进行分析讨论。发现基于运算思维三维模型设计的机器人课程对学生运算思维的发展具有显著的作用，而从课程中学生高、低表现组测试结果所存在的明显差异表明，进一步印证了基于运算思维三维模型设计的机器人课程的作用，但也反应出在课堂引导和活动设计中需要改进的问题。

关键字

运算思维；三维模型；课程设计；机器人课程

1. 前言

随着运算思维理论和实践研究的不断发展，这项能力已被研究者认为是数字时代的必备技能，成为了当今社会创新型人才的培养目标。机器人课程是运算思维培养的良好课程载体（Fagin & Merkle, 2003），通过让学生设计、制作、开发、运用和评价机器人，能有效提高其动手操作能力，形成其结构良好的问题解决模式（Blanchard, Freiman, & Lirrete-Pitre, 2010），提升其批判性思维等其它高阶思维能力（Chambers, Carbonaro, & Rex, 2007）。现有的面向运算思维和机器人课程的相关研究主要聚焦于不同年龄阶段的实施与应用（Norton, McRobbie, & Ginns, 2007）、以及相关工具的开发与检验（Koh, Basawapatna, Bennett, & Repenning, 2010; Seiter & Foreman, 2013），对于如何设计有效的课程活动体系，以激发学生的创造性思维，培养其问题解决能力和运算思维的研究还较少。基于此，本研究以机器人课程为载体，通过设计有效的教学活动，以期发展学生的运算思维。

2. 文献综述

2.1. 运算思维的概念及理论框架

运算思维最早由Jeanette Wing（2006）提出，被认为是利用计算机科学的基本概念来解决问题、设计方案和理解人类行为的一种能力。随着研究的深入，不同学者和机构对这一定义进行了优化，Aho（2012）将其简化为解决问题过程中涉及的思想过程；国际教育技术协会（ISTE）和计算机科学教师协会（CSTA）则将其界定为包括形象化问题、整理和分析数据、呈现数据、分析解决方案和应用迁移五个层面的问题解决阶段（International Society for Technology in Education and The Computer Science Teachers Association, 2011）。运算思维作为解决真实问题的一套规则和方法，能够帮助复杂问题模块化，将解决方案流程化。为了指导相关活动的开展，以期发展学生在活动过程中的运算思维能力，美国麻省理工学院媒体实验室（MIT）终身幼儿园研究小组（Lifelong Kindergarten Group）提出包含运算概念、运算实践和运算观念（Brennan & Resnick, 2012）三个维度的三维框架。对该框架而言，其各个维度的界定与我们所熟知的信息技术课程知识与技能、过程与方法、情感态度与价值观的三维目标相类似（王旭卿，2014），因而被广泛应用于相关运算思维培养的活动实践中。

运算思维的三维框架详细由以下部分组成：

运算概念（Computational Concept，指设计者在编程时所使用的概念）。

表 1 运算概念包含的概念和解释

运算概念	解 释
顺序	识别完成一个任务的一系列步骤
循环	重复运行相同的顺序多次
并行	使多件事情同时发生
事件	一个事件会促使另外的事情发生
条件	根据条件作出决策
运算符	支持数学和逻辑表达式
数据	存储、检索和更新数据

运算实践（Computational Practices，指设计者在编程中所发展的实践）

表 2 运算实践包含的实践和解释

运算实践	解 释
递增和重复	先开发一点点，然后不断试验，再开发更多的部分
测试和调试	确保可以运行，并发现和修正错误
再利用和再创作	在别人或自己已完成的作品基础上创建新作品
抽象和模块化	通过把更小的部件集合在一起创建更大的作品

运算观念（Computational Perspectives，指设计者形成的有关他们身边世界和他们自己的观念）

表 3 运算观念包含的观念和解释

运算观念	解　释
表达	认识到运算是创作的媒介（我能够创作）
联系	认识到利用和为他人创作的力量（当我与其他人联系的时候我能够做与众不同的事情）
质疑	感觉被赋予提出有关世界的问题的能力

2.2. STEM 教育与机器人课程

STEM 起源是科学（Science）、技术（Technology）、工程（Engineering）和数学（Mathematics）四门学科的缩写，是一种跨学科课程设计的趋势。各个国家纷纷制定相关政策法规，以支持该教育理念的发展。如专门针对 K-12 阶段 STEM 教育的《准备与激励：为了美国未来的 K - 12 科学、技术、工程和数学教育》（Executive Office of the President, 2015）政策文件，和美国国家科技委员会（NSTC）联合其他相关部门机构发布的《联邦 STEM 教育五年战略计划》（The White House, 2013）等。STEM 强调跨学科整合，不仅让学生掌握相关知识和技能，还能进行灵活迁移应用以解决真实世界的问题，发展其创造性思维（余胜泉和胡翔，2015）。

机器人课程是一门典型的 STEM 综合课程，既包含计算机编程、电子嵌入式系统等相关知识，也蕴含工程设计以及数学等相关概念（Flot et al., 2016）。通过机器人课程的开展，不仅能让学生了解计算机编程等技术发展的基础学科，也给学生问题解决、团队协作、时间管理等能力发展提供了平台（王娟、胡来林和安丽达，2017）。机器人教学活动也以其特有的学科交叉融合性和实践操作性，被认为是 STEM 教育理念的实践场，吸引了相关领域研究者的广泛关注。研究者通过将 STEM 教育理念与机器人课堂进行整合，来探究学生在活动开展过程中的能力提升与发展（杜娟和臧晶晶，2014；吴秀凤和陈奕贤，2015；秦换鱼，2016）。这类研究当中，虽然着重探讨了机器人课程中教学活动与 STEM 教育理念的映射关系，并设计了包含情境、操作、应用、总结等活动的课程，但在培养目标上，更关注培养学生创新能力、动手实践能力、问题解决能力等综合能力，一定程度上忽视了机器人课程对学生计算思维培养的价值。

2.3. 机器人课程与运算思维培养

机器人课程的学习在一定程度上与上述运算思维三维框架所描述的操作过程所吻合。学生在设计和搭建机器人程序块过程中，有对任务的顺序和模块化识别（Danli, Tingting, & Zhen, 2014），通过运用相关运算符和表达式，让事件在相应条件下循环（Bers, Flannery, Kazakoff, & Sullivan, 2014）、并行或轮换；通过调试机器人程序，进行相关模块的再创作和整合（Bers et al., 2014）；为了提高机器人的实用性，学生需要基于真实问题情境，理解机器人构造和程序与外部问题情境的联系性（Voogt, Fisser, Good, Mishra, & Yadav, 2015）。研究表明，使用机器人课程进行运算思维培养能有效激发学生的学习动机，提升其学习热情（Fagin & Merkle, 2003; Linder, Nestrick, Mulders, & Lavelle, 2001），促进学生的编程能力、问题解决能力的发展（Sullivan & Bers, 2016）。

综上可知机器人课程已经成为了探索运算思维培养的重要途径，且受到研究者的广泛关注。现有研究主要关注基于运算思维培养的机器人课程的应用和评价，如 Sullivan（2016）等将机器人课堂引入学前幼儿课堂；Chen 等研究者（2017）开发了一套评估五年级学生的运算思维测量工具；Chalmers（2018）探究了机器人教学活动对教师专业发展的影响等。然而对于机器人课程的具体流程设计和实施开展，以支持有效教与学的发生的相关研究还较少。基于此，本研究以运算思维的三维模型作为理论框架，设计了一系列基于项目学习的机器人课程，以发展学生的运算思维，并对学生的运算思维发展情况进行评估。拟主要解决的研究问题如下：

（1）基于三维模型的机器人课程设计是否能显著提高学生的运算思维？

（2）基于三维模型的机器人课程中，学生运算思维的培养受哪些因素的影响？

3. 研究方法

3.1. 基于运算思维三维模型的机器人课程

在基于运算思维三维模型的机器人课程正式开始之前，学生参加 6 个课时的基础课程。在这些课程中学生分别学习了机器人的组装方法、机器人的接口和拓展模块的连接方式、图形化编程软件（由机器人厂商开发的与 Scratch 类似的拖拽模块编程的软件）简单的使用方法以及如何将机器人连接到电脑并将编写好的控制程序下载到机器人进行控制。

在这部分的课程中，没有对学生进行编程方面的训练，仅引导学生对编程软件进行简单的熟悉。

基于运算思维三维模型设计的机器人课程包含了四个设计好的项目和一个学生自主设计完成的项目。四个设计好的项目为：控制机器人彩灯、声控机器车、可调光智能灯、自动车道闸。对这四个项目进行设计时，首先对项目最终完成的作品所包含的运算概念，即学生在通过项目学习编程控制时必须掌握的编程知识点进行标记，并将四个项目分别拆分成了几个不同的小任务。各个项目中不同的任务包含了运算实践中不同的实践内容。

四个项目中并非每个项目都会包含所有运算概念的学习和运算实践的训练内容，而是循序渐进添加包含更多的运算概念、运算实践的项目任务。如控制机器人彩灯项目中只包含顺序、循环、数据的运算概念，第二个声控机器人项目中包含的前三个任务增加了事件、

条件、运算符的概念，后两个任务包含了全部的运算概念。可调光智能灯、自动车道闸两个项目中不同的任务，包含了 7 个运算概念中对应所需运用到的运算概念。

表 4 教学任务与计算思维模型中项目的对应关系

项目	教学任务	对应计算思维模型的项目*
机器人彩灯	模拟红绿灯亮起信号	C1, C2, C7, Pra1, Pra2, Per1
	依次亮跑马灯	C1, C2, C7, Pra1, Pra2, Per2
	根据输入信号亮跑马灯	C1, C2, C5, C7, Pra1, Pra2, Per3
智能灯	传感器检测声音响度	C1, C2, C7, Pra2
	通过声音控制指示灯颜色和亮度	C1, C2, C4, C5, C6, C7, Pra1, Pra2, Pra3
声控机器车	通过拍手启动小车，让小车前进	C1, C2, C4, C5, C6, C7, Pra1, Pra2, Pra3
	没拍手时小车直行，闪烁绿灯，听到拍手声后转弯，同时闪烁红灯 2 次	C1, C2, C3, C4, C5, C6, C7, Pra1, Pra2, Pra3, Pra4
自动车道闸	分析小道闸的工作原理（抽象建模），车被超声波感应到，LED 亮绿灯，道闸打开，车进入，等待两秒道闸关闭，数码管数字（车位）减一，减为 0，LED 红灯，道闸不开	ALL

*计算概念的七个项目为 C1-C7, 计算实践的四个项目为 Pra1-Pra4,计算观念的三个项目记为 Per1-Per3

对每个项目中不同的任务进行设置时，将学生所需完成的项目根据运算思维中四种运算实践进行合理的设置。如学生在声控机器车项目中，共需完成五个任务。第一个任务是读取声音传感器数值，需进行测试和调试的实践；第二个任务中学生通过设定合理数值，使声音控制机器车亮灯启动，在编程时需进行测试和调试的实践、再利用和在创作的实践；随后三个任务为通过拍手启动小车前进、通过拍手控制小车前进和停止、通过拍手控制小车同是亮起不同的指示灯，则需学生进行运算实践中的全部实践内容。

对学生运算观念的培养除了在学生进行项目式学习完成不同任务的过程中被动培养，在课堂中也专门设置了对应教学活动对学生的运算观念进行培养。如在四个项目的开始阶段，向学生介绍项目作品的应用情境，引导学生讨论和回答如何使用运算进行创作，培养学生表达的观念。在分组的活动中，学生互相协作完成任务，培养学生联系的观念。在学生完成作品时，设置讨论和展示活动，培养学生质疑的观念。

学生自主设计完成项目的部分，教师引导学生进行以下环节的教学活动：学生分组提出想通过机器人解决的问题，学生在教师指导下进行讨论将问题进行分割，设计解决问题的流程图，创作作品原型，对作品进行修改，最终分组展示作品（如图 1）。

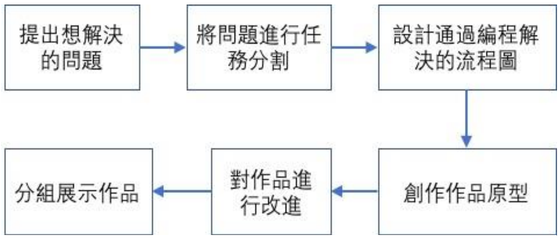


图 1 学生自主设计完成项目教学流程

3.2. 参与学生

参与研究的学生为上海市一所中学的学生，共 17 名学生参与了我们设计的机器人课程。因课程为学校内的学生按兴趣选课，所以在性别和年级方面没有做限制。参与的学生中有男生 16 名，女生一名，六年级 8 名，七年级 9 名。

全部的学生都参与到了机器人课程的学习中，受到场地、计算机数量和机器人数量的限制，他们被分成七个组，每组 2-3 人进行学习。每组学生共同使用一台电脑和一套机器人套件进行学习。

基于项目的课程开始前，所有学生参加了 6 个课时的基础课程，学习机器人组装工具的使用和机器人组装的基本方法，以及图形化编程如何烧录到机器人中。在三个课时的基础课程之后，对学生进行了前测，回收了共 17 份前测数据。

在所有学生完成基于项目的机器人课程后，对学生进行了后测。由于有两名学生没有认真完成后测题目，回收的有效后测数据为 15 份。

3.3. 用于教学的机器人及控制机器人的软件

本研究设计的机器人课程使用童心制物（Makeblock）公司的一款小车机器人 mbot，并配备了一些拓展的硬件模块如 RGB LED 灯模块。小车机器人 mbot 包含一块带有 USB 接口和四个拓展接口的控制板 mCore、两个直流电机、一个超声波传感器、一个巡线传感器、车轮、车框架等零件及螺丝、连接线若干（如图 2）。拓展硬件模块可用连接线连接到机器人主控板上。

图 2 小车机器人 mbot 结构图

控制机器人及拓展硬件的图形化编程软件“慧编程（mblock）”由童心制物公司免费提供，是一款基于 Scratch 开发的，使用方法与 Scratch 一样的编程软件（如图 3）。

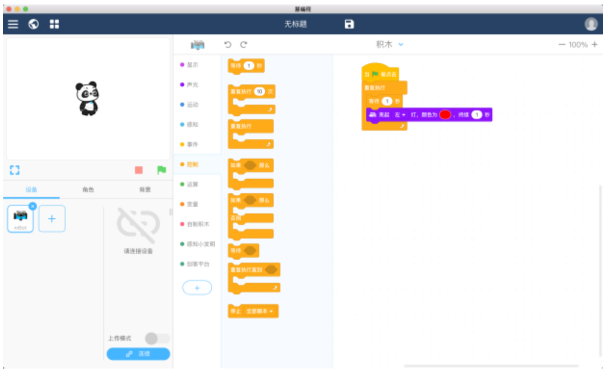


图 3 慧编程软件界面

3.4. *评估工具与评估方法*

对国内外已有的文献进行了综述的基础上发现，在计算思维的定义和如何测量它的问题上，学者们并没有达成共识，这也为本实验研究增加了一定的难度。通过对已有研究中量规量表的筛选和对比，本项目最终决定采用“Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test”一文中作者采用的计算思维测试题（CTt）作为测量工具（Román et al., 2016）。该文章及其 CTt 测试题为英文，因此本研究对其进行了翻译后修改使用。

该测量工具内部一致性指标的可靠性为 α=0.793≈0.8，这可以被认为是具有良好的可靠性。开发该工具的研究者认为该工具主要是针对 12 岁至 14 岁的西班牙学生（7 年级和 8 年级），不过该测试同样也可以用于较低的年级（5 和 6 年级）和较高年级（9 和 10 年级）。这与本研究的参与学生（6 年级与 7 年级）基本吻合。

该工具共 28 道单项选择题，每道题有四个选项，预计学生完成时间为 45 分钟。

使用该工具时，前测顺序按照 CTt 原有顺序进行施测，后测时为防止学生记忆题目影响测试效果，在题目内容不变更的情况下对题目的顺序以及选项的顺序进行了调整。在计分时，按照一题 1 分进行计算。

在课堂教学中，教师对学生的学习情况进行记录，分别从学生听课情况、发言情况、合作学习情况和课堂任务完成情况进行记录评分。并根据学生课程表现将学生分为高表现组和低表现组。对应回收的评估数据，共有高表现组学生 8 名，低表现组学生 7 名，共 15 名。

表 5 学生学习情况记录

项目	A 级	B 级	C 级
听课情况	上课认真听讲，作业认真，参与讨论态度认真	上课能认真听讲，作业依时完成，有参与讨论	上课无心听讲，经常欠交作业，极少参与讨论
发言情况	积极争取发言，积极参与课堂讨论与交流	能尝试发言，有参与讨论与交流	很少发言，极少参与讨论与交流
合作学习情况	善于互相合作，小组内明确分工或者共同协作	能与互相合作，能接受别人的意见	不互相合作甚至出现矛盾
课堂任务情况完成度	完全完成教师布置的任务并在拓展任务中做出自己的特色	能完成教师布置的任务	不能按时完成教师布置的任务

4. 結果與討論

4.1. *基于三维模型的机器人课程对学生运算思维的提升*

课程结束后，使用 SPSS 软件对所有学生运算思维测试（CTt）结果进行配对 T 检验，并计算效应量，结果如表 6。

表 6 全体学生前后测配对 T 检验结果

CTt	t	p	Effect Size（d）
前后测	-2.671	0.018*	0.736

*.表示在 0.05 级别（双尾），相关性显著

从表 6 中数据可知，全体学生在学习基于三维模型的机器人课程后，运算思维测试的前后测表现提高具有显著性（p=0.041），且效应量比较高（d≈0.74）。

根据授课教师的记录，在控制机器人彩灯和声控机器车的项目中，学生对使用图形化编程控制机器人还比较不熟练，在最初需参考教师展示的示范代码才能完成项目。项目的代码写法也与教师提供的示范方法基本一致。在可调光智能灯、自动车道闸两个项目中，学生们对使用图形化编程控制机器人逐渐熟悉，在教师完成任务讲解后，即能通过自行编程完成基本的任务，在完成拓展任务时，才需教师对编程进行进一步的指导。且在实现同一功能时，学生能够通过不同于教师提供的编程思路实现。在最后学生自行设计项目并完成的环节，学生在不同的分组中也设计出了不同于教师教学中示例的作品（如图 4）。可以认为，在完成基于三维模型的机器人课程后，学生总体的运算思维得到比较好的提升。

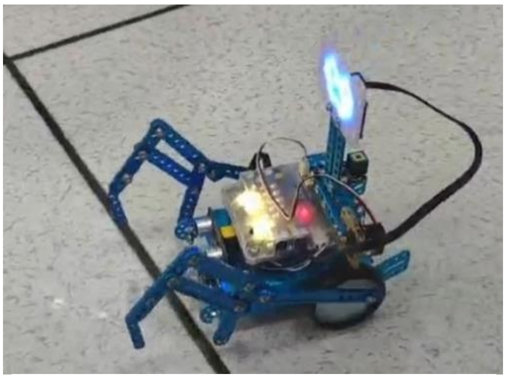


图 4 学生设计的不同于教师示例的机器人

4.2. 课程中的不同表现组运算思维发展的差异

为探究课程中哪些因素对学生的计算思维发展产生了影响，首先将学生的课堂表现进行高低表现组划分。在划分表现组后，发现两组样本的样本数较少，故对分为高低表现组的学生运算思维测试（CTt）前后测结果分别使用 SPSS 软件进行 Wilcoxon 检验，描述性统计见表 7，Wilcoxon 检验结果见表 8。

从表 7 中数据可知，高、低表现组学生在学习基于三维模型的机器人课程后，运算思维测试的前后测表现存在明显的差异。

从表 8 高、低表现组学生前后测 Wilcoxon 检验结果中可知两组学生运算思维测试前后测结果差异显著性的不同。

表 7 高、低表现组前后测结果描述性统计

组别	学生数（n）	前测>后测	前测=后测	前测<后测
高表现组	8	1	1	6
低表现组	7	3	1	3

表 8 高、低表现组学生前后测 Wilcoxon 检验结果

CTt	高表现组		低表现组	
	Z	p	Z	p
前后测	-2.120[a]	0.034*	-0.106[b]	0.916

*.表示在 0.05 级别（双尾），相关性显著

a.基于负秩

b.基于正秩

高表现组学生前后测表现提高具有显著性（p=0.034）。

而低表现组学生，从分析结果来看，认为他们的前后测表现不具有显著性（p=0.916）。

高、低表现组学生在学习基于三维模型的机器人课程后，运算思维发展差异相当明显，从一定程度上说，学生若认真参与课程的学习并积极完成任务和项目，是有助于其发展运算思维的。而高表现组学生运算思维的显著提升，亦说明课程对其有较大的帮助。为分析低表现组学生为何运算思维提升不显著，学习效果明显较高表现组学生差，再次与授课教师交流课程实施情况，并查阅其课堂记录，以分析高、低表现组学生的差异是由课程设计本身造成，还是由其他原因造成。

4.3. 不同表现组运算思维发展差异的原因

通过与授课教师交流并查阅其课堂记录，发现低表现组学生在课堂学习中存在以下情况：

1.有两个学习小组中，既存在高表现组的学生，亦存在低表现组学生。在完成项目时，在同一学习小组中的低表现组学生依赖高表现组学生进行编程，参与编程训练少于高表现组学生。

2.有一个学习小组中三名学生均为低表现组学生，在完成项目时常出现争论，出现多次争抢机器人进行操控及争抢电脑进行编程，浪费了许多学习时间，使项目的完成度与编程训练均表现较差。

而高表现组学生，在与低表现组学生同组时，为完成教师布置的任务，往往进行更长时间的编程。高表现组学生同处一组时，能进行合理分工，共同完成项目和编程的训练。

由此可以认为，基于运算思维三维模型的机器人课程，在项目式学习的内容和活动设计上，对学生提升运算思维是比较有效的。但在课程实施时，由于器材紧缺和学生学习过程引导存在缺失，使低表现组学生未能很好地参与到课程当中，使其运算思维提升不明显。

5. 总结

本研究通过基于运算思维三维模型设计机器人课程，并在初中进行实施，通过使用运算思维测试（CTt）进行前后测检验得到该课程设计能有效学生运算思维的结论。对今后设计以培养学生运算思维的课程，尤其是机器人课程、STEM 课程，具有一定的参考作用。怎样基于运算思维三维模型去进一步对现有的课程进行针对性的优化，将通过更多得到实践和研究进行探索。

而研究中发现课程高、低表现组学生在运算思维发展存在明显的差异，亦通过分析发现原因。针对这个发现提出以下几点建议：1.在发展学生运算思维的课程，尤其是项目式学习的课程当中，应配备充足的电脑和器材，确保学生能够充分进行学习和实践活动；2.在相关课程实施时，教师应关注到分组学习的学生协作学习的情况，并进行及时的干预，以防学生学习课程出现马太效应。

本研究亦存在几个缺陷。一是研究样本较少，虽然总体学生提升效果的统计分析结果由 15 名学生样本进行检验，于统计学上具有可靠性，但学生不同表现组中样本较少（分别为 8 个样本和 7 个样本），数据分析结

果不太稳定。二是未能对学生更长周期的学习进行跟踪研究，对学生运算思维培养非一蹴而就，仍需更多的实践和研究。

6. 参考文献

王旭卿（2014）。面向三维目标的国外中小学计算思维培养与评价研究。**电化教育研究**，**7**， 48-53。

王娟、胡来林和安丽达（2017）。 国外整合 stem 的教育机器人课程案例研究——以卡耐基梅隆大学机器人学院 robotic 课程为例。 **现代教育技术**，**04**， 34-39。

吴秀凤和陈奕贤（2015）。 Stem 理念下中小学 arduino 机器人教学模式研究。 **福建电脑**，**5**，138-139。

余胜泉和胡翔（2015）。 Stem 教育理念与跨学科整合模式。 **开放教育研究**，**4** ，13-22。

杜娟和臧晶晶（2014）。Stem 教育视野下小学低年级智能机器人教学模式研究。**中小学信息技术教育**，**5** ，52-54。

秦换鱼（2016）。Stem 理念下的简易机器人教学研究。 **中国信息技术教育**，**9** ，66-68。

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

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum. *Computers & Education*, *72*(1), 145-157.

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

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)*. Retrieved January 22, 2019, from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chalmers,C.. (2018). Robotics and Computational Thinking in Primary School. *International Journal of Child-Computer Interaction*, *17*, 93-100.

Chambers, J., Carbonaro, M., & Rex, M. (2007). Scaffolding Knowledge Construction through Robotic Technology: A Middle School Case Study. *Electronic Journal for the Integration of Technology in Education*, *6*, 55-70.

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming. *Computers & Education*, *109*, 162-175.

Danli, W., Tingting, W., & Zhen, L. (2014). A Tangible Programming Tool for Children to Cultivate Computational Thinking. *The Scientific World Journal*, *2014*, 1-10.

Executive Office of the President, & President's Council of Advisors on Science and Technology (2010). *Prepare and Inspire: K-12 Education in Science, Technology, Engineering and Math (STEM) for America's Future.* Retrieved January 22, 2019, from http://files.eric.ed.gov/fulltext/ED516009.pdf

Fagin, B., & Merkle, L. (2003). Measuring the Effectiveness of Robots in Teaching Computer Science. *Proceedings of the 34th Technical Symposium on Computer Science Education (SIGCSE)*. ACM, 307-311.

Flot J, Higashi R, McKenna J, Shoop R., et.al. (2016). *Using Model Eliciting Activities to Engage Students in Computational Thinking Practices*. Retrieved January 22, 2019, from https://www.researchgate.net/publication/305700803_Using_Model_Eliciting_Activities_t_Engage_Students_in_Computational_Thinking_Practices

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning. *Proceedings of 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE, 59-66.

Linder, S. P., Nestrick, B. E., Mulders, S., & Lavelle, C. L. (2001). Facilitating Active Learning with Inexpensive Mobile Robots. *Journal of Computing Sciences in Colleges*, *16*(4), 21-33.

Norton, S., McRobbie, C., & Ginns, I. (2007). Problem Solving in a Middle School Robotics Design Classroom. *Research in Science Education*, *37*(3), 261-277.

Seiter, L., & Foreman, B. (2013). Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*. ACM, 59-66.

Sullivan, A., & Bers, M. U. (2016). Robotics in the Early Childhood Classroom: Learning Outcomes from an 8-week Robotics Curriculum in Pre-kindergarten through Second Grade. *International Journal of Technology & Design Education*, *26*(1), 3-20.

The White House (2013). *Federal Science, Technology, Engineering and Mathematics (STEM) education: 5-year Stragetic Plan*. Retrieved January 22, 2019, from http://www.whitehouse.gov/sites/default/files/microsites/ostp/stem_stratplan_2013.pdf

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

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

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2016). Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678-691.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking in STEM Task Design: Authentic, Useful, Experiential, and Visual

Hao-min TIEN [1], Jung-chuan YEN [2*]
[1] Graduate School of Mathematics and Information Education in National Taipei University of Education, Taiwan
[2] Department of Mathematics and Information Education in National Taipei University of Education, Taiwan
wingtien23@hmes.tp.edu.tw, jcyen.ntue@gmail.com

ABSTRACT
The instructional design of this study is based on the learner center approach. We advocate the development of learning activities by the 6E model of inquiry learning, including engage, explore, explain, elaborate, enrich, evaluate. The framework of learning tasks were project-based design by using IoT sensor technology and visual programming language. Through integration of health and physical education, mathematics, science and technology, computers and integrative activities, we have designed a cross-domain inquiry context, the data analysis and statistical charting of the weather factors in the campus environment, to promote the collaborative learning and computational thinking of primary school students. This study specifies the connotation of the complete instructional design and learning tasks of computational thinking, and summarize the key points of design principle are authentic, useful, experiential, and visual.

KEYWORDS
STEM activity, internet of things, computational thinking, instructional design

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

跨領域運算思維學習任務設計：真實、有用、體驗、視覺化

田昊民 [1]，顏榮泉 [2*]
[1] 國立臺北教育大學數學暨資訊教育研究所，臺灣
[2] 國立臺北教育大學數學暨資訊教育學系，臺灣
wingtien23@hmes.tp.edu.tw，jcyen.ntue@gmail.com

摘要

本研究以 STEM 之參與、探索、解釋、精熟、深化、評量等 6E 教學模式為學習活動發展架構，以運用物聯網感測科技與視覺化程式設計之專題活動為學習任務，整合國小健康、數學、生活科技、電腦與綜合活動等課程，設計校園環境空汙偵測之統計圖表與數據分析的跨領域運算思維學習活動。本研究具體說明教學設計架構與學習任務之內涵，並提出真實（authentic）、有用（useful）、體驗（experiential）、與視覺化（visual）等四項運算思維跨領域活動設計的原則，期能提供相關教學活動設計之參考。

關鍵字

STEM 教學活動；物聯網；運算思維；教學設計

1. 前言

由人工智慧、機器人與虛擬科技所建構的人類未來生活樣貌，正不斷地挑戰現有的工作世界與知識體系。近年來，隨著愈來愈多國家開始在基礎教育階段，推動以運算思維（Computational Thinking）為內涵的全球素養（Global Competence）教育後，這股風潮已逐漸形成世界各國群起競逐的教育政策，也使得培養公民必備適應未來科技生活所需知識與關鍵能力的跨領域 STEM 教學，成為素養教育領域中相當重要的教學實務研究議題。

STEM 教學是指教學活動設計中能融合數學、科學、科技與工程等跨學科的學習內涵。傳統的數理學科教學模式常將知識與其應用情境分離，著重於傳授知識的記憶、定理與計算，缺乏提供學習者應用這些知識以解決日常生活中所面臨問題的脈絡，導致學習成效不彰。而針對此種惰性知識（inert knowledge）的問題，許多學者指出透過強調能與真實世界經驗連結的 STEM 課程，應能藉由解決實際生活中所面臨的問題，培養學習者獨立思考與終身學習的能力，並能促進其深層學習與知識應用的機會（Blackwell & Henkin, 1989；Fortus etc., 2005）。

台灣即將於 2019 年推動的十二年國民基本教育課程綱要，特別將科技領域獨立成為新的必修學科領域。然臺北市政府為了因應此課綱在國小階段並未規劃相關資訊素養教育的學習時數，特別著手訂定台北市國小資訊科技課程教學綱要，將資訊教育、生活科技與科技相關議題整合成為**資訊科學與科技應用**、**運算與設計思維**、**資訊科技與人類社會**等三面向的課程實施方向。本研究即據此規畫針對國小六年級學生之STEM跨領域學習課程，以具備物聯網程式設計教學功能之 webduino 套件作為課程實踐之教材模組，整合健康、數學、資訊等學科教學，設計以溫溼度及空氣汙染等環境偵測為主題的雲端資料記錄與應用，再透過數學課進行數據分析與統計圖表的繪製，將學科知識落實於生活情境中的環境保護議題討論，藉以培養學生具備以運算思維解決真實生活問題的素養能力。

2. 文獻探討

2.1. 以體驗式學習為策略的 STEM 運算思維教學

ISTE 將運算思維的學習目標視為幫助學習者發展出一套運用計算機解決問題的心智模式，使其能有效地使用資訊科學工具解決複雜的問題（ISTE, 2011）；STEM 的 6E 科學探究活動則是強調學習者自發性的發掘與尋求科學問題解答的歷程，來培養學習者的探究能力（Zimmerman, 2007），兩者在學習目標的本質上均強調從具體到抽象的高層次思考與循序漸進的問題解決步驟。因此，本研究以 Kolb 體驗式學習理論中的資訊獲取（information perceiving）與資訊處理（information processing）兩觀點，嘗試融合運算思維與 6E 探究學習之教學模式（Kolb, 2014），詳如圖 1 所示。

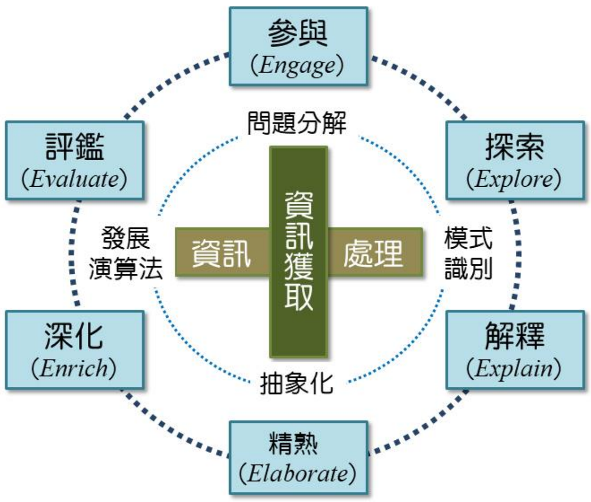


圖 1 以運算思維融入 6E 探究學習之教學模式

此教學模式實際上包含內圈的課堂引導教學及外圈的網路探究學習活動兩部分，是一種混合式的數位學習模式（blended learning）。教師在課堂上，以符合經驗學習理念的資訊獲取原則（具體→抽象）來設計教學範例，也就是必須從學習者熟悉的具體經驗出發，逐步累積經驗後發展至抽象的上層概念。其次，教師必須透過範例的解說進行資訊處理程序的思考引導，讓

學習者從識別問題、發展解題計畫、採取行動到撰寫程式碼與除錯，協助學習者建立實際解決問題步驟的心智模式（對小學生來說其實就是一種解決問題的思考習慣）。教師的引導活動完成後，學習活動的軸心則轉移至後續課堂與課後的網路探究學習任務。

2.2. 運用物聯網科技串連跨領域的探究學習

融合STEM與運算思維的教學設計，應是未來中小學教學現場相當重要的跨領域教學模式。Barry（2014）基於探究學習與建構主義之學理，提出 STEM 的 6E 教學設計模式，建議教師以參與（engage）、探索（explore）、解釋（explain）、精熟（elaborate）、深化（enrich）、評量（evaluate）等架構，進行能促進學習者跨域整合與探究學習的活動設計。然而，除了教學設計的理論依據外，教學實務上需要設計能提供孩子們具體操作與反思觀察的學習任務，以及既能支撐此學習任務又具備運算思維教學內容的教材與工具（Mahajan, Wu, Tsai, & Chen, 2018；Tseng, Tissenbaum, Kuan, Hsu, & Wong, 2018）。

因此，本研究以學習者中心為理念，參考類似主題之教學相關研究（張立農、江孟玲、林昭遠，2015；王翊芬，2018；張嘉玲、吳翎華、王秀文，2018），主張以STEM探究學習之參與、探索、解釋、精熟、深化、評量為學習活動發展架構，決定以物聯網科技的超音波、PIR、溫溼度、PM2.5 等環境感測探究專題活動為學習任務內容，串聯整合國小健康、數學、生活科技、電腦與綜合活動等課程，設計一個能促進國小學生透過長時間的雲端資料庫紀錄與分享、共同合作完成校園環境氣候因素之數據分析與統計圖表比較的運算思維跨領域探究學習課程。

本研究具體的學習內容包含兩階段：第一階段為運算思維與物聯網科技之學習，主要在建構學生使用具備Wi-Fi 連網的控制板（Webduino Smart）與環境感測元件的能力，透過積木式的程式語言學習輸入、控制、輸出的科技系統概念，並在學習過程中融入問題拆解、模式識別、抽象化及演算法則的邏輯思考與問題解決能力；第二階段為校園生活環境氣候因素的跨領域探究學習，此階段將結合數位學習資源提供相關教學範例之實作指引，以專題導向的小組合作學習為方法，鼓勵學生採用物聯網科技之控制板與感測元件，針對教師引導討論的校園環境空汙議題提出小組想要深入探討的主題，並且規劃如何蒐集與分析校園環境如溫、溼度與空氣品質等相關數據，進行以 6E 學習模式為架構之假設與驗證學習活動。

3. 教學設計

3.1. 教學對象

本研究之教學對象為臺北市某國小六年級之學生。研究對象自三年級開始學習資訊課程，熟悉電腦基本功能操作，了解基本電腦知能。並曾於資訊課程中學習視覺化程式語言（Scratch、Hour of code），已具基礎的程式邏輯概念。

3.2. STEM 活動架構

本活動設計以STEM教學為核心，進行橫跨四個學科領域的內容知識，分別是科學（Science）中的自然環境探索與保護、科技（Technology）中的物聯網開發板程式設計、工程（Engineering）中的開發板電路接線及數學（Mathematics）中統計圖表的繪製與判讀。

研究者為該實驗班級資訊課及健康課任課老師，規劃於健康課時延伸課本教學內容，進行校園空氣品質旗辨讀認識與空氣品質數據的辨讀教學；並於資訊課時結合物聯網設備，實作 Webduino 空氣 PM2.5 即時偵測站。再將測站所回報之每小時空氣品質數據提供給數學科目授課的導師，進行折線圖的繪製與數據判讀教學，完成本次STEM跨領域課程。以下為STEM活動架構與 6E 探究學習教學模式之課程規劃：

表 1 本研究之 STEM 教學活動設計

週次	主題	實施領域	教學內容
1	認識空氣汙染	健康	認識空氣污染、校園空氣品質旗幟介紹並預測校園內空氣品質較差的區域。
2	認識物聯網	電腦	探索影音平台上有趣的物聯網應用影片，預測其工作原理並與同學分享。
3	電路接線模擬器紅綠燈系統	電腦	運用模擬器操作電路接線，並完成各種顏色燈號秒數設定的紅綠燈系統。
4	輸入、控制與輸出-LED 燈光秀	電腦	加深程式執行概念的複雜度，透過模擬器設計程式使 LED 燈亮、滅及閃爍，了解輸入、控制與輸出的運算思維問題解決概念。
5	迴圈與變數	數學	介紹迴圈與變數的抽象概念，透過模擬器實作能執行輸入初始值、累加變量及執行次數的迴圈，使其能依條件計算總和。
6	IF 判斷式-多段開關燈	電腦	運用模擬器，整合變數、迴圈、IF 判斷式，並使用提供的網頁互動遙控器，製作出多段開關燈功能。
7	智慧燈泡 DIY	生活科技	開發板與感測器的專題實作，應用光敏電阻和燈泡，實作能依照光敏電阻偵測到的數據，以程式控制自動亮滅的智慧燈泡。
8	溫溼度感測與記錄	電腦	開發板與感測器的專題實作，串接溫溼度感測器，並將偵測到的溫度、溼度數據顯示在網頁中。
9	哈啾！我的 PM2.5 感測裝置	綜合活動	開發板與感測器的專題實作，運用空氣懸浮物質感感器（PM2.5），將環境

			偵測到的數據顯示在網頁中。
10	校園空污熱點大搜查	數學	將完成的 PM2.5 感測裝置放在各組預測的校園空氣品質較差區域，完整蒐集 24 小時的數據；由數學老師帶領進行統計折線圖的繪製與數據判讀教學，各組發表預測與驗證結果。

表 2 本研究教學活動具備之 STEM 領域與 6E 學習內涵

主題	S	T	E	M	參與	探索	解釋	精熟	深化	評量
認識空氣汙染	V				V		V			
認識物聯網		V	V		V	V	V			
電路接線模擬器		V	V		V		V	V		
輸入、控制與輸出		V	V		V		V	V		
迴圈與變數		V	V	V	V		V	V	V	
IF 判斷式多段開關燈		V	V	V	V		V	V	V	
智慧燈泡 DIY		V	V		V			V	V	V
溫溼度感測與紀錄	V	V	V	V	V			V	V	V
我的 PM2.5 感測裝置	V	V	V	V	V			V	V	V
校園空污熱點大搜查	V	V	V	V	V	V	V	V	V	V

3.3. *教學環境與工具*

本研究以 STEM 跨領域學科教學為主，藉由具備 WiFi 模組的整合式物聯網開發環境，實施專題式的問題導向學習。本計畫採用由臺灣本土科技廠商所研發的 Webduino 學習平台與教材包進行教學。從教學功能而言，此教材環境具備以下幾種功能與教學內涵：

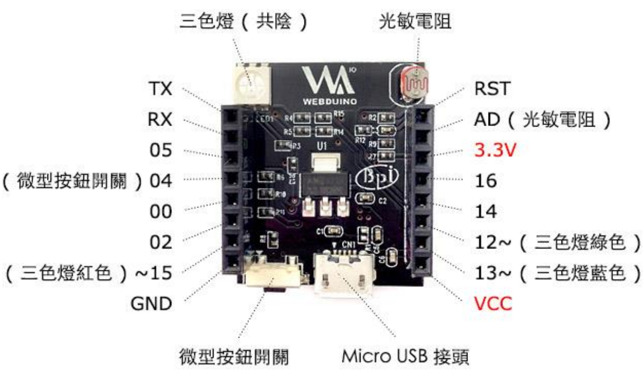


圖 2 相容 Arduino 且具 Wi-Fi 功能的 Smart 開發板

3.3.1. *相容 Arduino 的 Smart 開發板*

Webduino 的開發板 Smart 是基於 Arduino Pro Mini 的電路模組與功能架構，再附加 WiFi 連網晶片所重新設計而成（如圖 2），因此幾乎相容於 Arduino 的所有周邊感測器或零件。Arduino 為目前全世界相當普及的創客教學與多媒體互動裝置開發板，相容 Arduino 的教學意涵在於能夠獲得數量相當多的軟硬體教學資源，以及社群貢獻的開源程式碼範例。

3.3.2. *視覺化積木程式設計環境*

Webduino 基於 Google Blockly 語言架構，開發可與 JavaScript 語言互換的視覺化積木程式設計環境，只要打開 chrome 瀏覽器，就能以網頁互動與滑鼠拖拉的方式學習撰寫程式，能降低初學程式設計的焦慮。

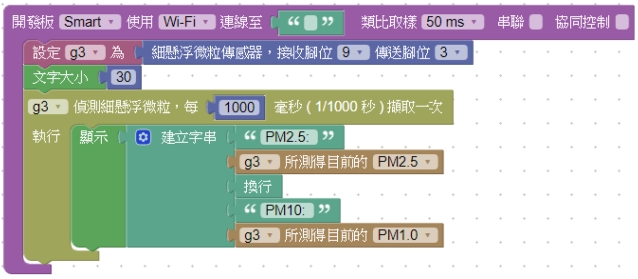


圖 3 視覺化的積木程式設計環境能降低學習門檻

3.3.3. *提供電路接線測試練習的模擬器*

電子元件或感測模組都具有正負極性，當線路接錯時容易造成元件短路而損毀。因此提供模擬器能讓初學者線進行電路接線的練習，並模擬接上電源後是否會能正常運作，等確認在模擬器上的執行都正確後，材進行實際的電路實習。這能大幅降低實驗材料的損耗，亦能培養學習者程式除錯的能力。

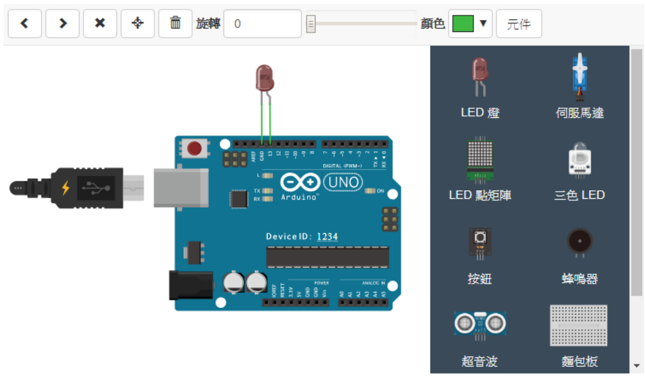


圖 4 提供電路接線與測試除錯的模擬器

3.3.4. *提供多種感測元件的教學範例*

本研究教學活動設計，選擇了 LED 燈、光敏電阻、超音波、溫溼度、PM2.5 等學生在日常生活中就常接觸到的感測器，選擇的開發板若具備豐富的教學步驟說明與影音範例網站，能降低教師教學準備的負擔。

3.4. 資料蒐集與反思討論

本研究在教學過程中，每節課均提供學習單，透過引導問題促進學童針對專題要解決的問題思考。課程實施過程，另邀請跨領域協同教學的夥伴進行觀察記錄與拍照；部分戶外課程活動，另安排錄影記錄或課後訪談。教學活動結束後，研究者邀請參與跨領域授課之教學夥伴，以教師專業社群方式進行反思討論教學活動過程所蒐集之各項師生互動與學習歷程資料。

4. 結果與討論

本研究透過教師社群之教學反思討論會議，歸納跨領域運算思維學習任務的設計要素包含真實（authentic）、有用（useful）、體驗（experiential）、與視覺化（visual）。

4.1. Authentic: 生活中的真實學習任務

隨著都市化的發展，都會地區因汽機車數量的持續增加使得廢氣排放問題難以解決，而生活品質的需求提高也使得燃煤發電的比例逐漸增高，連帶造成環境負荷與空氣污染的問題日益嚴重，這是目前台灣社會大眾所面臨的重要議題。透過教師的觀察紀錄及學習單的回饋，學習者對於本研究學習任務的設計，以每天生活都需面對的校園周邊環境空污的偵測為主軸，均展現比原本課程教學內容更高的興趣，學習過程中亦表達願意配合改變個人習慣，確實落實污染防治的基本功。這與 Sawatzki（2017）以真實生活情境問題為學習任務，能促進學習者高度投入動機的主張相符。

圖 5 研究者於健康課程中介紹空氣污染與防治之概念

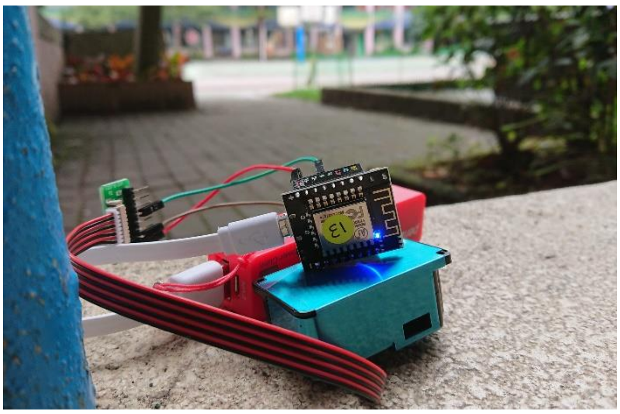


圖 6 學習者將感測裝置放在校園角落進行 PM2.5 偵測

4.2. Useful: 有用的問題解決經驗

以設計真實世界經驗連結的 STEM 課程為出發點，研究者擷取國小不同領域的課程學習經驗、協同各領域的老師進行跨領域教學，藉由讓學生面對實際生活中所面臨的問題，以輸入-控制-輸出之系統概念，引導學生跨學科學習運用各個課程所學的知識與技能，有效促進學習者把所學轉化成有用的問題解決經驗。例如：學生學會運用資訊課所教的物聯網感測裝置，在健康課實際紀錄校園溫濕度與 PM2.5 的空污指數，進而深刻思考校園的汙染排放問題（圖 6）。此外，運用試算表軟體可以將物聯網感測裝置所記錄的空污數據，在數學課中繪製統計圖表並分享解釋個人對圖表的判讀與心得，這都是培養學童具備使用資訊科技的工具或思考方式，解決未知問題的終身學習能力。

4.3. Experiential: 可體驗的科學探究過程

台灣受升學主義的影響，許多基礎教育中能提供具體操作經驗的生活科技或動手實作課程均不受重視。本研究規劃的物聯網裝置，先透過模擬器的電子電路接線模擬，能在安全無虞的模擬操作中學習工程與科技概念，當反覆操作確認熟悉後再提供實際接線的實習。從教師的課堂觀察與紀錄中發現，學習者對模擬器的操作相當專注投入，反思發問的次數與平時課堂的沉悶相比有迥然不同的結果，這種因教學策略與工具的選擇改變，造成學習者學習態度上的轉變，令參與教師們感到相當雀躍。

圖 7 學生透過模擬器相當專注於電路接線與程式設計

圖 8 視覺化積木介面能讓學習者直覺式的操作與溝通

4.4. *Visual: 視覺化的程式學習體驗*

視覺化程式設計語言（visual programming language）與傳統文字語法式的程式設計語言不同，它是一種能讓操作者使用圖形化元素進行程式設計的直覺式語言（Haeberli, 1988）。視覺化程式設計語言能免除初學程式設計語法指令的枯燥與挫折，具備增加趣味、迅速獲得回饋、易於操作等優點，非常適合作為基礎教育層級學童學習運算思維的工具（徐宏義、羅曼如，2016）。本計畫使用之 Blockly 積木式語言，最大的特色就是完成程式積木的堆疊後，可將視覺化的積木自由轉換成 Javascript、Python、PHP 或其它常見的文字式語言，能提供程度超前或較佳的學習，作為深入自學與進階 C 語言、Java 語言的橋接工具。本計畫在教學實驗期間，由於學習者之前已經學過 Scratch 程式設計語言，對於導入物聯網電子電路接線與感測裝置，多數學習者表現出能直接上手且主動思考如何解決老師提供學習單上的引導問題。超過 95%的學習者表示喜歡這種視覺化程式語言的直覺介面，並全部都表達有意願持續學習更進階的課程。

5. 結論與建議

本研究以 STEM 探究學習之參與、探索、解釋、精熟、深化、評量為學習活動發展架構，以運用物聯網感測科技與視覺化程式設計語言之專題活動為學習任務，整合國小健康、數學、生活科技、電腦與綜合活動等課程，設計出一套校園環境空污偵測之統計圖表與數據分析的運算思維跨領域探究學習課程。本研究具體說明完整教學設計架構與學習任務之內涵，並透過教師專業社群歸納教學實驗歷程所蒐集之資料，提出真實（authentic）、有用（useful）、體驗（experiential）、與視覺化（visual）等四項運算思維跨領域活動設計的原則，期能提供相關教學活動設計之參考。

6. 致謝

本研究承蒙科技部 MOST 106-2511-S-152-001 與 MOST 107-2511-H-152-009 專題研究計畫之經費補助，謹此致謝。

7. 參考文獻

王翊芬（2018）。**以自發性地理資訊探討民眾對空氣污染的環境識覺**（未出版的碩士論文）。臺北市：國立台灣大學。

張立農、江孟玲和林昭遠（2015）。台灣交通空氣品質監測站 PM10 變異影響因素之研究。**水土保持學報，47（1）**，1235-1246。

張嘉玲、吳翎華、王秀文（2018）。自造者製作空氣盒子之知識探索行為。**工業設計，137**，20-24。

徐宏義、羅曼如（2016）。**軟體打造科技大未來-程式設計是下一代最重要的生存技能**。台北，商周出版社。

Barry, N. B. (2014). The ITEEA 6E Learning by DeSIGNTM Model. *Technology and Engineering Teacher, March*, 14-19.

Blackwell, D., & Henkin, L. (1989). *Mathematics: Report of the Project 2061 Phase I Mathematics Panel*. Washington, D. C.: American Association for the Advancement of Science.

Fortus, D., Krajcikb, J., Dershimerb, R. C., Marx, R. W., & Mamlok-Naamand, R. (2005). Design-based Science and Real-world Problem Solving. *International Journal of Science Education*, *27*(7), 855–879.

Haeberli, P. E. (1988). ConMan: A Visual Programming Language for Interactive Graphics. *ACM SigGraph Computer Graphics, 22*(4), 103-111.

ISTE (2011). *Operational Definition of Computational Thinking for K–12 Education*. Retrieved January 14, 2019, from http://www.iste.org/ docs/ct-documents/compu tational-thinking-operational-definition-flyer.pdf

Kolb, D. A. (2014). *Experiential Learning: Experience as the Source of Learning and Development*. FT press.

Mahajan, S., Wu, W. L., Tsai, T. C., & Chen, L. J. (2018, September). Design and Implementation of IoT-enabled Personal Air Quality Assistant on Instant Messenger. *Proceedings of the 10th International Conference on Management of Digital EcoSystems.* ACM, 165-170.

Sawatzki, C. (2017). Lessons in Financial Literacy Task Design: Authentic, Imaginable, Useful. *Mathematics Education Research Journal, 29*(1), 25-43.

Tseng, C. H., Tissenbaum, M., Kuan, W. H., Hsu, F. C., & Wong, C. C. (2018). A Design-based Approach to Implementing a Computational Thinking Curriculum with App Inventor and the Internet of Things. *Proceedings of the International Conference on Computational Thinking Education 2018 (CTE 2018)*. Hong Kong: The Education University of Hong Kong.

Zimmerman, C. (2007). The Development of Scientific Thinking Skills in Elementary and Middle School. *Developmental Review, 27*(2), 172-223.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Research on STEM Curriculum Design for Computational Thinking:

Framework Design and Case Analysis

Hui SHI[1*], Feng LI[2*]
[1] Department of Education Information Technology, East China Normal University, China
[2] School of Open Learning and Education, East China Normal University, China
51184108034@stu.ecnu.edu.cn, fli@srcc.ecnu.edu.cn

ABSTRACT
The rapid development and popularization of information technology has profoundly changed people's behaviors and thinking characteristics. Computational thinking has become an important content of information technology education in primary and secondary schools. As a kind of learning mode oriented to subject integration and project orientation, STEM education is practice training. Computational thinking provides a mode of operation. On the basis of combing the development of computational thinking and computational thinking education and related research, this paper clarifies the relationship between computational thinking and STEM education, and focuses on the core concept of computational thinking, drawing on the typical STEM curriculum practice model, from teaching themes and teaching. Five aspects of goal, teaching method, teaching resources and teaching evaluation, designing the STEM curriculum framework for computational thinking, and in-depth analysis with specific teaching cases, explaining and explaining in case mode, in order to provide practical reference for the cultivation of students' computational thinking.

KEYWORDS
computational thinking, K-12, STEM education, curriculum framework, instructional case

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019*. Hong Kong: The Education University of Hong Kong.

面向计算思维的 STEM 课程设计研究：框架设计与案例分析

时慧[1*]，李锋[2*]
[1]华东师范大学教育信息技术学系，中国
[2]华东师范大学开放教育学院，中国
51184108034@stu.ecnu.edu.cn，fli@srcc.ecnu.edu.cn

摘要

信息技术的快速发展与普及深刻改变着人们的行为方式与思维特征，计算思维成为中小学信息技术教育的一项重要内容，作为一种面向学科融合与项目导向的学习方式，STEM 教育为培养计算思维提供了一种可供操作的途径。本文在梳理计算思维和计算思维教育发展历程的基础上，通过厘清计算思维与 STEM 教育之间的关系，围绕计算思维的核心理念，借鉴典型的 STEM 课程实践模型，从教学主题、教学目标、教学方式、教学资源、教学评价五个方面，设计面向计算思维的 STEM 课程框架，同时结合具体的教学案例进行深入剖析，以期为学生计算思维的培养提供实践参考。

关键字

计算思维；K-12；STEM 教育；课程框架；教学案例

1. 前言

在信息和通信技术快速发展的时代，世界经济格局发生急剧变化，各国人才竞争愈演愈烈，新时代为年轻一代具备更多现实技能提出更高要求，以应对未来的需求和挑战，其中计算思维被认为是改善年轻一代逻辑推理、提高分析和解决问题能力的 21 世纪成功的关键属性，越来越多的国家将计算思维列入学校课程标准，甚至写入国家发展战略规划中。2017 年，我国在新修订的高中信息技术课程标准中明确将“计算思维”信息技术学科核心素养的一项核心内容，并将计算思维定义为“个体运用计算机科学领域的思想方法，在形成问题解决方案的过程中产生的一系列思维活动，主要表现为形式化、模型化、自动化和系统化四个方面。”（李锋和赵健，2016）2018 年 12 月，特朗普的白宫公布了新的五年战略计划——《制定成功路线：美国的 STEM 教育战略》，该计划重点提出，要使计算思维成为所有教育的必要组成部分。 计算思维越来越多地被视为一种数字化生存的一种普适能力，它需要成为学校教育教学活动的一部分，那么如何将计算思维的培养真正融入实际学校教学中，就成为问题的关键。因此，探索面向计算思维的教学模式，助力学生计算思维能力的发展与突破，成为新时代教育领域关注的热点话题。使每个学习者都有能力评估信息，分解问题，并通过适当使用数据和逻辑制定解决方案。作为一种面向学科融合与项目导向的学习方式，STEM 教育为实践培养计算思维提供了一种可供操作的模式。基于此，本文从计算思维和 STEM 教育的内涵出发，设计面向计算思维的 STEM 课程框架，以期为计算思维的培养提供有价值的借鉴。

2. 计算思维和计算思维教育的发展历程

2.1. 计算思维的概念及内涵

计算思维这个专业术语在 20 世纪 80 年代被广泛使用，美国卡内基·梅隆大学周以真教授提出，计算思维是运用计算机科学的基础概念进行问题解决、系统设计与人类行为理解的过程，代表着一种普遍的认识和一项普适的技能。（Wing, 2006）

在美国，2005 年 6 月美国总统信息技术咨询委员会（The President’s Information Technology Advisory Committee, PITAC）在向美国总统提交的一份名为《计算科学：确保美国竞争力》（Computational Science: Ensuring America’s Competitiveness）的报告中提出了计算思维，并得出结论：（1）“计算”学科具有促进其他学科发展的作用；（2）21 世纪科学上最重要的、经济上最有前途的研究前沿有可能通过熟练地掌握先进的计算技术和运用计算科学而得到解决。（President’s Information Technology Advisory Committee, 2005）在中国，1992 年黄崇福即给出了计算思维的定义：“计算思维就是思维过程或功能的计算模拟方法论，其研究的目的是提供适当的方法，是人们借助现代和将来的计算机，逐步达到人工智能的较高目标。”（黄崇福，1992）

根据周以真教授提出的计算思维的概念，计算思维的内涵体现在以下三个方面：在一个真实的问题情境中，学会用计算机科学的基础概念以及计算机科学家的思维方式设计系统、求解问题，从而理解人类的行为。

2.2. 中小学计算思维教育的发展历程

计算机学科知识与技能教育以及渗透计算思维的教学方式是开展计算思维教育的重要途径。在中小学阶段，与计算思维培养最为密切的则是信息技术课程，在各个国家的信息技术课程或计算机科学课程标准中，倾向于将计算思维教育作为其核心内容。中小学计算思维教育大事记如图 1 所示。

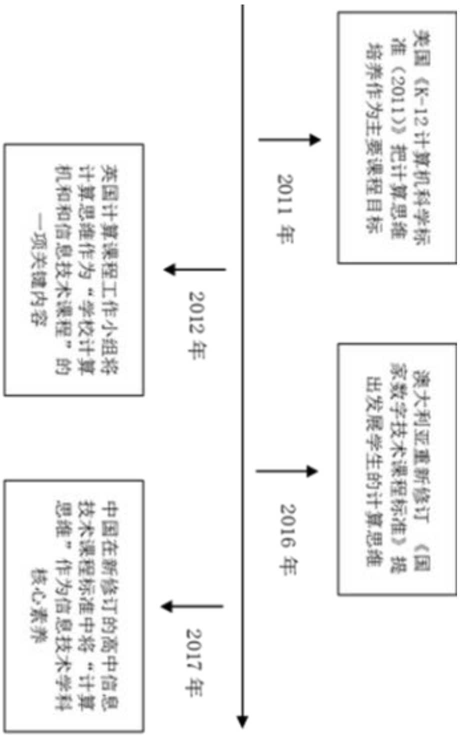


图 1 中小学计算思维教育大事记

2011 年，美国计算机科学教师协会（CSTA）在《K-12 计算机科学标准（2011）》中指出，要把计算思维培养作为计算机科学课程的主要课程目标，（邱美玲、李海霞和罗丹，2018）这是美国继高等教育开展计算思维教育后在中小学开展计算思维教育的具体行动。2012 年，英国学校计算课程工作小组（Computing at School Working Group, CAS）将计算思维作为“学校计算机和和信息技术课程”的一项关键内容，强调计算思维作为解决问题能力的特征。（Computing at School Working Group, 2012）2016 年澳大利亚重新修订《国家数字技术课程标准》，明确提出发展学生的计算思维，要求学生“掌握组织数据、分解问题、解释模型、设计和实施算法等方法，具备设计和实施解决问题的方案，利用数字化工具实施方案，解决问题的能力”。（李锋，2018）2017 年，我国在新修订的高中信息技术课程标准中明确将“计算思维”信息技术学科核心素养的一项核心内容。

3. 面向计算思维的 STEM 课程设计框架

3.1. 计算思维与 STEM 教育的融合

有学者指出 STEM 领域在实验方法上有所复兴，主要是依赖强大的计算机功能的可用性，以及高度细节化的计算模型的开发。（Augustine, 2005）现在几乎所有的科学研究过程均涉及到“计算”，由于高速计算和分析方法的发展，科学家、工程师、数学家都在为越来越多的“计算”而努力。。计算技术的进步使跨学科的 STEM 研究人员能够设想出新的问题求解策略，并可在虚拟世界和现实世界中测试新的解决方案，在教育领域学生也迫切要学会使用计算方法和技术来支持 STEM 中快速变化的学科领域，那么将计算思维引入 STEM 课堂就十分必要了。

美国国家科学院（National Research Council）将计算思维的关键要素确定为——抽象、数据、检索、算法、设计、评估和可视化，（National Research Council, 2011）而 STEM 教育是知识经济时代下全新的跨学科教育形态，通过项目式学习、问题导向式学习等多种学习模式，面向真实问题解决，使用合适的思维方法与资源，进行探究，让学生参与更多社区交流，提高学生的学习兴趣，掌握知识和学习方法，培养学生的时代素养、创新思维和沟通能力。将计算思维注入 STEM 课堂符合 STEM 教育日益增长的计算需要，二者的具体关系如图 2 所示。

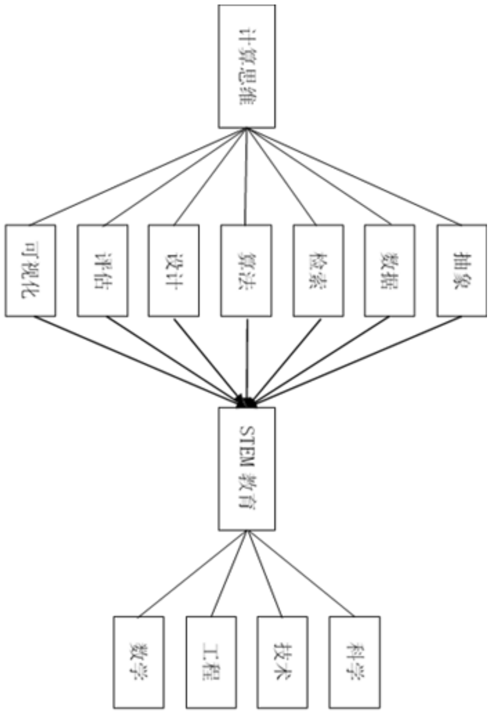


图 2 计算思维与 STEM 教育

3.2. 典型的 STEM 课程实践模型

国外较为成型的 STEM 教学模式大致有 3 种：5E 教学模式、6E 教学模式和 PIRPOSAL 模型。5E 教学模式最早是由美国生物学课程研究提出的一种基于建构主义学习视角的模式，其基本环节有引入（Engagement）、探究（Exploration）、解释（Explanation）、精致化（Elaboration）和评价（Evaluation）。在 5E 教学模式的基础上，Burke 提出了基于设计的 6E 学习模式，分别为引入（Engage）、探究（Explore）、解释（Explain）、设计（Engineer）、拓展（Enrich）和评价（Evaluate）。Wells 在分析科学探究、技术设计和工程设计过程的基础上，提出了整合性的 STEM 教学模型一一PIRPOSAL 模型，分成八个学习阶段：问题识别、产生想法、调查研究、可能的解决方案、最优化、方案评估、修改、学习成果。（蔡海云，2017）而根据现有研究，国内尚未形成比较完备的 STEM 教学模式。

3.3. *面向计算思维培养的 STEM 课程框架*

面向计算思维的 STEM 的课程关键是要凸显计算思维“系统设计、数据建模、系统设计”的属性，通过项目式学习、问题导向式学习的学习方式，引导学生在探究问题的过程中发展计算思维，提高利用计算思维分析和解决问题的能力。项目结构与内容设计是 STEM 课程设计的关键部分，借鉴已有研究成果，面向计算思维培养的STEM课程设计框架主要包括教学主题、教学目标、教学方式、教学资源、教学评价五个方面。图 3 是面向计算思维培养的 STEM 课程框架设计图。

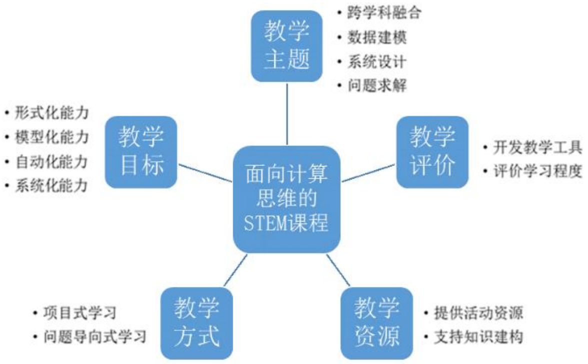


图 3 面向计算思维培养的 STEM 课程框架

4. 面向计算思维的 STEM 课程案例分析

4.1. *案例背景*

STEM + C（STEM 课程+计算思维）是美国全国卫生基金会（National Sanitation Foundation，NSF)资助的项目，该项目旨在为 4 至 6 年级学生在课外设计和开展 STEM 课程，并且将计算思维的相关元素注入STEM课程中，以促进学生学会利用计算思维分析和解决实际问题。项目以周以真（Jeannette M. Wing）教授提出的计算思维理念（计算思维是一种分析和解决问题的逻辑和过程，而不是简单的程序和编码）为基础，项目侧重于计算思维的两个方面：（1）在分析和解决问题的过程中涉及到的多种抽象（abstraction）层次；（2）在实验研究期间以计算机科学的方式进行研究成果的交流分享以及知识的可视化呈现。（Yang, Swasnon, Chittoori, & Baek, 2018）

4.2. *案例简纲*

该项目以基于项目的学习方法（PBL）为指导，通过围绕复杂、真实的问题，引导学生开展相关主题的研究，并通过积极制作作品来让学生学习知识和技能。在 PBL 单元结束时，学生通常通过竞赛或展览的方式来展示他们的最终产品。项目包括两个主题——设计探测火星生命的机器人和建造抗震桥梁，课程的详细内容如下表 1 所示。

表 1 STEM+C 项目课程内容

项目主题	火星上的生命	抗震桥梁设计与建造
项目描述	学生组成 2-3 人的学习小组，探究不同的生命形式以及火星的环境，设计并组装机器人以检测火星上的生命。	学生组成 4-6 人的小组，探究地震和桥梁的不同类型，设计并建造抗震桥，并且在模拟地震的情景下测试桥梁的稳定性。
学习目标	·探究监测火星生命的方法 ·设计并组装机器人监测火星上的生命	·调研地震和桥梁的相关知识 ·设计、建造抗震桥并进行测试
涉及学科	工程、科学、数学、技术、计算机科学	工程、地理科学、技术、数学
学习活动	·火星生命的调查研究 ·设计并组装机器人	·地震和桥梁的调查研究 ·设计、建造和测试抗震桥
学习评价	机器人在最短的时间内监测到火星上的生命	抵抗地震最强的桥梁

4.3. *案例说明*

该项目的两个课程主题均主要采用工程设计的方式，培养学生的实验探究和动手实践额能力。此外该项目将问题求解的过程划分为七个阶段：识别问题，研究问题，设计解决方案，选择最佳解决方案，构建原型，测试原型和评估原型，并且问题求解的每个阶段均与相关的计算思维元素相映射，以帮助学生在解决问题的过程中理解、培养、学习和应用计算思维，如图 4 所示。

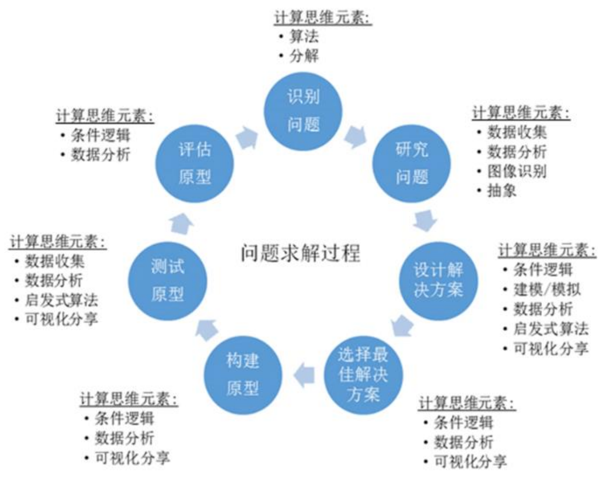


图 4 问题求解过程

课程结束后，研究人员对学生开展问卷调查和访谈，实验结果表明，完成 STEM + C 项目后，学生对数学有

了显著的更积极的态度。此外，STEM + C 项目为学生提供了一个可以进行科学探究和工程设计的学习环境，并且在这个学习过程中有助于学生学习和应用计算思维。例如，学生通过使用计算思维工具方法和专业术语来表达他们的知识和问题解决策略，通过数据分析、数据的可视化以及问题求解的相关算法，设计、开发、测试、制作相关产品，因此，该项目成功地将计算思维整合到中小学 STEM 课程学习中，培养和发展学生的计算思维能力，同时也助于学生计算思维的评估。

5. 结语

当以"程序驱动"为特征的信息技术工具渗透到社会各个领域并改变人们的学习、生活和工作方式时，计算思维教育成为中小学的一项重要教育内容，STEM 教育是对教学方式的一种质性描述，强调多学科知识的交叉融合，帮助学生解决生活中的真实问题，积极探索计算思维的培养模式，将其注入 STEM 课程以丰富 STEM 课程内容成为进一步培养计算思维的关键。本文从培养学生计算思维能力的目标出发，在厘清 计算思维与 STEM 教育关系的基础上，构建了一个面向计算思维培养的 STEM 课程框架，并给出具体教学案例进行解释和说明。以期将计算思维的培养落地于实际教育教学当中，为计算思维培养的实践应用提供有价值的借鉴。

6. 参考文献

黄崇福（1992）。**信息扩散原理与计算思维及其在地震工程中的应用**。北京：北京师范大学。

李锋（2018）。中小学计算思维教育:STEM 课程的视角。**中国远程教育**，**2**，44-49。

李锋和赵健（2016）。高中信息技术课程标准修订:理念与内容。**中国电化教育**，**12**，4-9。

邱美玲、李海霞和罗丹等（2018）。美国《K-12 计算机科学框架》对我国信息技术教学的启示。**现代教育技术**，**28**，41-47。

蔡海云（2017）。**STEM 教学模式的设计与实践研究**。上海:华东师范大学。

Augustine, N. R. (2005). *Rising Above the Gathering Storm: Energizing and Employing America for a Brighter Economic Future.* Washington D.C.: National Academies Press.

Computing at School Working Group (2012). *Computer Science: A Curriculum for School.* Retrieved March 1, 2012, from http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf.

National Research Council (2014). *How People Learn: Brain, Mind, Experience and School*: *Expanded Edition.* Retrieved March 1, 2014, from http://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf.http://www.csun.edu/~sb4310/How%20People%20Learn.pdf.

President's Information Technology Advisory Committee (2005). *Computational Science: Ensuring America's Competitiveness.* Retrieved May 27, 2005, from http://vis.cs.brown.edu/docs/pdf/Pitac-2005-CSE.pdf.

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

Yang, D., Swasnon, S. R., Chittoori, B. B. C., & Baek, Y. (2018). *Board 70: Work in Progress: Integrating Computational Thinking in STEM Education through a Project-based Learning Approach Paper.* Retrieved June 23, 2018, from https://peer.asee.org/30091.

Computational Thinking and Data Science

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Block Affordances for GraphQL in MIT App Inventor

Lujing CEN[1*], Evan W. PATTON[2*]
[12] MIT App Inventor, Massachusetts Institute of Technology, Cambridge, The United States
lujing@mit.edu, ewpatton@mit.edu

ABSTRACT

The rise of cloud computing and software as a service has brought along a significant change in the paradigm of application development. In this paper, we present how a relatively new Web application programming interface (API) abstraction, GraphQL, can be effectively represented in a block-based programming environment. Our work adds GraphQL client support to one such environment, MIT App Inventor, in order to demonstrate how this abstraction can help developers manage the complexity associated with the growing data layer of applications. In addition, we argue that although our implementation of the GraphQL component in App Inventor is not without limitations, it has significant advantages for students who are beginning to learn about Web APIs when compared to more traditional methods of interacting with endpoints. Our objective through future work is to prepare the GraphQL component for public release and further study how this abstraction will help students engage in computational action.

KEYWORDS

GraphQL, App Inventor, block languages, data science, web services

1. INTRODUCTION

1.1. What is GraphQL?

GraphQL is a strongly-typed query language developed by Facebook for describing and interacting with APIs (Schrok, 2015; Facebook, 2018). It can be seen as an alternative to traditional web frameworks that seeks to eliminate data over-fetching, minimize network round trips, and provide an introspective type system (Eizinger, 2017). Recently, GraphQL has gained significant traction in the web and software development communities through better tooling and increased commercial adoption.

```
type Character {
  id: ID!
  name: String!
  appearsIn: [Episode]!
}
```

Figure 1. An example GraphQL type definition for a character that appears in one or more TV show episodes.

GraphQL queries operate by selecting fields on objects. An endpoint will define a number of object types, each with its own set of fields (Figure 1). There are two main entry points for queries, the root *Query* and *Mutation* types, used for read and write operations respectively. Each query can be viewed as a tree, where leaves of the tree represent fields returning scalars (e.g., int, string). The response to a query is formatted in JavaScript Object Notation (JSON) and has the same shape as the query, making it easy to interpret and access the data (Figure 2).

```
{
  category(id: "pop") {
    playlists(limit: 1) {
      items {
        name
        owner {
          display_name
        }
      }
    }
  }
}
```

```
{
  "data": {
    "category": {
      "playlists": {
        "items": [
          {
            "name": "Today's Top Hits",
            "owner": {
              "display_name": "Spotify"
            }
          }
        ]
      }
    }
  }
}
```

Figure 2. An example GraphQL query (left) and the corresponding response (right).

The introspection system of GraphQL allows for queries on schema and type information. This means that it is possible to determine all of the field names, field arguments, and field descriptions for an object solely from the endpoint without consulting additional resources. With the proper tooling, introspection queries make it easier for users to construct well-formed queries through autocompletion, type checking, and documentation. Another benefit of introspection is that evolutions to the schema can be effectively communicated through the endpoint. Breaking changes and deprecations can be caught at compile time.

1.2. GraphQL Use Case in App Inventor

We present one sample application that a developer can build using GraphQL and App Inventor. Consider Alice, a student who is familiar with the block-based programming environment. She recently discovered that her digital music streaming service, Spotify, has a public Web API that lets her analyze tracks for their audio features. She also noticed that Spotify's own playlist search system does not provide users with enough details about whether a particular playlist is appropriate for a given event based on the properties of the tracks inside. Therefore, Alice decides to build a Spotify playlist analyzer that provides users with details about a playlist's danceability (how suitable the tracks are for dancing) and valence (how happy and cheerful the tracks are). That way, she can more easily find a playlist with low danceability and high valence for her next study session, or a playlist with high danceability and high valence for a party.

This application must interact with different resources of the Spotify API. Given that Alice has a rough idea of the data that she wants but does not know all the details of the API, she hopes to use the GraphQL component in App Inventor along with a Spotify GraphQL endpoint to help her construct the queries for her playlist analyzer. As we can see in Figure 3, the contents of the queries are very similar to a high-level description of the data that needs to be retrieved. This is one general benefit of GraphQL that we will explore in more depth through implementation details and discussions.

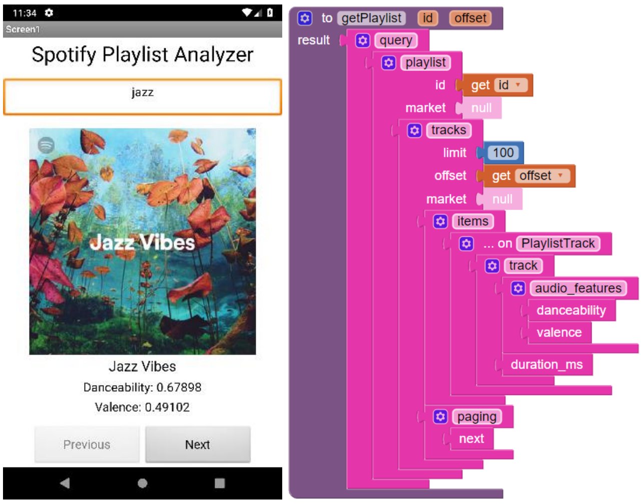


Figure 3. An example MIT App Inventor app for querying Spotify using GraphQL (left) and the corresponding query blocks (right).

2. THE GRAPHQL COMPONENT

We built a prototype of the GraphQL component in App Inventor that supports non-mutation and mutation queries against arbitrary GraphQL endpoints. The component was designed to be intuitive for users who are unfamiliar with the query language and consistent with existing App Inventor semantics. To that end, we have introduced a new dynamic block type that facilitates the construction of queries. We also designed various abstractions that make it easier to execute queries and interact with response data.

2.1. *Endpoint*

In order to interact with an endpoint, the user must supply a URL, which is stored in the *EndpointURL* property of the component. Some GraphQL endpoints may require client authentication before they can be queried. Therefore, the component exposes an additional property, *HttpHeaders*, where the user can specify headers that will be sent along with each query. Both of these properties are used to send a full introspection query to the target endpoint. The response schema is stored in memory for later use. Note that the introspection process is an exclusive exchange between the user's browser and the endpoint; it does not involve the App Inventor server in any way.

2.2. GQL Block Type

A GraphQL query consists of many selections, where a single selection is usually a field of an object. In a block-based environment, each selection can be modeled by a separate block. Given that selections act like functions, we eventually settled on a block representation similar to that of procedure calls in App Inventor (Figure 4). However, selections returning non-scalar values must also be able to specify its selection set, or the collection of desired fields from each of its returned objects. This can be done by augmenting the block with a mutator that allows users to add or remove spaces for items in the selection set. Together, these insights led to the creation of the new *gql* block type consisting of a field name, input values for arguments, indented input values for items in the selection set, and an output.

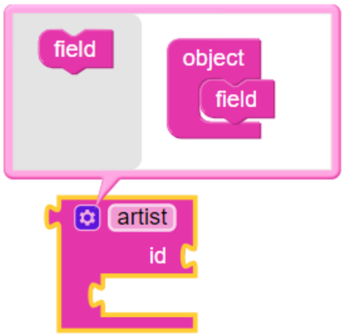


Figure 4. Mutator to extend a GraphQL query block.

All *gql* blocks are dynamically generated based on the associated endpoint. After the GraphQL schema is fetched and processed, the root level *query* and/or *mutation* selection blocks are injected into the instance's block list depending on what operations the endpoint supports. The user can then drag one of these blocks onto the workspace to begin the construction of a GraphQL query. Using type information from the schema, a *gql* block can automatically generate candidate blocks for its own selection set. These candidates are displayed in a flydown when the user hovers over the name of a non-scalar selection (Figure 5). If documentation for a selection exists, it is loaded into the tooltip of a *gql* block automatically.

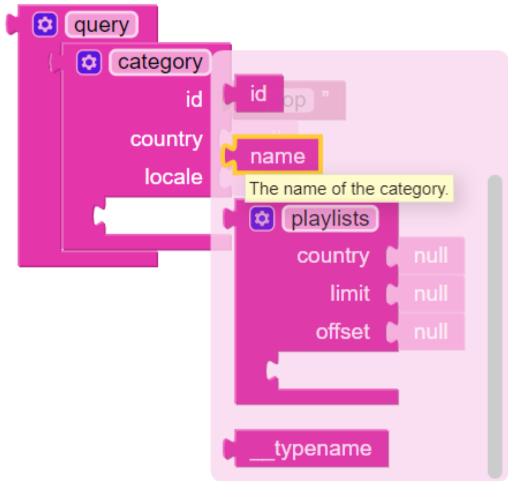


Figure 5. Autocompleted GraphQL blocks based on the GraphQL endpoint's schema.

At compile time, a *gql* block loses all type information and becomes a string join of its name and selection set with the appropriate whitespace and grouping brackets. To minimize the likelihood of constructing an invalid query prior to compilation, *gql* blocks leverage the existing connection compatibility system of MIT App Inventor to perform validations. Therefore, *gql* blocks will not allow themselves to be directly attached to invalid parent selections or to GraphQL component methods from other instances with different endpoints. Similarly, a *gql* block will ensure that all of its arguments are of the correct type. Nullable arguments, which are not present in App Inventor, are represented by *gql_null* shadow blocks. This ensures that users do not have to manually add or remove null value blocks during query construction.

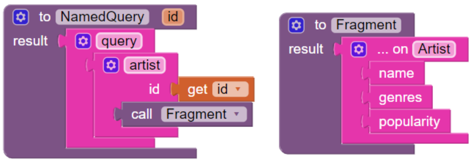


Figure 6. Example GraphQL fragment that can be plugged into other query definition blocks for query remixing.

Advanced GraphQL features, such as query fragments and fragment spreads, are also supported through native App Inventor semantics. GraphQL fragments are defined from an object type and can be reused through a fragment spread. They are useful for cases where the same query fragment might be used in multiple places. In App Inventor, fragments are built by creating a function that returns a fragment *gql* block (Figure 6). These fragment *gql* blocks are injected into a component instance's library during introspection, similar to the behavior of root selection blocks. A fragment spread is simply a function call in place of a selection *gql* block. Support for fragments means that this implementation is also able to query endpoints involving interfaces and union types.

2.3. Component Blocks

The GraphQL component has a single method *Query* that accepts two arguments—*queryName* and *query*. The *query* can be a string representing a GraphQL query, or a root *gql* block. The *queryName* is an identifier given to the *query*. This argument is necessary because App Inventor uses events to handle asynchronous execution. Some information must be passed between the request and the response, otherwise different query data will be difficult to distinguish. Upon calling the *Query* method, the component will send an appropriately formatted HTTP POST request to the target GraphQL endpoint.

When the JSON response to a query is received, the component will fire the *GotResponse* event if it received data or the *GotError* event if there were errors (Figure 7). This abstraction makes it much easier for users to interpret the response, since it is no longer necessary to manually determine whether a query was successful. The *response* variable of the *GotResponse* event is a map of the query results, encoded to the list of pairs format of App Inventor. Note that the map is intentionally stripped of the top-level *data* field. The *error* variable of the *GotError* event is a list of string messages reported by the GraphQL endpoint or a singleton list consisting of an error message from the component due to an error in sending the query or parsing the response.

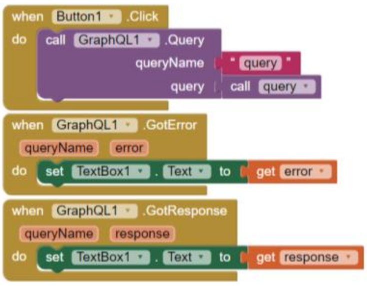


Figure 7. The GraphQL method and event blocks.

3. DISCUSSIONS

In this section, we present some of the pedagogical benefits of using the GraphQL component as a means of presenting Web APIs to students. We will also analyze some of the trade-offs present in GraphQL and the limitations of the current implementation in App Inventor. Finally, we look at ways of improving the GraphQL component through gathering feedback from users.

3.1. *Benefits*

GraphQL presents a relatively good tradeoff between flexibility and ease-of-use. Compared with traditional Web APIs, which are extremely versatile, GraphQL is more restrictive in the operations that it permits (Hartig & Pérez, 2017). However, this lowers the barrier of entry for students who are not familiar with the HTTP protocol or the API endpoint itself. When students are building an application that interfaces with a data source, a significant amount of time is spent on how to fetch data rather than what data needs to be fetched. Consider some of the steps necessary in interacting with a traditional Web API that is absent from GraphQL—encoding query parameters, choosing the right endpoint, interpreting the response code, and decoding the data. Ideally, these operations should be abstracted away from students when introducing them to Web APIs, which is exactly what GraphQL permits without being tied to a single endpoint.

There are some additional benefits that arise from the design of the GraphQL component in App Inventor. Queries constructed using dynamically generated *gql* blocks closely resemble the format of an actual GraphQL query, especially when inputs are inlined. This will help students smoothly transition from writing GraphQL queries in App Inventor to writing GraphQL queries in other environments. Certain GraphQL features such as fragments and named operations could have been implemented using GraphQL-specific blocks but are instead delegated to native App Inventor procedures. This presents opportunities to teach students about code reuse and function calls inside of queries, which is a feature present in many other query languages. Finally, generating candidate blocks can help reduce the likelihood of cognitive overload that results from working with an unfamiliar Web API. Students only need to be concerned with a limited number of possible selections for an object at each step of query construction.

3.2. *Limitations*

There are a few notable limitations to the current GraphQL component that the reader should be aware of. Most importantly, as with any Web API, the semantics of a GraphQL query is mostly dependent on the implementation of the endpoint. The user may still have to read through some documentation regarding particular arguments or fields to understand how to fetch the desired data and how to interpret the response. Some GraphQL endpoints define custom scalar types such as dates and binary-to-text encodings. It is up to the client to build support for encoding and decoding those data types. However, the current version of the GraphQL component treats unknown types as strings, which may be inconvenient to work with. Due to the lack of flexibility in GraphQL requests, there is not yet a formalized standard regarding mutations involving larger file uploads.

This is a trade-off made by the designers of GraphQL and is currently being addressed in the community (Seric, 2019).

3.3. Future Work

We are looking to improve various aspects of the GraphQL component for App Inventor and prepare it for an initial component release. One desirable feature that is currently missing is the ability to input object arguments in a way that does not involve using a JSON string. While the type information is available, more work needs to be done in terms of representing and validating non-scalar inputs. A reasonable solution is to use a dictionary to represent an input object once support for dictionaries is added to App Inventor (Patton & Tang, 2018). Some query optimizations can also be applied during compilation. For example, query fragments can be automatically located, extracted, and reused to reduce the request size. Implicit variables which are passed into a query can be separated from the query string itself because GraphQL permits named operations to have arguments that are sent along as a separate field in the HTTP POST request. This can benefit runtime performance since executing a query will no longer involve rebuilding the entire query string.

The efficacy of the GraphQL component in introducing students to Web APIs has yet to be evaluated. We hope to perform a more thorough analysis and comparison using the assessment framework presented by (Brennan & Resnick, 2012) once a full tutorial is developed for this component. Gathering feedback will allow for a better understanding of how students reason about query construction and data fetching. It will also provide insights into what potential features should be added to the component.

3.4. Related Work

Other query languages have been explored using a block-based paradigm. SPE Systemhaus, a company in Germany, provides a Google Blockly-based application for building Structured Query Language (SQL) queries. The SPARQL Protocol and RDF Query Language (SPARQL) is a query language for distributed information on the web represented using the Resource Description Framework (RDF) (Harris & Seaborne, 2013). A block-based programming tool for constructing SPARQL queries has also been explored by (Bottini & Ceriani, 2015; Ceriani & Bottini, 2017).

Some of the challenges associated with teaching students about SQL query construction have been addressed through the usage of Constraint-Based Modelling (Mitrovic, 1998). Instructional strategies for presenting relational thinking and evaluation methodology for assessing student understanding of SQL-based languages are outlined in (Dijk, 1992).

4. CONCLUSIONS

The importance of data in modern applications cannot be overstated. As a tool that teaches students programming and mobile application development, MIT App Inventor should also seek to emphasize interactions with external data sources. In this paper, we introduced GraphQL as a potential alternative for students who are starting to learn about Web APIs, query languages, and data manipulation. We also presented implementation details for our realization of the GraphQL component in App Inventor, which provides users with the proper abstractions and tooling to effectively write and execute GraphQL queries in a block-based environment. Although a more formal analysis of the purported benefits of using the GraphQL component is necessary, we believe that our work serves as a stepping stone for integrating GraphQL into App Inventor and educational curricula involving Web APIs.

5. REFERENCES

Bottoni, P., & Ceriani, M. (2015). SPARQL Playground: A Block Programming Tool to Experiment with SPARQL. In *VOILA@ ISWC*, 103.

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.* AERA, 25.

Ceriani, M., & Bottoni, P. (2017). SparqlBlocks: Using Blocks to Design Structured Linked Data Queries. *Journal of Visual Languages and Sentient Systems, 3*.

Dijk, E. M. (1992). Instructional Strategies for Teaching Database Query Languages. In *Instructional Models in Computer-based Learning Environments*, 279-289.

Eizinger, T. (2017). *API Design in Distributed Systems: A Comparison between GraphQL and REST*. Master's Thesis, University of Applied Sciences Technikum Wein.

Facebook. (2018). *GraphQL Specifications.* Retrieved January 17, 2019, from https://facebook.github.io/graphql/June2018/.

Harris, S. & Seaborne, A. (Eds.) (2013). *SPARQL 1.1 Query Language.* Retrieved January 26, 2019, from https://www.w3.org/TR/sparql11-query/.

Hartig, O., & Pérez, J. (2017). An Initial Analysis of Facebook's GraphQL language. In *AMW 2017 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web.*

Mitrovic, A. (1998). A Knowledge-based Teaching System for SQL. In *Proceedings of ED-MEDIA, 98*, 1027-1032.

Patton, E. W. & Tang, D. (2018). JSON Interoperability in MIT App Inventor: Thoughts on Terseness Versus Learnability. In *Proceedings of BLOCKS+ 2018.*

Schrok, N. (2015). *GraphQL Introduction.* Retrieved January 23, 2019, from https://reactjs.org/blog/2015/05/01/graphql-introduction.html

Seric, J. (2019). *GraphQL Multipart Request Specification.* Retrieved January 26, 2019, from https://github.com/jaydenseric/graphql-multipart-request-spec.

SPE Systemhaus. (2018). *SQL Blockly Workspace.* Retrieved January 26, 2019, from https://github.com/SPE-Systemhaus/blockly-sql-app.

Taelman, R., Vander Sande, M., & Verborgh, R. (2018). GraphQLLD: Linked Data Querying with GraphQL. In *17th International Semantic Web Conference*.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

An Integration of Computational Thinking into Teaching Activity Design for Learning Data Analysis and its Application

Yuan-yi, HUANG[1], Sung-chiang LIN[2*]
[12] Department of Mathematics and Information Education, National Taipei University of Education, Taiwan
huangyi905@gmail.com, lschiang@mail.ntue.edu.tw

ABSTRACT

With the advance in technology, the digital data increase rapidly. It is future trend to analyze massive data for evaluation, improvement and prediction. Data science is an interdisciplinary field including statistical methods, computer technologies and domain knowledge, and can apply scientific method to discover useful information from data. However, how to effectively analyze the original data to get useful information becomes an important research topic. Hence, this study will apply computational thinking on learning data analysis and train students' logical thinking and basic information skills through exploring and analyzing data.

KEYWORDS

computational thinking, cooperative learning, data analysis, statistical thinking

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

探究融入運算思維於學習資料分析的教學設計與應用

黃圓懿 [1]，林松江 [2*]
[1] 國立臺北教育大學數學暨資訊教育研究所，臺灣
[2] 國立臺北教育大學數學暨資訊教育學系，臺灣
huangyi905@gmail.com，lschiang@mail.ntue.edu.tw

摘要
科技的進步使得數位資料大量成長，透過運用大量數據的分析結果協助進行評估、改善及預測，已是未來的發展趨勢，包含教育專業領域也不例外。而資料科學包含的資料分析技術與觀念，有助於發掘資料中潛藏的有用資訊。但如何將原始雜亂的資料做有效的處理、分解與分析是重要的研究議題。因此，本研究結合運算思維於學習資料分析的議題上，透過資料探索的過程，培育學生的邏輯思考能力，強化基礎資訊技能。

關鍵字
運算思維；資料分析；合作學習；統計思維

1. 前言
資訊科技的蓬勃發展，使得各種計算與儲存的資料正快速地產生與累積，並隱含了豐富的訊息，而這些資料數據也在各行各業產生很多有效的應用，例如美國職棒、職籃等球隊，會依據球或球員的位置，跳的高度、角度等，來擬定有效的作戰策略，或是分析球員的體能表現；而在商業應用上，則可以分析出具有哪些特質的消費者會喜歡什麼樣的產品等。Mayer-Schönberger & Cukier （2014）在《大數據》一書中也提到「巨量資料的價值鏈中，誰掌握了最大的價值？在今天看來，似乎是那些掌握創新巨量資料思維的人。」而現今資料快速大量累積的情況下，如何培養統計思維（statistical thinking）已是重要的研究議題，更是有效進行資料處理與分析的基礎（黃文璋，2009）。

此外，利用網際網路進行教學的數位學習時代來臨，再加上國內外教育單位的重視與推廣，使得網路學習成為主流（陳年興和林甘敏，2002）。有許多廣為使用的數位教學平台，如 MOOC、可汗學院、均一教育平台等，逐漸成為學習者學習的重要管道之一。而由於線上學習以及行動學習的推廣，讓教學模式有了許多的改變，從學生和老師的教學應用和使用行為中，記錄了許多學習相關的數位化資料，若能透過新型的資訊科技方法讀取與處理，並利用統計方法分析大量資料，找尋資料中某一議題或特定現象的關鍵因素，進而探究資料裡潛藏的訊息，將可作為提供具體現況、趨勢預測和決策之參考（蔡明學和黃建翔，2015）。而培生教育研究與出版集團則在2014年2月出版的《數位海洋對教育的影響》（Impacts of the Digital Ocean on Education）報告書中指出快速累積的龐大數位教育資訊，對於教育可能的發展趨勢與未來願景提供了許多訊息，分析這些資訊將可作為了解學生學習、行為的表現，也是協助了解師生互動情形與成效的重要依據（DiCerbo & Behrens, 2014; Herold, 2014）。

而運算思維（computational thinking）則是一種用電腦的邏輯來解決問題的思維（Wing, 2008）；Google 在 Exploring Computational Thinking 網站中指出運算思維是一系列包含許多特性的問題解決歷程，例如：邏輯化進行排序與分析資料、循序的產出問題解決方法等，可適用於任一門學科（Google, 2015），包含拆解（Decomposition）、找出規律 （Pattern Recognition）、歸納與抽象化 （Pattern Generalization and Abstraction）、以及設計演算法 （Algorithm Design）等。

因此，本研究將探究結合運算思維於學習資料分析時的教學活動設計，應用於分析數位學習平台之測試資料，透過學習資料數據的分析與探索過程，培育學生的邏輯思考能力。

2. 文獻探討

2.1. 運算思維
近年來，運算思維已逐漸成為資訊科學教育的重要觀念，被認為是因應未來生活的基本能力之一，各國學者也相繼提出運算思維之定義，如「一種能利用電腦解決問題的思維，包含使用如抽象化、遞迴、迭代等概念來處理與分析資料，並產出實體與虛擬作品的能力」（CSTA, 2011）。換言之，運算思維具備以下特點：（1）是種觀念，而非撰寫程式；（2）是種基本的，而非死板的技能；（3）是關於人類解決問題的方法，而非電腦的；（4）結合了數學以及工程思維；（5）是種構想，而非作品；（6）能適用於每個人、每個地方（Wing, 2006）。Aho 則指出運算思維是架構問題的想法與流程，可運用步驟化及演算法方式呈現，最重要的部分為找出制定問題和解決方法的合適模型（Aho, 2012）。因此，運算思維可定義為能有效應用運算方法與工具解決問題之思維能力（林育慈和吳正己，2016）。美國國際科技教育協會（The International Society for Technology in Education, ISTE）則歸納描述運算思維的核心技能包含（1）把問題轉換成可用電腦或其他工具解決的形式、（2）有邏輯地整理與分析資料、（3）以抽象化的模式或模擬來表徵資料、（4）以演算法則建立問題解決的步驟、（5）分析各種有效解決問題的方案與資源、（6）能將問題解決的過程一般化並套用至解決其它的問題（ISTE, 2011），並以此延伸出九項運算思維思考歷程，包含：資料蒐集、資料分析、資料呈現、問題分解、抽象化、演算法與程序、自動化、模擬、平行化等。而現今教育領域對於資訊科技的應用越來越緊密，如何整理、擷取數據資

料並妥善使用、分析擷取關鍵訊息，已成為重要的議題與教師應具備之能力。

2.2. 資料科學、統計思維與資料分析

資料科學（Data Science）是一門跨領域的學問，目標是利用科學研究的方法從繁雜的「資料」中萃取出「有用的資訊」，並將這些資訊運用至等各個領域，以解決各領域面臨的問題，其內容涵蓋了資訊科學、數學與統計學以及專業領域知識等三大面向。許多數據資料分析的方向著重在資料指標的建立、資料的統計、資料之間的聯繫等，利用探索性資料分析的方式找出規律與知識，或者對未來事物的預測。因此，統計學可說是資料科學發展的源頭之一，包含了收集、分類、摘要、組織、分析與解釋資料訊息；統計思維（statistical thinking）則是用於描述變異無所不在的思維過程，其識別，表徵，量化，控制和精簡提供了改善決策的機會（Snee, 1986）。

而從資料本身來看，要進行探索與分析通常需要有：資訊收集、資料整合、資料精簡、資料清理、資料轉換、資料採擷過程、模式評估、以及知識表示等 8 個步驟（譚磊，2013），如下所述：

a) 資訊收集：根據確定的資料分析物件，選擇合適的資料收集方法，將收集到的資料存入資料庫。

b) 資料整合：把不同來源、格式、特性的資料在邏輯上或實體上集中，提供資料共用。

c) 資料精簡：當資料量大時，探索分析執行耗時，運用精簡技術可以得到資料集的精簡表示，並且精簡後的執行結果與精簡前執行結果相同或幾乎相同。

d) 資料清理：資料庫中的資料可能是不完整、不一致、或含雜訊的（包含錯誤的屬性值），因此，需要進行清理，確保資料的完整性、正確性與一致性。

e) 資料轉換：將資料轉換成適用的形式，例如透過概念分層和資料的離散化來轉換資料。

f) 資料採擷過程：根據資料資訊，選擇合適的分析工具，應用統計方法、決策樹、神經網路等機器學習演算法，以得出有用的分析資訊。

g) 模式評估：由領域專家來驗證資料探索、採擷結果的正確性。

h) 知識表示：將得到的分析資訊以視覺化得方式呈現給使用者，做為決策支援使用。

換言之，面對龐大的數據資料，其效益在於從中發現事物的模式與彼此的關聯性。以教育資料分析而言可分為兩類：（1）有關學生基本資訊的資料、（2）基於學生學習活動用以提升學習效果的數據，包括學習互動資料等（蔡明學和黃建翔，2015）。因此，對於教師來說，若能掌握核心運算思維概念與能力，於數位學習平台系統中進行「資料蒐集」，如結構性資料、動態數據等，接著利用已蒐集完成的數據進行「資料分析」使資料意義化，找出樣式及做出結論，再以合適的方式「呈現資料」。其目的為學生學習之「問題解決」，最後將概念「抽象化」，以數據資料為後盾，總結事實，從事實中預測學生行為，將有助於實施適性化教學。

2.3. 數位學習平台

隨著學習科技的發展，數位學習是目前教育中不可或缺的部分，而多元化的數位學習平台，除了可以幫助教師跟學生在網路上運用數位化教學工具系統進行數位學習，使學習者突破時間與空間的限制外，也使得收集學習者的活動資料加以分析成為可能，一方面可以用來提供新課程設計的參考，另一方面則可用來改善學習經驗。目前廣為使用的數位學習平台包含可汗學院以及均一教育平台等。

2.3.1. 可汗學院

西元 2006 年，由孟加拉裔美國人、麻省理工學院及哈佛大學商學院畢業生薩爾曼·可汗於 2006 年創立的一所非營利教育機構—可汗學院（Khan Academy）。此機構通過網絡提供一系列免費教材，內容適用於各個年齡層的個性化學習資源，並提供練習習題、教學視頻和個性化的學習介面，讓學習者能夠在課堂內外按照自己的進度學習（https://www.khanacademy.org/）。內容涉及數學、科學、電腦程式設計、歷史、藝術史、經濟學等。

2.3.2. 均一教育平台

2012 年成立的均一教育平台（Junyi Academy）（https://www.junyiacademy.org/），是目前全台最大的免費線上教育平台，每週穩定使用人數超過 4.8 萬人，內容涵蓋國小到高中課程的教學影片、練習題等，並持續由國內各領域專家學者錄製核心學習概念影片、互動式習題，提供學生自主學習及多元、零時差學習機會，創造更豐富優質的學習環境。

而近年來，陸續有許多均一教育平台之相關研究，如陳雪芝（2016）探討均一教育平台運用於國小四年級數學輔助教學之成效：以概數單元教學為例，發現均一教育平台的學習內容對於學生學習數學是適合的並能提升學習興趣與成效、張志豪（2017）的均一教育平台融入小組遊戲競賽進行國中數學補救教學之研究-以等差數列為例、以及張家豪（2018）的國民中小學運用均一教育平台於數學領域之個案研究，根據實際上操作的經驗來發現使用均一教育平台所面臨的困境與限制，從而發展出提升均一教育平台效益的策略。

3. 研究方法與設計

Mayer-Schönberger & Cukier（2014）指出運用大量資料分析，有助於讓教師了解怎麼教學最有效，學生又該如何學習，並舉出其在網路上開設的「機器學習」課程為例，透過追蹤學生觀看教學影片的動作，看學生會不會按暫停、快轉，甚至是提前切掉影片等跟學習有關的資料，做為幫助改善教學方式的參考，並用以提升學生的理解力。因此，要如何有效發揮這些資料的價值，釐清資料間錯綜複雜的關聯與因果，並從中探索分析，以協助教師作為改善教學的參考，也是資

訊教育領域重要的研究議題。而資料科學從「問出對的問題，才會得到對的答案。」觀點進行資料觀察與探索，其中包含的技術與觀念，將可協助發掘資料中潛藏的有用資訊。

3.1. 合作學習與 R 程式設計任務

程式設計是個需要邏輯推理能力的認知活動，在學習過程中能夠培養學生高層次思考以及邏輯推理的能力（Costelloe, 2004），過程中需經歷兩個階段：（1）必須先了解題目想出解決的方法（演算法），即問題解決的階段、（2）將問題的解法轉換為程式碼。而 R 語言的語法簡單、直覺，被認為是進行數據資料分析的合適工具（https://www.r-project.org/）。

合作學習（cooperative learning）則是以學生為本的一種教學策略與方式，透過將個別的學生組成小組或團隊，鼓勵小組成員間互助合作，共同討論和澄清想法、探究、推理以及解決問題，有助提升學習效益；Slavin（1985）也指出合作學習是一種有結構、有系統的教學策略，適用於大部分的學科及各個不同的年級。因此，本研究將以臺灣北部某大學學生為對象，在實驗過程中進行教學記錄，以進行結果分析與討論。

3.2. 資料來源與教育議題

本研究以 Datashop 網站上的學習互動資料集為對象（https://pslcdatashop.web.cmu.edu/），並挑選均一教育平台（Junyi Academy）做為研究測試資料集，資料收集區間為 2012 年 11 月 5 日至 2015 年 1 月 11 日，包含 359,804 筆資料紀錄。Mayer-Schönberger & Cukier（2014）認為資料分析對於學習而言將有助於即時讓學生知道學習成效、教師也能藉此修正教學內容，其在教學上的可能應用包含學習歷程的紀錄與分析、教學歷程的紀錄與分析、以及評量考試的紀錄與分析等三方面（孫憶明，2014）。換言之，透過數位學習平台上的互動紀錄，如學習者學習活動與時間、答題正確率、錯誤問題分析、解題提示等，均有助於教師進行教學分析與診斷。

3.3. 運算思維元素與資料分析

本研究的主要目的是結合運算思維於學習資料分析上，例如透過學生觀看教學影片的動作（如：暫停、快轉、回放、提前關閉影片）等跟學習有關的資料，利用運算思維與資料科學的資料分析技術，讓學習可以應用特定方法進行學習數據的統計分析與思考，做為改善教學與擬定教學策略的參考，將有助於探索與解決不同學習者的學習問題，以提升每個學生的學習成效並預測期結果。如下表 1 所示為統計資料分析步驟。表 2 則是運算思維元素與資料分析對照表。

表 1 資料分析步驟與流程

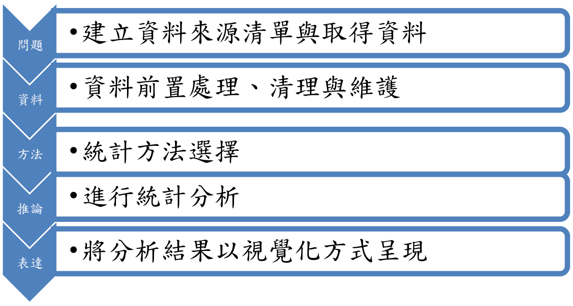


表 2 運算思維元素與資料分析參考（本研究整理）

運算思維元素	應用於資料分析
拆解	以學習者於平台操作的表現為基礎，將學習者拆解為不同學習行為進而探討分析
找出規律	從不同學習行為中找出隱性潛藏的行為規律
歸納	行為規律導向之測驗結果、做出結論
資料表示	將分析結果用適合的圖表、文字或圖片等視覺化方式呈現

同時，利用 R 程式語言進行運算思維與程式設計-資料分析活動設計與程序如下表 3；進行分後組，各小組在（1）定義問題時，須先提出待解問題說明、（2）在進行分工整理需要的資料、以及（3）說明需進行統計分析的資料，舉例來說，若想了解學生使用數位學習的情況，可透過收集「登入次數」、「觀看次數」、「瀏覽時間」、「問題回答次數」、「練習題的提示次數和質量」、「錯誤答案的重複次數」、「回答問題的反應時間」等紀錄來進行統計分析，例如 Pearson 相關分析、迴歸分析、或是分群等機器學習方法。

而由於使用者的學習行為非常多樣化，因此，以影片觀看記錄為例，本研究參考吳弘凱（2004）對於國小學童數位學習擷取課程行為樣式分析的結果（緩慢型、短暫型、深入型），依據學習者的使用行為記錄，區分為（1）略讀者：尚未將小節影片觀賞完畢即點選下一個小節、（2）閱讀者：完整將影片播放完畢，無作任何點選調整影片、（3）學習者：觀看影片時，有點選時間軸、暫停、回放、重看等動作，並將影片完整看完、以及（4）複習者：點選已觀看過的影片，進行影片回顧等 4 種。在練習題作答方面，則以計算正確率與錯誤率等作為歸納分類依據，並參考蔡旻芳（2001）分析學習路徑以輔助網路學習行為評量之研究，區分為精熟：連續作答正確、提示：請求提示深度（三階段）、錯誤：堅持不使用提示，並嘗試錯誤、以及回顧：點選相關影片回顧做題技巧與問題理解等。

同時，列舉有哪些圖表適合解釋資料，並將學習行為分析結果用視覺化的圖表等組織資料表示出來，以總結並表達趨勢進行預測，而每個單元的「後測」則做為學習行為歸納之檢驗分析。最後再依據分析結果提出建議與方案。表 3 則是本研究學習活動設計與任務，在準備教材與活動設計時，從現實問題中開始定義問

題，先具體後抽象，先簡易後複雜的將運算思維與統計知識融入，進而提高思考問題的層次，培養資料分析的邏輯思考與問題解決能力。

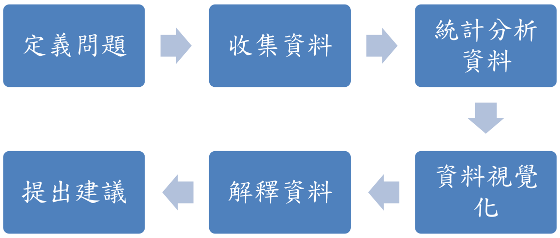


圖 1 學習活動設計與任務

4. 結論與討論

數位科技的發展，使得有越來越多的數據資料需要處理，而運算思維課程是為了培育學生的邏輯思考能力，強化基礎資訊技能。而資料科學從「問出對的問題，才會得到對的答案。」觀點進行資料觀察與探索，其中包含的資料分析技術與觀念，可協助發掘資料中潛藏的有用資訊。因此，本研究透過結合運算思維與資料分析的探索過程，讓使用者在運用 R 程式語言進行資料分析的過程中，透過將雜亂無章的資料做有效的處理、分解進而分析其結果，培養其邏輯思考與解決問題的能力，而適當的視覺化圖表呈現更能將分析結果有效呈現，以表達資訊擷取結果之關聯性。本研究以 Datashop 的數位學習平台測試資料集為例，來探索與協助識別出重要的學習活動和行為，其分析結果與建議應有助於改善教師教學策略，達到師生雙贏的效益。

5. 參考文獻

胡藝齡、顧小青和趙春（2014）。在線學習行為分析建模及挖掘，**開放教育研究**，**20**（**2**），102-110。

黃文璋（2009）。統計思維。**數學傳播**，**33**（**4**），30-46。

黃國禎、蘇俊銘和陳年興（2012）。**數位學習導論與實務**。新北市：博碩文化。

陳年興和林甘敏 （2002）。網路學習之學習行為與學習成效分析，**資訊管理學報**，**8**（**2**），121-133。

吳弘凱（2004）。**國小學童數位學習擷取課程行為樣式分析**（未出版之碩士論文）。台南：國立台南大學資訊教育研究所教學碩士班。

蔡明學和黃建翔（2015）。大數據分析在我國教育發展應用上之探討。**教育脈動**，**4**，154-164。

蔡昱芳（2001）。**分析學習路徑以輔助網路學習行為評量之研究**（未出版之碩士論文）。臺北：國立中山大學資訊管理學系研究所。

陳雪芝（2016）。**探討均一教育平台運用於國小四年級數學輔助教學之成效：以概數單元教學為例**（未出版之碩士論文）。高雄：義守大學資訊管理學系。

張志豪（2017）。**均一教育平台融入小組遊戲競賽進行國中數學補救教學之研究-以等差數列為例**（未出版之碩士論文）。臺北：淡江大學教程與教學研究所。

張家豪（2018）。**國民中小學運用均一教育平台於數學領域之個案研究**（未出版之碩士論文）。臺東：國立臺東大學教育學系教育行政碩士在職專班。

曾龍（2016）。大數據與巨量資料分析。**科學發展**，**524**，66-71。

孫憶明（2014）。**大數據（Big Data）改變未來教育樣貌的三種可能**。2014 年 2 月 10 日，取自 https://www.thenewslens.com/article/1970

譚磊（2013）。**大數據挖掘：從巨量資料發現別人看不到的秘密**。臺灣：上奇時代。

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

Costelloe, E. (2004). *Teaching Programming the State of the Art.* Dublin: The Center for Research in IT in Education, Dept. of Computer Science Education, Trinity College.

DiCerbo, K. E., Behrens, J. T., & Barber, M. (2014). *Impacts of the Digital Ocean on Education*. London: Pearson.

Google (2015). *Exploring Computational Thinking*. Retrieved January 18, 2019, from https://edu.google.com/resources/programs/exploring-computational-thinking/

Herold, B. (2014). *'Ocean' of Digital Data to Reshape Education, Pearson Report Predicts. The Education Week.* Retrieved March 19, 2014, from http://blogs.edweek.org/edweek/DigitalEducation/2014/03/ocean_of_digital_data_to_resha.html

ISTE (2011). *Operational Definition of Computational Thinking for K–12 Education.* Retrieved January 18, 2019, from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Lu, J. J., & Fletcher, G. H. (2009). Thinking about Computational Thinking. *ACM SIGCSE Bulletin*, *41*(1), 260-264.

Mayer-Schönberger, V., & Cukier, K. (2014)。**大數據：教育篇**（林俊宏譯）。臺北市：遠見天下文化。（原著出版於 2014）

Qualls, J. A., & Sherrell, L. B. (2010). Why Computational Thinking Should be Integrated into the Curriculum. *Journal of Computing Sciences in Colleges*, *25*(5), 66-71.

Slavin, R. E. (1985). An Instruction to Cooperative Learning Research. In Slavin, R., et al. (Eds.). *Learning to Cooperative, Cooperating to Learn*. Boston, MA: Springer, 5-16.

Snee, R. D. (1986). In Pursuit of Total Quality. *Quality Progress 20*(8), 25-31.

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

Zhang, M. (2014). Who are Interested in Online Science s Simulations? Tracking a Trend of Digital Divide in Internet Use. *Computer & Education*, *76*, 205-214.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Artificial Intelligence Education

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Classroom Activities for Teaching Artificial Intelligence to Primary School Students

Joshua W. K. HO[1*], Matthew SCADDING[2]
[1] School of Biomedical Sciences, The University of Hong Kong, Hong Kong
[1] Victor Chang Cardiac Research Institute, Darlinghurst, NSW 2010, Australia
[2] Ravenswood School for Girls, Gordon, NSW 2072, Australia
jwkho@hku.hk, mscadding@ravenswood.nsw.edu.au

ABSTRACT

It is inevitable that the primary school students today will grow up in a world in which computer programs that incorporate artificial intelligence (AI) capabilities will be prevalent in many aspects of their lives and their future workplaces. In our experience, primary school students often find the concept of AI quite mysterious and possibly scary. Teachers are often not well equipped to thoroughly explain key concepts in AI. This paper summarises our own experience designing and implementing classroom activities for teaching fundamental concepts of AI to Year 6 students in a school in Sydney, Australia. The main goal of our activities is to demystify AI by showing them AI can be thought of different ways in which a computer simulates human-like behaviours. In particular, we present two hands-on activities – an unplugged activity on facial recognition, and a simple robotic exercise that introduces the concept of machine learning. We hope this paper will ignite discussions about how AI can be taught effectively at the K-12 levels.

KEYWORDS

K-12 education, artificial intelligence, robotics, education, unplugged activities

1. INTRODUCTION

We now live in a world in which children grow up with software programs that have a large variety of artificial intelligence (AI) capabilities, such as facial recognition and speech recognition. These AI-enabled programs are widely accessible through smartphones, and websites. Our own experience indicates that primary school students are often fascinated about all these AI technologies, but at the same time they found the idea that computers can exhibit such intelligent behaviors quite mysterious, and sometimes scary. Standard K-12 curricula often do not explicitly introduce the concept of AI. Furthermore, most teachers are not well equipped to discuss the computational aspects of AI in their classroom in a way that is suitable at the K-12 levels.

Currently, AI is often only taught at the university level, and these courses are mostly designed for computer science and engineering students. Some game-based activities have been developed to teach AI at the university level (DeNero & Klein, 2010; Wong et al, 2010). The main concept is to make AI classes engaging by incorporating games into programming exercises. These activities involve computer programming and knowledge of search algorithms; therefore they are not necessarily suitable for K-12 students.

We would like to explore whether some computational concepts underlying modern AI applications can be introduced at the primary school level. Through the CSIRO STEM Professionals in Schools program, we have been designing and running new classroom activities that introduces some basic computational concepts of AI into Year 6 STEM classes at the Ravenswood School for Girls at Sydney, Australia, between 2016-2017 (Earp, 2017).

This paper summarises the key design principles behind our classroom activities for teaching AI to primary school students, and present a brief summary of two specific activities, in which their details can be found online (Ho, 2018a, 2018b):

1. An unplugged activity for teaching the concept of facial recognition
2. A simple robotic activity for teaching the concept of machine learning

2. DESIGN PRINCIPLES

In our classes, we always stress that a computer program is said to have AI capability because it *simulate*s some aspects of human behaviours. We started our series of AI classes by asking our students whether they thought a computer could think. This often led to some interesting discussions about the role of the programmer and what was the nature of human intelligence. We then explained that this exact question was as old as modern computer science. We then introduced the Turing Test – an imaginary test developed by Alan Turing that states that a computer can be said to have intelligence if it is indistinguishable from a human by a human interrogator who can only interact with a computer and a human through natural language conversation. The point of introducing the Turing Test is to illustrate that a computer program can be said to have AI if it exhibits some aspects of human-like behaviours. From our experience, introducing the Turing Test is useful because it *demystifies* AI. It shifts the focus from defining the arguably vague concept of 'intelligence' into more concrete computational tasks of simulating specific human-like behaviours. The activities that we developed involve breaking down various human-like behaviours and showing our students how some simple algorithms can be used to simulate behaviours such as 'seeing' (facial recognition) and 'learning' (machine learning).

We following several simple design principles when developing our activities:

1. Explain how a problem can be solved by simple computational elements. In every lesson we always stressed that all seemingly complicated algorithms consisted of simple computational elements in which they were familiar, such as loops (for repetition), if-else statements (for decision), and variables (for data storage). We found that this principle was useful in demystifying AI.
2. Create group activities that involve both design and execution. All activities were designed to involve group work inside the classroom. Time was given to let them discuss the design the algorithm, either as the whole class or as individual groups, prior to the teacher revealing a solution.
3. Incorporate the element of game into the activities. To engage our Year 6 students, we designed our activities as games. We utilised unplugged activities as much as possible.
4. Select AI tasks in which the students are familiar. AI encompasses a wide variety of areas. We chose areas in which primary school students are familiar.

3. ACTIVITY 1: FACIAL RECOGNITION

Our first activity involves facial recognition. In the beginning of the lesson, we showed a video about how a facial recognition program can identify the name of people walking pass a security camera in real-time. We led a discussion about how the students might design an algorithm that could name the person in a picture. One key point we would like them to learn was that for a facial recognition program to function properly, it required a large database of personal photos where the name of the person in each photo was known. This is the concept of training data. Another key that we wanted to get them to understand was that this task was much more difficult than comparing the picture with a collection of facial images, because the same person could have slightly different physical appearances in different pictures due to movement, lighting, clothing, and other factors. A simple pixel-by-pixel comparison would not work. Usually after some discussions, some students would work out that it was possible to match a person based on some characteristics – such as hair colour, hair length, eye colour, glasses, height, and so on. This is the concept of feature extraction. Following this discussion, we proceeded to an unplugged activity called 'Who is this princess?' which illustrated how training data and feature extraction could be used to develop a facial recognition algorithm.

In this activity, we first asked six students to come to the front of the classroom, and picked up a photo of a Disney princess. Disney princesses were used in this game because all the students in this girl's school are familiar with these cartoon characters. The choice of these characters also made the class more engaging. It is possible to replace them with other characters that are well known to the students. These six students were asked to answer a series of five questions about the character they picked without showing the photo to anyone else (Figure 1). These six photos would become the 'database' for this facial recognition task. Afterward, one additional student was called from the rest of the class to pick up a new photo (which contained the same character as one of the six photos in the 'database'). This student was asked to answer the same five questions without showing the photo to anyone else (Figure 1). Then, this last student had to walk in front every student in the database group and compare the answers of the five questions, and assign a similarity score which indicated how many answers were the same. In this end, all the students were asked to reveal their photos to the class, and we could check whether the photo with the highest score was indeed the correctly chosen photo.

Name tag	Snow white	Belle	Jasmine	Jasmine
Dress colour	Blue	Yellow	Green	Green
Long hair?	No	Yes	Yes	Yes
Dark hair colour?	Yes	No	Yes	Yes
Dark skin colour?	No	No	Yes	Yes
Holding something?	No	Yes	Yes	No
Similarity score	2	1	4	

Figure 1. Extraction and comparison of features of different Disney princesses.

In the example in Figure 1, the database consists of three princesses (Snow white, Belle, and Jasmine). Based on the features, the unnamed princess (the right most column) is likely Jasmine since it has the highly similarity score (bottom row) to Jasmine in the database.

This unplugged activity is a physical way to act out the process of feature extraction and the nearest-neighbour algorithm in machine learning. One important concept to explain to the students is that feature extraction is the key to convert a seemingly difficult problem (matching images) into a simpler problem (matching simple features). Also, it should be pointed out that the success of this facial recognition program depends on having a large number of high quality photos in the database. This is an opportunity to introduce the concept of big data.

4. ACTIVITY 2: MACHINE LEARNING

Our next activity explores how an AI program can learn from past experience. This is the field of machine learning. We conducted this activity across two weeks.

In the first week, we played a 'number guessing game'. This activity involved building a simple program using the Lego Mindstorms EV3 kit. The goal of this activity is to build a program (the robot) that can 'intelligently' guess a number that the human opponent is guessing in the smallest number of attempts. This game proceeded as follows: the human player (i.e., the student) came up with a number between 1 and 100 in their mind. The robot needed to guess that number. After each robot guess, the human had to provide feedback to tell the robot whether its guess was too large, too small or correct. This process was repeated until the correct answer was achieved. A robot can be thought of as more intelligent if it could make the correct guess in a smaller number of attempts.

Before the students started coding, we asked them to consider three strategies a robot could take:

1. Each time randomly generate a number between 1 and 100, regardless of the feedback received in the previous rounds
2. Systematically guess 1, 2, 3, ... until the correct answer was reached
3. Each time randomly generate a number within the current range (start with 1-100), and progressively narrow down this range to a smaller range using feedback provided in previous attempts.

Most students could see that the third strategy was the best because it was ‘learning’ from the feedback provided by the human player. Both strategy 1 and strategy 2 do not make use of feedback from the human player. While all three strategies would eventually find the correct answer, only the third strategy could be considered as having the ability to ‘learn’. This strategy is essentially the binary search algorithm in computer science. It is one of the simplest search algorithms. This type of search and optimisation algorithms is at the heart of modern machine learning algorithms. This number guessing game illustrates the concept of learning by trial-and-error. Using this simple concept, this number guessing game can be seen as a simple machine learning program.

In the second week of the machine learning activity, we turned the number guessing robot into a self-learning lawn bowling robot. Lawn bowling is a game of accuracy. The goal of the game is to bowl the ball so that it is closest to the target. In this sense, the game of lawn bowling is similar to the number guessing game. In the number guessing game, the robot iteratively narrows down the range which contains the human’s number. In the lawn bowling game, the robot iteratively learns the range of speed its robotic arm needs to swing in order for the ball to be as close as possible to the target. In both games, human feedback after a robot’s action is critical to the learning process.

We asked the students to build a robotic arm and a lawn bowling ball for the robot (Figure 2). The students could re-use the source code from the number guessing program, and modified the code to make the robotic arm swing instead of displaying a number guess. This means this lawn bowling robot ‘learns’ the optimal bowling speed based on repeated bowling attempts. These repeated attempts can be seen as ‘training’ in the context of machine learning. This lawn bowling robot can be seen as ‘learning’ by trial-and-error. From our experience, most students were amazed by the apparent ability of the robot to learn how to bowl in just a few rounds of training.

To create further engagement, we asked the students to participate in a ‘robotic lawn bowling competition’ inside the classroom. All the students were given sufficient time to train their robots. They could even come up with their own design for the robotic arms and computer source code. Our students had a lot of fun.

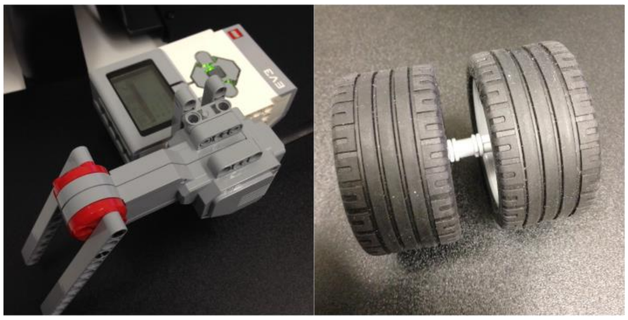


Figure 2. Pictures of the lawn bawling robot and the lawn ball that were assembled using Lego.

The core concept is that machine learning can be achieved by trial-and-error. This is a concept that is quite readily understood by our Year 6 students because it relates to how they learn new skills, such as mastering a new sport, learning to play a new musical instruments, and learning to speak in a new language. A learner is more effective if it readily incorporates feedback into their learning process. Therefore, in addition to teaching them machine learning, this activity also provided an opportunity for our students to reflect on their own learning experience and how they can utilise feedback more effectively in their own learning.

5. DISCUSSION

In this paper, we presented our own experience in designing and implementing classroom activities for teaching basic AI concepts to Year 6 students. The goal of our work is to demonstrate that some seemingly complex concepts such as facial recognition and machine learning can be explained in terms of simple computer algorithms that simulate specific human-like behaviours. We hope that these classroom activities can spark further development of proper pedagogical approaches to teaching AI at the K-12 levels.

6. REFERENCES

DeNero, J., & Klein, D. (2010). Teaching Introductory Artificial Intelligence with Pac-Man. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence.* AAAI, 1885-1889.

Earp, J. (2017). Artificial Intelligence, Robotics and Coding. Retrieved September 26, 2017, from https://www.teachermagazine.com.au/articles/artificial-intelligence-robotics-and-coding

Ho, J.W.K. (2018a). AI Classroom Activity: Facial Recognition. Retrieved March 21, 2018, from https://www.teachermagazine.com.au/articles/ai-classroom-activity-facial-recognition

Ho, J. W. K. (2018b). AI Classroom Activity: Machine Learning. Retrieved March 28, 2018, from https://www.teachermagazine.com.au/articles/ai-classroom-activity-machine-learning

Wong, D., Zink, R., & Koenig, S. (2010). Teaching Artificial Intelligence and Robotics Via Games. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence.* AAAI, 1917-1918.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Popstar, the Poet, and the Grinch: Relating Artificial Intelligence to the Computational Thinking Framework with Block-based Coding

Jessica Van BRUMMELEN[1*], Judy Hanwen SHEN[2], Evan W. PATTON[3]
[123] Massachusetts Institute of Technology, The United States
jess@csail.mit.edu, judyshen@mit.edu, ewpatton@mit.edu

ABSTRACT

As the world becomes increasingly saturated with artificial intelligence (AI) technology, computational thinking (CT) frameworks must be updated to incorporate AI concepts. In this paper, we propose five AI-related computation concepts, practice, and perspective: classification, prediction, generation, training/validating/testing, and evaluation. We propose adding them to a widely-used CT framework and present an MIT App Inventor extension that explores this framework through project-based learning.

KEYWORDS

artificial intelligence, conversational artificial intelligence, machine learning, computational thinking, K-12 education

1. INTRODUCTION

There are many who would like to understand and use artificial intelligence (AI) models but lack the tools and knowledge to do so. We propose adding AI-related concepts to Brennan and Resnick's (2012) CT framework, as well as present block-based coding tools to democratize AI education and programming. Our proposed tools were developed for MIT App Inventor, an open-source platform that enables anyone to develop mobile apps using block-based coding with eight million registered users from primary-school aged students to working-class adults (MIT App Inventor, 2017).

Malyn-Smith et. al. (2018) developed a CT framework from a disciplinary perspective which includes machine learning as an element. To this end, we propose computational concepts, practice, and perspective that more effectively capture the skills and competencies necessary to understand AI. Using conversational AI, AI-related components, including classification, prediction, generation, training/validating/testing, and evaluation, are explained. We present an MIT App Inventor extension to enable students to learn the proposed AI CT components.

2. EXTENDING CT WITH AI

Artificial intelligence can be understood within a symbolic-rule/machine-learning paradigm. In symbolic rule-based AI, collections of *if-then* statements or other rules determine how AI agents behave (Winston & Shellard, 1990). In machine learning-based AI, machines determine how to behave through extracting patterns. Both methods have shortcomings, such as the difficulty of programming an exhaustive list of rules for rule-based AI, and the limited interpretability of machine learning models.

Within the symbolic-rule/machine-learning paradigm, designers use AI to classify, predict, and generate information. For example, an autonomous vehicle may perceive objects on the road and *classify* them as "pedestrians" or "motor vehicles"; *predict* objects' motion; and *generate* vehicle speed (Van Brummelen, O'Brien, Gruyer, & Najjaran, 2018). We propose adding three AI-related concepts to Brennan and Resnick's CT framework: *Classification*, *Prediction*, and *Generation*.

Classification. Machines often sort information into categories for downstream decision-making through rules (e.g., "the sentence is positive because it contains 'happy'") or learning algorithms (e.g., after observing "positive" sentences, similar sentences are classified as "positive").

Prediction. To act intelligently, machines predict future values and behavior. This includes predicting the category an object may fall into, an object's future behavior, or the best action to take next (e.g., after saying, "I am a", a conversational agent may predict the next best word to be "robot").

Generation. Using information gathered, machines can generate new data. This may include synthesizing previous examples, creating new information, or making decisions (e.g., a machine constructing and speaking a new sentence**).**

We also propose the following AI-related practice:

Training, Validating, and Testing. Developing a robust ML model requires waiting for the model to learn to recognize patterns, testing if it generates correct predictions, and determining if it is sufficient for the task. Training involves providing examples (or an environment) for the model to iteratively learn from (or experiment in). Testing and validating involves providing different examples (or environments) to observe how the model behaves, comparing the model's behavior to other models, and determining whether the model is sufficient. This includes assessing the *accuracy* of the model (e.g., the percentage of correct classifications) using a test and/or validation dataset and the model *loss* (a value used to update model weights during training).

Finally, we propose the following AI-related perspectives:

Evaluation. Some AI (e.g., neural networks and other learning techniques) behavior can be difficult to predict or unintuitive to humans. Programmers must think about how well the program behaves and whether it achieves the necessary goals (e.g., How can we improve the program? Did we over- or under-train the model? Is the model biased towards certain people because it was trained by them?). These considerations are especially pertinent when considering the large number of input-output relationships with ML.

Evaluation is performed in the context of the final product or application, whereas testing and validating are performed only considering the model itself. For example, during evaluation, one might ask, "Is my app biased towards classifying people as middle-aged?"; whereas during validation, one may ask, "Why is my model achieving 42% accuracy?".

3. CONVERSATIONAL AI EXTENSION

The conversational AI extension for MIT App Inventor, or *Text Mixer*, generates text based on three input corpora: Dr. Seuss books, Taylor Swift lyrics, and Shakespearean poetry. It enables students to generate text resembling the input corpora by providing corpora weights (*mixture coefficients)*. The model contains three, single-layer LSTM models with 30 hidden units and static GloVe embeddings (*6B*, *d=300*) (Pennington, Socher, & Manning, 2014).

- **SEUSS**: The Dr. Seuss collection of children's poetry contains word coverage representative of children's literature at a K-3 reading level (Foster & Mackie, 2013).
- **SWIFT**: Lyrics from the popular artist Taylor Swift contain colloquialisms, interjections, and repetitions and focus on the themes of love and heartbreak from an adolescent perspective (Kotarba, 2013).
- **SHAKESPEARE**: These works are featured in university and high school English courses, and consist of Early Modern English verses (G. T. Wright, 1983).

To generate response-sentences from a mixture of language models, we resampled the model at each word-generation step. For each word, we sample from the three models based on the mixture coefficients inputted to the block (See Figure 1). Using the sampled model, we feed in the input sequence of previously generated words until an end-of-sentence token or the maximum length has been reached. Upon completion, we have both a sequence of newly generated words and the corresponding list of language models each word was generated from.

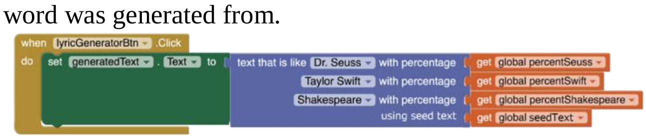


Figure 1. Example Text Mixer block in MIT App Inventor

4. RELATION TO CT FRAMEWORK

With the conversational AI *Text Mixer* extension, students can explore the proposed AI concepts, practice, and perspective.

Classification. In the Duet App, the *Text Mixer* extension contains ML models to classify and organize words in latent spaces (the representation spaces where neural networks organize information). The models use this organization to determine words' likelihood to appear next in the sentence.

Prediction. The *Text Mixer* extension predicts the best next word in the sentence by using three ML language models. The mixture coefficients determine how often a model is chosen (e.g., if the Dr. Seuss mixture coefficient is relatively high, then the Dr. Seuss language model will likely be chosen). The ML language models then use the "seed text", the previous words in the sentence being generated (if any), and previously-trained weights to predict the best next word.

Generation. After predicting a word, the *Text Mixer* adds this word to the sentence. This repeats until a full sentence is generated. Once a sentence has been constructed, the *Duet App* speaks the words aloud while playing music.

Training, Validating, and Testing. The *Text Mixer* block exemplifies training by enabling students to choose machine learning models trained on different input corpora. This block is meant to be a high-level introduction to ML, so it does not necessarily exemplify testing and validating. The authors plan to develop blocks for testing and validation in future work.

Evaluation. After developing the app, students can evaluate the output to determine whether the model generates song lyrics adequate for their application.

5. CONCLUSION

CT frameworks need to be updated to continue to be relevant in an increasingly AI-powered world. This paper proposes AI-related CT concepts, practice, and perspective, building on Brennan and Resnick's (2012) framework and presents an MIT App Inventor AI extension. The skills learned through experimenting with the *Text Mixer* integrate smoothly with the CT framework and help students to better understand AI.

6. REFERENCES

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: American Educational Research Association.

Foster, J., & Mackie, C. (2013). Lexical Analysis of the Dr. Seuss Corpus. *Concordia Working Papers in Applied Linguistics, 4,* 1-21.

Kotarba, J. A. (2013). *Understanding Society through Popular Music*. New York: Routledge.

Malyn-Smith, J., Lee, I. A, Martin, F., Grover, S., Evans, M. A, & Pillai, S. (2018). Developing a Framework for Computational Thinking from a Disciplinary Perspective. *Proceedings of the International Conference on Computational Thinking in Education 2018*. Hong Kong: The Education University of Hong Kong, 182-186.

MIT App Inventor. (2017). *Anyone Can Build Apps That Impact the World.* Retrieved January 24, 2019, from http://appinventor.mit.edu/explore/

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).* Association for Computational Linguistics, 1532-1543.

Van Brummelen, J., O'Brien, M., Gruyer, D., & Najjaran, H. (2018). Autonomous Vehicle Perception: The Technology of Today and Tomorrow. *Transportation Research Part C: Emerging Technologies, 89,* 384-406.

Winston, P. H., & Shellard, S. A. (1990). *Artificial Intelligence at MIT: Expanding Frontiers*. US: MIT Press.

Wright, G. T. (1983). The Play of Phrase and Line In Shakespeare's Iambic Pentameter. *Shakespeare Quarterly, 34*(2), 147-158.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking Development in Higher Education

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Preliminary Study of Project-based Learning Teaching Activity for Programming based on Computational Thinking

Zi-yun LU[1], Sung-chiang LIN[2*]
[12] Department of Mathematics and Information Education, National Taipei University of Education, Taiwan
d03410019@ems.npu.edu.tw, lschiang@mail.ntue.edu.tw

ABSTRACT
The purpose of programming education is to train students with the abilities of logical thinking and solving problems. In the basic education stage, many developed countries are promoting programming education based on computational thinking in order to respond to rapid advance on science and technology. However, it is an important issue that how to teach programming is beneficial to students and let students be able to apply the problem-solving skills to other knowledge fields. In addition, project-based learning is a teaching method in which students gain knowledge and skills through learning by doing and can train students' self-confidence, teamwork and self-learning ability. Hence, in this study, we designed a project-based learning teaching activity to teach students programming based on computational thinking. The results show that project-based learning can effectively enhance students' motivation, satisfaction and participation.

KEYWORDS
computational thinking, programming instruction, project-based learning, virtual reality

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

基於運算思維之 PBL 程式設計教學活動成效初探

呂子芸 [1]，林松江 [2*]
[1] 國立臺北教育大學數學暨資訊教育研究所，臺灣
[2] 國立臺北教育大學數學暨資訊教育學系，臺灣
d03410019@ems.npu.edu.tw，lschiang@mail.ntue.edu.tw

摘要

程式設計教學目標在於培養學生的邏輯思考與解決問題的能力，許多先進國家開始在基礎教育階段推動以運算思維為核心的程式設計教學，以因應科技的快速發展。但如何進行程式設計教學，才有利於學習者學習，以促進將所習得的問題解決技能類化到其它知識領域，則一直是重要的研究議題。而專題導向學習的特色便在於可從做中學培養學生的自信心、團隊合作和自學能力。因此，本研究將以運算思維為基礎，結合專題導向學習的方式進行程式設計教學活動設計。研究結果指出，透過專題導向學習的方式可有效提升學生自主學習的動力、成就感、以及參與感。

關鍵字

運算思維；程式設計教學；專題導向學習；虛擬實境。

1. 前言

科技日新月異快速發展，使得許多先進國家積極推廣資訊科技課程，因此，培養國民的資訊科技知能已成為世界各國教育的新趨勢，其中，美國電腦科學教師協會（Computer Science Teachers Association, CSTA）則進一步指出資訊科學課程應包含五大面向，分別是運算思維（Computational thinking）、合作（Collaboration）、運算實作與程式設計（Computing Practice and Programming）、電腦與通訊設備（Computers and Communications Devices）以及社會、全球與倫理衝擊（Community Global and Ethical Impacts）等（CSTA, 2011）。

此外，隨著許多國家開始在基礎教育階段推動以運算思維（Computational thinking）為核心的程式設計教學後，也使得程式設計教學活動再度成為資訊教育研究領域相當熱門的研究議題，例如美國、英國等國家，也已將程式設計課程納入課綱中（Schoolnet, 2015）。教育部（2016）則在運算思維計畫中，以扎根程式學習、培養運算思維、接軌國際活動為目標，鼓勵學生動手做，發揮創意，透過觀察，運用電腦科學的角度培養邏輯思考的能力。而程式設計教學目標主要在於讓學生思考如何運用程式語言的演算法則與設計技巧來解決特定的問題，其學習重點在於培養學生邏輯與抽象思考的能力，進而發展在不同領域問題解決的認知基模（Wing, 2006；Fernaeus, Kindborg, & Scholz, 2006；Wing, 2008）。

換言之，如何讓學生在程式設計實作的過程中，經由設計出有意義的專案，並從中獲得重要的知識與技能是非常重要的；而專題導向學習（Project-based learning, PBL）模式的特色便在於可從做中學（learning by doing）培養學生的自信心、團隊合作和自學能力，有助於學生在培養解決問題能力的學習成效（Larry, 2015）。因此，本研究將結合運算思維與專題導向學習進行程式設計教學活動設計與探究。

2. 文獻探討

2.1. 運算思維

自從運算思維被提出後，許多學者對於運算思維一詞都有其不同的見解，其中多數學者均認為運算思維應包含抽象化（Abstraction）、資料表示（Data Representation）、問題解析（Problem Decomposition）及演算法思維（Algorithm Thinking）（Wing, 2006；Grover & Pea, 2013；Google, 2019）。Barr 與 Stephenson（2011）則提出運算思維運用於各領域之範例，並將運算思維元素分為資料搜集、資料分析、資料表示、問題解析、抽象化、演算法思維、自動化、同步、模擬等，並將各項元素對應至資訊科學、數學、科學、社會研究與語言藝術，如下所述：

10) 資料蒐集（Data Collection）：透過發現問題蒐集可以分析的資料。

11) 資料分析（Data Analysis）：將問題進行分類。

12) 資料表示（Data Representation）：將問題進行拆解，例如在資訊科學方面可以使用資料結構，將資料以陣列（Array）、鏈結串列（Linked list）、堆疊（Stack）、佇列（Queue）、圖片（Graph）和雜湊表（Hash table）等進行配置。

13) 問題解析（Problem Decomposition）：定義問題可以用哪種解決方法。

14) 抽象化（Abstraction）：透過有系統的包裝，並將問題利用現有的方法解決。

15) 演算法思維（Algorithms & Procedures）：以資料科學為例，透過學習經典演算法，並針對某一特定領域的問題進行實作演算法。

16) 自動化（Automation）：將問題透過自動化的方式解決，如生活中常碰到的，天黑時電燈要自動亮起。

17) 同步（Parallelization）：把可以同時執行之狀況，同步進行。

18) 模擬（Simulation）：模擬實際情形。

以資訊科學來說，常利用程式語言教導學生如何寫程式，透過程式邏輯的培養，建構學生對於運算思維的概念。而研究也指出大學生透過學習程式邏輯對運算思維有幫助，且理學院和科技與工程學院在「重複結

構」、「列表應用」及「列表綜合應用」方面表現明顯優於教育學院與文學院（李恩萱，2018）。

2.2. 積木式程式語言

學者 Lee 等人認為藉由程式設計的學習，有助於培養學生在模式化、設計與開發的運算思維能力（Lee *et al.*, 2011）。而隨著資訊科技的不斷演進，程式語言的類型相當多元，對程式初學者來說，要先弄清楚程式邏輯觀念，也要知道每項功能指令，才能夠順利寫出完整的程式碼。舉例來說，許多資訊相關科系，多以 C 或 Java 語言為入門程式語言，但學生在學習時常常會花較多時間在程式語言的細節與除錯上，進而影響學生的學習成效（Johnson, 1995）。因此，許多學者專家認為不應再著重程式語言的語法、結構與設計技巧，並建議應採用視覺化的程式設計學習工具，讓初學者降低學習過多低階技巧的負擔，進而著重在學習高階的思維能力上（林育慈和吳正己，2016），同時，強調從動手實作的具體經驗中學習拆解、模式識別、抽象化及演算法則的邏輯思考與問題解決能力（Grover, & Pea, 2013；Google, 2019；林育慈和吳正己，2016）。

而積木式程式語言一般是指讓操作者使用圖形化元素進行程式設計的直覺式語言（Haeberli, 1988），具備易於操作及迅速獲得回饋等優點，且可降低初學程式設計語法指令的枯燥與挫折，適合作為學習運算思維的工具（徐宏義和羅曼如，2016）。常見的 Scratch、App Inventor、及 Blockly 等均屬於積木式語言類型；其中，積木式程式 Blockly 是 Google 與 MIT Media Lab 為了降低程式初學者的學習門檻所設計，透過不同顏色區分不同類型的程式積木，包含動作（Motion）、表達（Looks）、聲音（Sound）、事件（Events）、控制（Control）、觀感（Sensing）、運算（Operators）、及變數（Variables），利用圖像化的方式呈現，讓初學者可以快速理解各個程式積木的功能，並專注於學習運算思維。

2.3. 專題導向學習

專題導向學習（Project-based learning）是建構取向的學習方法，融合了建構主義、合作學習與情境學習之基礎，強調教師為學習輔助角色，透過安排學習任務，讓學生能夠從中主動學習，在學習任務中建構自己的知識，並有助於提升學生的自我調節能力（Lin & Tsai, 2016）。學習者可決定如何解決問題與需要的活動，進而蒐集多元資料整合、分析和擷取知識，並產出作品。而以解決問題為目標的教學模式可分為問題導向與專題導向，皆強調在真實世界的環境中學習，以學生成效為依歸，其中，專題導向學習是屬於行動傾向的，學生需主動設計與自己興趣相符的作品、並持續增進自己的表現（張玟慧、吳佳娣、陳彤宣，2016），Bouhuijs、Perrenet 與 Smits（2000）則認為專題導向學習比問題導向學習更需要知識的「應用」。

2.4. 虛擬實境

虛擬實境（Virtual Reality, VR）是指利用電腦科技產生一個三維空間的虛擬世界，並透過影像、聲音、週邊設備等讓使用者有身歷其境的感受，具有沈浸性（Immerse）、互動性（Interactive）、以及想像性（Imagination）；沈浸性會讓使用者沈浸於虛擬的世界，與真實世界脫離；互動性則是在虛擬世界碰觸物體時會得到與真實世界相同之反應；想像性，使用者會用在真實世界的感受，想像出虛擬世界的種種事件，因此充滿了想像性（Krueger, 1991）。虛擬實境技術常應用於飛行員訓練、醫療手術模擬、科學實驗、立體觀念等，在飛行員訓練、科學實驗上不但降低了危險性，且可以多次的重複操作，進而降低了訓練成本；而在娛樂方面也有諸多運用，包含遊戲、演唱會、電影、虛擬博物館等，其中數位遊戲為最主要項目，在其他方面，搭配虛擬實境也帶給使用者不同的感受。

3. 研究方法與設計

本研究將以設計虛擬實境旅遊導覽為主題的專題導向學習方式，進行程式設計教學之研究，透過實作的方式評量學習，學生也能依據本身的經驗來設計有興趣的主題，並探究其影響。實驗對象為臺灣北部某大學學生，在實驗過程中進行教學記錄，以進行結果分析與討論。

3.1. 主題內容設計-旅遊導覽

研究指出大學生有意圖透過科技設備進行觀光（Chou & Lin, 2015），搭配行動科技設備有畫龍點睛的作用，近似於傳統導遊在進行導覽時手上拿的圖片、字卡等，在行動科技設備上也可放進動畫，遊客可以透過動畫加速了解導遊想要表達的重點。而運用虛擬實境與網路技術，更能連接至任何觀光景點（吳世光和陳建和，2002），Google 則以「將全世界帶到使用者的眼前」為主軸（Google developers, 2019），在 2007 年啟用了 Google 街景服務的功能，透過此服務民眾可以走訪地球上著名地標，以及世界奇觀，還可以進入博物館、運動場、小商家等查看內部實景，透過 Google 提供的各項服務，遊客做旅遊規劃時可以先查看環景照片觀察附近環境，再決定是否安排進行程當中。

由此可知，旅遊導覽是生活化的實用主題，其設計技巧包含動線規劃、道具選用、導覽內容之選用、互動方式等，例如動線規劃從選定旅遊地區開始，根據遊客的背景，例如：學歷、年齡、興趣等，作為導覽重要的參考依據，導遊在某些特定景點進行解說時必須在「特定點位」才能看見「特殊樣貌」，並且進行動線安排（施鍾武，2017）。因此，旅遊導覽設計起點便是從資料蒐集開始。

3.2. 程式設計工具

目前虛擬實境產品蓬勃發展，除了有價格昂貴的 HTC vive 外，也有輕巧、便宜的 Cardboard，且可透過自行開發的方式設計出屬於自己的虛擬實境場景。因此，本研究將以運算思維教學概念為基礎設計教材，教學工具使用由 Delightex 開發的 CoSpaces Edu 虛擬實境教育平台，此平台結合有趣的樂高積木色塊與 CoBlocks，作為適合程式設計初學者的程式編譯器，搭配 Cardboard 即可進入虛擬實境的世界，依照程式設計的

不同，可以製作出動畫且具有故事性、互動性豐富的場景畫面，並探討應用 CoSpaces 之運算思維教材實施程式設計教學後的學習成效。

3.3. 運算思維教學內容設計

本研究從旅遊導覽規劃的角度切入，並依據 Barr 與 Stephenson 提出的運算思維元素，對照到旅遊導覽規劃設計，如表 1 所示。將旅遊導覽規劃拆解為搜集資訊、動線規劃、道具選用、導覽內容之適用、情境包裝、撰寫講稿、實際與遊客互動；程式設計活動中含：搜集素材、製作場景、定義各場景之功能、編寫腳本、編寫程式、物件同步執行、使用 Cardboard 觀賞場景等。

表 1 運算思維元素對照表（本研究整理）

運算思維元素	旅遊導覽規劃	程式設計活動
資料蒐集	搜集資訊	搜集素材
資料分析	動線規劃	製作場景
資料表示	道具選用	定義各場景之功能
問題解析	導覽內容之適用	編寫腳本
演算法思維	情境包裝	
抽象化	撰寫講稿	編寫程式
自動化		
並行		物件同步執行
模擬	實際與遊客互動	使用 Cardboard 觀賞場景

如上表 1 所示，舉例來說，搜集素材對應至資料蒐集，使用者須先思考想要製作成虛擬實境的導覽場景與內容，並蒐集相關影音素材，而編寫腳本則對應至問題解析，可在設計導覽的過程中加入問題詢問，達到互動的效果，亦可透過故事包裝，提升使用者在場景中的沉浸感，編寫程式則對應至抽象化，透過條件敘述、迴圈等概念，進行程式設計，並運用 Cardboard 觀賞場景對應至模擬，讓使用者進入自己設計的導覽場景，帶來成就感。

3.4. 教學活動設計與流程

本研究設計之運算思維教材內容使用 CoSpaces 之圖形化積木作為程式設計教學工具，共設計五個單元，包含搜集素材、製作場景、編寫腳本、編寫程式、觀賞場景等，讓學生在操作過程中，培養運算思維概念，如圖 1 所示。

圖 1 教學活動設計與流程

4. 研究結果與討論

本研究讓使用者思考生活中有哪些適合進行旅遊導覽介紹之景點，並介紹 CoSpaces 平台，讓使用者知道有哪些素材可以放進場景中，並透過實地探訪之方式搜集素材，如：360 照片、錄音檔等，並依檔案類型不同加以分類整理，明確定義出場景的介紹動線。在製作場景時，首先進行場景建置的教學，受試者可以使用 3D 環境或是 360 影像的方式創建場景；並將人物、動物、物件等放進場景中，完成製作場景的步驟，如圖 2 所示。

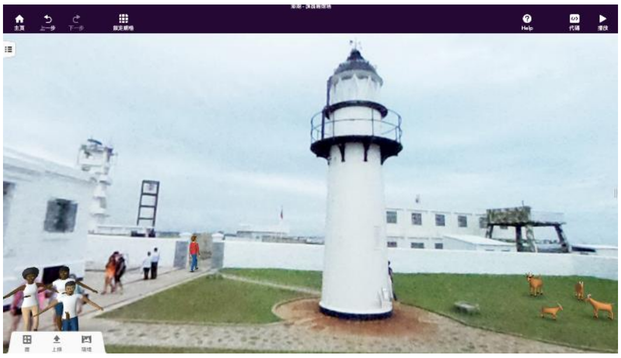


圖 2 製作場景畫面

接著受試者可以設計相關腳本，如：設計問答之方式或是用故事的方式描述想要講解之內容，以利後續進行程式設計時更為順利。為了讓場景看起來更豐富，受試者會使用到轉變、動作、事件、控制、項目、資料、函數與物理等程式積木功能，進行物件移動、觸發產生事件、講解介紹詞等與觀看者互動；也會讓使用者透過同步之積木，讓同一場景裡的多個物件一起動作，如圖 3 所示為編寫程式之畫面。

圖 3 編寫程式畫面

最後讓使用者將其產出之成果作品，透過播放場景並搭配 Cardboard 觀賞運行畫面，運用虛擬實境觀賞方式呈現其執行畫面，讓使用者有身歷其境的感受，如圖 4 所示為程式運行畫面。圖 5 則為使用者操作過程。

圖 4 程式運行畫面

圖 5 使用者操作過程

5. 結論與建議

研究指出透過專題導向的方式學習對學生會有正向影響（張玟慧等人，2016），而本研究結合運算思維進行專題導向程式設計教學，在課程設計方面，學生透過老師的引導將製作專題之流程進行拆解，讓學生對於專題製作能有較清楚的架構，並且能夠透過此流程學習到程式設計邏輯。另一方面，本研究運用積木程式語言的方式進行教學，讓學生能夠在短時間內製作出自己預想的內容，但在除錯方面，則因受試者為初學者，有時會遇到「程式執行後卻沒有看見預期結果」的問題，此時，則適時加以引導尋找解決的方法，而學生也會因為製作出成果後獲得成就感，並與同儕進行分享。而從教學記錄中也可發現，受試者對於專題設計很感興趣，也會進行自主性思考，製作互動遊戲等方式，在最後階段觀賞場景時學生反應：

頭好暈；看這些會動的人覺得很好笑；可以跟裡面的人物互動很有趣；之前製作時是在電腦上看；但最後用 Cardboard 看時有更多的震撼感；從搜集資料開始就很自由，老師不會侷限我們要用哪種方式呈現成品，也會給我們方向。

由此可見，本研究基於運算思維之教學設計方式，可以增進學生程式設計自主學習的能力與提升學生成就感。

6. 參考文獻

林育慈和吳正己（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**，5-20。

吳世光和陳建和（2002）。影像式虛擬實境之發展及其在觀光產業應用之研究。**觀光研究學報**，**8**（1），109-125。

李恩萱（2018）。**大學生運算思維與程式設計學習成就研究**（未出版之碩士論文）。臺北市：國立臺灣師範大學。

張玟慧、吳佳娣和陳彤宣（2016）。專題式程式設計教學對國小學童問題解決歷程之研究。**教育科技與學習**，**4**（2），137-162。

施鍾武（2017）。**旅遊導覽技巧之探討**，2017 年 2 月 18 日，取自 http://www1.tourguide.org.tw/UserFiles/2017218105233582.pdf。

Barr, V., & Stephenson, C. (2011). *Bringing Computational Thinking to K-12. ACM Inroads, 2*(1), 48. doi:10.1145/1929887.1929905

Bouhuijs, P. A. J., Perrenet, J. C., & Smits, J. G. M. M. (2000). The Suitability of Problem Based Learning for Engineering Education: Theory And Practice. *Teaching in Higher Education, 5*(3), 345-358.

Chou, R. T. C., & Lin, K. (2015). A Model of Taiwanese College Student Tourists' Intention to Use Mobile Technology. *Journal of Tourism and Leisure Studies, 21*(1), 79-102. doi:10.6267/JTLS.2015.21(1)4

CSTA. (2011). *CSTA K-12 Computer Science Standards. The ACM K-12 Education Task Force*. Retrieved January 18, 2019, from https://cdn.ymaws.com/www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf

Fernaeus, Y., Kindborg, M., & Scholz, R. (2006). Programming and Tools: Rethinking Children's Programming with Contextual Signs. *Proceedings of Conference on Interaction Design and Children IDC 06*. ACM, 121-128.

Ferlazzo, L. (2015). *Building a Community of Self-motivated Learners: Strategies to Help Students Thrive in School* and *Beyond*. UK: Routledge.

Google (2019). *Exploring Computational Thinking*. Retrieved January 18, 2019, from https://edu.google.com/resources/programs/exploring-computational-thinking/

Google Developer. (2019). *Street view*. Retrieved January 18, 2019, from https://developers.google.com/streetview/?hl=zh-tw

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

Haeberli, P. E. (1988). ConMan: A Visual Programming Language for Interactive Graphics. *ACM SigGraph Computer Graphics, 22*(4), 103-111.

Jane, I. M., MEDLOCK-WALTON, P., & TISSENBAUM, M. (2017). App Inventor VR Editor for Computational Thinking. *Proceedings of International Conference on Computational Thinking Education 2017*. The Education University of Hong Kong, 160-163.

Johnson, L. F. (1995). C in the First Course Considered Harmful. *Communications of the ACM*, *38*(5), 99-101.

Krueger, M. (1991). *Artificial Reality II.* US: Addison-Wesley.

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

Lin, J. W., & Tsai, C. W. (2016). The Impact of an Online Project-based Learning Environment with Group Awareness Support on Students with Different Self-regulation Levels: An Extended-period Experiment. *Computers & Education*, *99*, 28-38.

Schoolnet. (2015). *Computing Our Future: Computer Programming and Coding Priorities, School Curricula and Initiatives Across Europe.* Retrieved October 1, 2015, from http://fcl.eun.org/documents/10180/14689/Computing+our+future_final.pdf/746e36b1-e1a6-4bf1-8105-ea27c0d2bbe0

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

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Flipped Learning Approach for Coding Education in Higher Education

Hui-chun HUNG
Taipei Medical University, Taiwan
hch@tmu.edu.tw

ABSTRACT

Programming skills are seen as an essential competency in the 21st century. In response to the new curriculum for Basic Education announced by the Ministry of Education of Taiwan, most universities in Taiwan had announced the ability of programming as the key indicators. The traditional programming courses emphasize coding statement and algorithmic theory might. It is a high-level entry for students with non-information backgrounds, and it is easy for students to lack motivation to learn. Therefore, this exploratory work-in-progress study attempts to present an undergraduate programming course to conduct empirical evidence-based research at universities. Therefore, the improvement of programming teaching is an urgent task. To enhance the programming learning outcomes and motivation of university students, this project conduct empirical teaching experimental research in universities, from the integration of flip classrooms, combined with the development of live-coding and code annotations materials. A compulsory course, "Introduction to Python Programming", has been designed for the general programming skills of the school. It is expected to be a valuable reference for programming innovative teaching.

KEYWORDS

programming, python, SPOCs, higher education, flipped classroom

1. INTRODUCTION

Programming has brought a wave of learning around the world. More and more learners from different backgrounds have joined the ranks of learning programming, starting with a number of online courses. In higher education, the information literacy of students has become one of the key abilities. In response to the new curriculum for Basic Education announced by the Ministry of Education of Taiwan, most universities in Taiwan had announced the ability of programming as the key indicators. Programming is an important tool to carry out computational thinking which can help students to critically think and solve problems.

Nowadays, programming is no longer a professional skill that only the specific information engineers have. Programming skills have been regarded as one of the basic core competencies, which can affect the national economy and competitiveness. It is regarded as an essential ability in the 21st century. Higher education also emphasizes that students have a broad knowledge base, logical thinking ability, and critical thinking through general education. The Ministry of Education is scheduled to implement the new 12-year National Curriculum in the 2008 academic year. It also lists information technology and programming as a compulsory curriculum for national and high school students (National Institute of Education, 2016). Therefore, it is imperative to enhance the programming ability of university students.

However, traditional programming learning emphasizes the theory of programming language structure, algorithm. It is a high-level entry for students with the non-computer science background, and it is easy to lead students to lack motivation for practical programming, problem-solving and software development capabilities. Therefore, the improvement of programming learning is an urgent task.

The rise of information technology has spurred traditional higher education toward the reform of technology-assisted learning. With the popularity and flourishing of open educational resources, Massive Open Online Courses (MOOCs) are seen as a revolution in global higher education and scientific literature (Lépez-Meneses, Vázquez-Cano, & Román, 2015), which aims to provide students with an open and flexible learning pipeline. MOOCs allowing students to master self-directed learning according to their learning pace, conditions, and characteristics. Fox (2013) proposed Small Private Online Courses (SPOCs) to use MOOCs' high-quality teaching videos to match the teaching design and classroom activities of school instructors to improve the effectiveness of teaching and learning. At the same time, it also uses large-scale data analysis to improve classroom effectiveness. SPOCs is designed to combine MOOCs' courses and increase the lecturer's influence, students' productivity and engagement.

To enhance the programming learning outcomes and motivation of university students, Therefore, this exploratory work-in-progress study try to conduct empirical teaching experimental research in universities, from the integration of flip classrooms, combined with the development of live-coding and code annotations materials. A compulsory course, "Introduction to Python Programming", has been designed for the general programming skills of the school. It is expected to be a valuable reference for programming innovative teaching.

2. LITERATURE REVIEW

Programming is a tool aims to solve the problem in real life. Traditional assessments are not easily adaptable to new developments in computer programming education. New learning methods, such as collaborative learning, project-based learning (PBL), e-learning, and mobile learning, are working to explore new ways to enhance the programming learning outcomes. Various tools have been introduced in the education process to strengthen teaching and learning activities. These tools play an important role in enriching students' learning experiences (Rubin, 2013) These digital tools are essential in programming teaching and learning because programming software and environments are closely related to computers and require a computer as a platform for implementing and testing programming grammar (Syahanim, Mohd, & Salleha, 2013). The programming process involves a variety of activities, including planning, design, testing, and debugging. To learn how to develop a program, students need to understand the syntax of a programming language. The complexity of programming and the difficulty of understanding program logic often lead to frustration and lack of motivation in learning programming (Kelleher, 2005).

3. SETTING

In this study, we present an exploratory work-in-progress study to conduct empirical teaching experimental research in universities. This course plans to conduct empirical evidence-based research in a compulsory course, “Introduction to Python Programming” focus on the integration of flipped classroom, live streaming video and deep learning to explore the smart programming education in the era of big data. Based on the characteristics of SPOCs, this study develops a curriculum learning strategy combined with online digital courses, physical classroom courses, online programming annotations, and peer-to-peer evaluations.

The study has produced an online course for Python programming for beginners and has conducted a one-semester empirical experiment. There are 42 students participated in this course. Before the class, the learner logs in the learning management system to watch the course videos produced by this study, and then completes the pre-course unit exercises and tests, and returns the learning status to the teacher.

The course is conducted as follows: Before the class, the learner needs to watch the online course videos produced by this study. In the Face-to-Face classroom, students could run the program implementation and execution, debugging, etc. After the class, students share their own online programming annotations. After that, their annotations and codes are reviewed by peers. The four major mechanisms of flipping the classroom are shown in Figure 1.

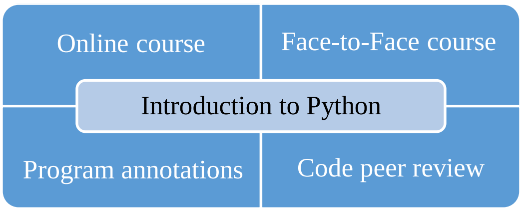


Figure 1. SPOCs theory applied to the programming course

4. RESULTS

In this study, the curriculum with the flipped learning for programming course for different background students was proposed. Moreover, a total of 18-week python programming online course has been designed based on the students of different backgrounds in the general education curriculum. The video material produced by the real teacher, the content is combined with the teacher's actual explanation, screen recording for demonstrations, and program source code.

In this study, the digital design curriculum is integrated into the digital learning management system with SPOCs teaching strategy. In the future, this study will focus on the learners' satisfaction and the learning outcomes of students. More details about students' learning outcomes and learning attitudes toward Python programming will be explored by formative assessment and questionnaires. It is expected to be a valuable reference for programming innovative teaching.

5. ACKNOWLEDGEMENTS

This research project is funded both by the 107 MOE Teaching Practice Research Program by Ministry of Education in Taiwan and Taipei Medical University, under the project codes: TMU105-AE1-B47.

6. REFERENCES

Salleh, S. M., Shukur, Z. and Judi, H. M. (2013). Analysis of Research in Programming Teaching Tools: An Initial Review. *Procedia-Social and Behavioral Sciences, 103*, 127-135.

Rubin, M. J. (2013). The Effectiveness of Live-coding to Teach Introductory Programming. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education.* ACM, 651. doi: 10.1145/2445196.2445388

Herala, A., Vanhala, E., Knutas, A., & Ikonen, J. (2015). Teaching Programming with Flipped Classroom Method: A Study from two Programming Courses. *15th Koli Calling Conference on Computing Education Research*, 165-166. doi.:10.1145/2828959.2828983

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Special Education Needs

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Integrating Computational Thinking and Mathematics for Children with Learning Disabilities with Google Blockly

Chen-huei LIAO[1*], Bor-chen KUO[2*], Kai-chih PAI[3*], Pei-chen WU[4*], Chih-wei YANG[5*]
[1] The Department of Special Education, National Taichung University of Education, Taiwan
[2345] Graduate Institute of Educational, National Taichung University of Education, Taiwan
chenhueiliao@gmail.com, kbc@mail.ntcu.edu.tw, cms100106@gm.ntcu.edu.tw, seashellpeach@gmail.com, yangcw@mail.ntcu.edu.tw

ABSTRACT
The aim of the study was to develop instructional materials that integrate the concept of computational thinking and mathematical thinking with Google Blockly for children with learning disabilities. There are totally eight units to introduce the concepts of orientation, direction, path and distance. Google Blockly was adopted to provide manipulatable visual stimuli which help children with learning disabilities to solve the problem through computational and mathematical thinking.

KEYWORDS
computational thinking, mathematics, learning disabilities, blockly

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019*. Hong Kong: The Education University of Hong Kong.

學習障礙學生運算思維與數學的 Google Blockly 教學設計

廖晨惠 [1*]，郭伯臣 [2*]，白鎧誌 [3*]，鄔珮甄 [4*]，楊智為 [5*]
[1] 國立臺中教育大學特殊教育學系，臺灣
[2345] 國立臺中教育大學教育資訊與測驗統計研究所，臺灣
chenhueiliao@gmail.com，kbc@mail.ntcu.edu.tw，cms100106@gm.ntcu.edu.tw，seashellpeach@gmail.com，yangcw@mail.ntcu.edu.tw

摘要

本研究已進行至第二年，第一年的教材內容根據明確教學原則（Explicit Instruction）設計，教材內容結合運算思維和數學思維概念，從發展活動到活動 7 共 8 個單元，透過教導學生認識方位、方向、路徑和距離等習得空間概念，教材的對象為學習障礙的學生，經過試教後，將教材進行調整，於第二年階段，利用 Google Blockly 開發為一教學系統，幫助學生直接利用程式積木，在解題的過程中，學習並透過運算思維中的概念和實踐解決問題。

關鍵字

運算思維；數學；學習障礙；Blockly

1. 前言

運算思維是近幾年來在教育界中相當重要的研究議題，其意旨運用電腦科學的基礎概念解決問題、設計系統和理解人類的行為（Wing, 2006），是一種問題解決的過程（Google, 2019；ISTE, 2011），所有人都應具備的基本能力。已經被廣泛應用在不同的學科，如數學（Jenkins et al., 2012；Snodgrass et al., 2016）、科學（Ahamed et al., 2010；Swaid, 2015）、生物學（Qin, 2009）、電腦科學（Repenning, 2012；Grover, Pea, & Cooper, 2015；Good, Yadav, & Mishra, 2017）、語言（Wolz et al., 2010, 2011；Howell et al., 2011）等。在這些結合運算思維進行教學的學科課程中，程式設計是常見的教學工具，當前的許多研究都顯示使用程式設計進行教學是具有相當教學成效的（Kazimoglu et al., 2011；Pellas & Peroutseas, 2017；Snodgrass et al., 2016；Ber et al., 2014）。

本研究將延續第一年的研究目的和結果，擬設計一套結合運算思維教學的數學活動教材，幫助學生透過其中的運算思維概念進行數學解題，此次的研究會將第一年所設計的不插電活動教材，使用視覺化程式設計工具（Visual Programming Language）開發為電腦化教材，讓學生透過程式設計的方式運用運算思維概念解決問題。

2. 視覺化程式設計工具（Visual Programming Language, VPL）

一般常見的程式設計工具為文字型程式設計工具（Text-based Programming Language），如 C, JAVA, VB, RUBY, PYTHON 等，透過文字敘述完成程式編寫，不同語言有不同的語法，但是程式邏輯大致相同，不過門檻較高，需要花較多時間學習，因程式設計是學習運算思維的一種模式、一種工具，當學生花太多的時間學習工具，反而會失去其意義。而視覺化程式設計工具指的是使用者可以透過已經圖形化的元件進行程式撰寫，使用上較傳統的文字型程式設計工具容易上手和直覺，目前應用在教育上的相關工具有 AgentCubes, Alice, Kudo, Google Blockly, Scratch, Code.org 等，其中 Scratch 就標榜低地板（容易入門），高天花板（可以製作出很複雜的作品）和牆面寬廣（每個學習者都可以根據自己的興趣創建不同性質的專案），其是一種積木式的視覺化程式設計工具（Block-based Programming），可以讓學生拖拉程式積木即可完成作品的一種工具，在 Snodgrass, Israel 和 Reese（2016）的研究中，他們利用 Scratch 做為教學工具來幫助特殊需求學生學習數學，學習如何計算時間，結果是有成效的。在本研究中，也嘗試使用這類型的工具（Block-Based Programming）做為教學工具，但為能配合教材設計，關卡由簡單到複雜，在程式積木的部分需依關卡內容設計有所限制，所以採用可以自定義積木的 Google Blockly 進行教材開發。

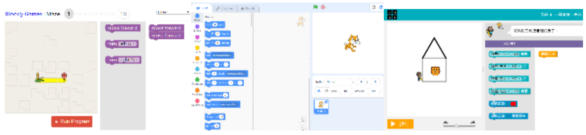


圖 1 Google Blockly, Scratch, Code.org 等視覺化程式設計工具

3. 教材開發

本研究的教材是針對學習障礙的學生所設計的，學習障礙有不同的類型，主要可以分成三類，分別是數學障礙、書寫障礙及閱讀障礙，本研究在教材設計時會考量三種不同障礙類別學生的特殊需求，並且搭配明確教學原則（Explicit Instruction）進行課程設計，Israel 等學者（2015）提到明確教學原則是一套有系統且直接的教學策略，並有研究顯示此教學模式在面對較複雜的任務，需要多步驟進行的運算任務時可以有效的幫助學習障礙的學生進行學習，例如儘量以圖示代替文字，如積木程式區塊以圖像化牌卡的樣式呈現，讓其概念具象化，減少學生認知負荷，以達到最適切的效果。

原先的教材從發展活動到活動 7 共有 8 個單元，從最基礎的方位概念建構到進階運用牌卡規劃最短路徑，希望可以讓學生藉這一系列活動認識方位、規劃路徑以建構空間概念。於此次的研究中，將不插電教材開發為電腦化版本，選擇 Google Blockly 進行課程開發，其

一原因是因為 Blockly 可以限制可使用的程式積木和數量與自定義積木，以利教材的規劃。

目前正在開發階段，預計將原本的 8 個活動，配合教材電腦化後所需要的調整，修訂為 10 至 15 關卡，從最基礎的方位移動到加入障礙物等條件規劃最近和最遠路線。但和原本不插電活動的差異之一，就是關卡一開始所使用的積木，直接以牌卡的介面呈現，學生藉由操作這些程式積木，過程中運用運算思維概念如指令、序列、迴圈、條件等架構程式積木，利用運算實踐如計畫與設計、抽象化與建模等進行解題，透過完成關卡活動的整個流程，運用且也習得運算思維這種問題解決模式，解決數學問題，在這整個活動流程中，學生所有的操作歷程皆會被記錄下來，結果將會進行分析，做為之後教材修訂的參考。

圖 2 Blockly 教材與牌卡介面

4. 未來研究工作

目前系統還在進行修正中，預計再加入回饋和提示的機制，在學習過程中可即時提供教學回饋，幫助學生解題。

5. 致謝

本研究由臺灣科技部計畫編號 MOST 106-2511-S-142 -004 -MY3 補助支持，特此誌謝。

6. 參考文獻

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). Computational Thinking for the Sciences: A Three Day Workshop for High School Science Teachers. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education.* ACM, 42-46.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational Thinking and Tinkering: Exploration of an Early Childhood Robotics Curriculum. *Computers & Education*, *72*, 145-157.

Good, J., Yadav, A., & Mishra, P. (2017). Computational Thinking in Computer Science Classrooms: Viewpoints from CS Educators. *Proceedings of Society for Information Technology & Teacher Education International Conference*. Association for the Advancement of Computing in Education (AACE), 51-59.

Google (2019). *Exploring Computational Thinking.* Retrieved January 4, 2019, from https://edu.google.com/resources/programs/exploring-computational-thinking/

Grover, S., Pea, R., & Cooper, S. (2015). Designing for Deeper Learning in a Blended Computer Science Course for Middle School Students. *Computer Science Education*, *25*(2), 199–237.

Howell, L., Jamba, L., Kimball, A. S., & Sanchez-Ruiz, A. (March). Computational Thinking: Modeling Applied to the Teaching and Learning of English. *Proceedings of the 49th Annual Southeast Regional Conference*. ACM, 48-53.

ISTE (2011). *Operational Definition of Computational Thinking for K–12 Education*. Retrieved January 4, 2019, from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S., & Tapia, T. (2015). Empowering K–12 Students with Disabilities to Learn Computational Thinking and Computer Programming. *TEACHING Exceptional Children*, *48*(1), 45-53.

Jenkins, J. T., Jerkins, J. A., & Stenger, C. L. (2012). A Plan for Immediate Immersion of Computational Thinking into the High School Math Classroom through a Partnership with the Alabama Math, Science, and Technology Initiative. *Proceedings of the 50th Annual Southeast Regional Conference*. ACM, 148-152.

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2011). Understanding Computational Thinking Before Programming: Developing Guidelines for the Design of Games to Learn Introductory Programming through Game-play. *International Journal of Game-Based Learning (IJGBL)*, *1*(3), 30-52.

Pellas, N., & Peroutseas, E. (2017). Leveraging Scratch4SL and Second Life to Motivate High School Students' Participation in Introductory Programming Courses: Findings from a Case Study. *New Review of Hypermedia and Multimedia*, *23*(1), 51-79.

Qin, H. (2009). Teaching Computational Thinking through Bioinformatics to Biology Students. *ACM SIGCSE Bulletin*, *41*(1), 188-191.

Repenning, A. (2012). Programming Goes Back to School. *Communications of the ACM*, *55*(5), 38-40.

Snodgrass, M. R., Israel, M., & Reese, G. C. (2016). Instructional Supports for Students with Disabilities in K-5 Computing: Findings from a Cross-case Analysis. *Computers & Education*, *100*, 1-17.

Swaid, S. I. (2015). Bringing Computational Thinking to STEM Education. *Procedia Manufacturing*, *3*, 3657-3662.

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

Wolz, U., Stone, M., Pulimood, S. M., & Pearson, K. (2010). Computational Thinking via Interactive Journalism in Middle School. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education.* ACM, 239-243.

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.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Evaluation

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Measurement of Computational Thinking Performance Using Multiple-choice Questions

Yerkhan MINDETBAY[1*], Christian BOKHOVE[2], John WOOLLARD[3]
[123] University of Southampton, The United Kingdom
y.a.mindetbay@soton.ac.uk, c.bokhove@soton.ac.uk, j.woollard@soton.ac.uk

ABSTRACT
This study investigates the measurement of computational thinking performance of secondary school students using multiple-choice questions. The sample group of 775 grade eight students are drawn from 28 secondary schools across Kazakhstan. Students responded to a Computational Thinking Performance test of 50 multiple-choice questions. The test covers the concepts: logical thinking, generalisation and abstraction. The validity and reliability of the multiple-choice questions are determined using an Item Response Theory model. The item difficulty and discrimination coefficients are calculated, and the item characteristic curves for each question and test information functions for each quiz are generated. The results of the study show that the multiple-choice question assessment is a valid and reliable tool to measure computational thinking performance of students.

KEYWORDS
computational thinking, measurement, evaluation, multiple-choice questions, item response theory

1. INTRODUCTION
As computational thinking is becoming more popular trend in education, many countries integrated it into their national curricula. The most common way of delivering computational thinking in schools is through teaching computer programming**,** in some cases applying the pair programming technique and using unplugged activities (Bell, Witten, & Fellows, 2015) to teach computer science concepts in classrooms. The increased use of educational robots and programmable kits is also spreading the teaching of computational thinking. However, teaching methods are still in the early stage of development. The evaluation of computational thinking is as important as its integration into curricula, as without clear and verified assessment, attempts to integrate computational thinking into any curriculum cannot be verified. Moreover, in order to judge the effectiveness of computational thinking teaching strategies, measures must be approved that would allow teachers to assess what children learn (Grover & Pea, 2013). There is a need for standardized tests that can assess whether students can think computationally (Linn et al., 2010). The aim of this research is to establish a valid measurement of computational thinking performance of students by using multiple-choice questions.

1.1. Computational Thinking and Evaluation
Thinking is a mental process with a high-order cognitive function used in the process of making choices and judgments (Athreya & Mouza, 2017). The thinking process consists of lower-order and higher-order sub-processes, where a higher-order thinking is related to problem-solving, critical thinking, creative thinking, and decision-making. Computational thinking is a cognitive process, which reflects the ability to think in abstractions, algorithmically and in terms of decomposition, generalisation and evaluation (Selby, 2014, p.38). Computational thinking is related to spatial ability (Ham, 2018), academic success (Ambrosio, Almeida, Franco, & Macedo, 2014; Durak & Saritepeci, 2017; Gouws, Bradshaw, & Wentworth, 2013) and problem-solving ability (Román-González, Pérez-González, & Jiménez-Fernández, 2016).

2. METHODOLOGY
A bespoke computational thinking assessment was designed because most of the assessment tools for computational thinking are based on particular programming languages (Jamil, 2017) or some specific tools (Moreno-Leon & Robles, 2015; Oluk & Korkmaz, 2016; Seiter, 2015; Weese, 2016; Zhong, Wang, Chen, & Li, 2015). Context-specific evaluations of computational thinking might be biased due to students' prior knowledge and experience in those particular programming languages or tools. In this study, the test is more neutral as it is not a language-specific measurement. The national curricula of the Kazakhstani schools, the annual plans of "Bilim Innovation" Lyceums and students' problem-solving experience have been explored in order to construct test questions. The multiple-choice test is written taking into consideration the national curriculum, annual plans for informatics and Informatics textbooks (Shaniyev et al. 2017) and students' experience with problem solving.

2.1. Multiple-choice questions
As a frequently used assessment type in school, multiple-choice questions (MCQ) have several advantages including: efficiency for large-scale studies (Becker & Johnston, 1999; Dufresne, Leonard, & Gerace, 2002; Roberts, 2006); accuracy (Holder & Mills, 2001); objectivity (Becker & Johnston, 1999; Haladyna & Steven, 1989; Simkin & Kuechler, 2005; Zeidner, 1987); and compatibility with classical and item response theories (Haladyna & Steven, 1989). Multiple-choice questions are the most suitable format for assessment of higher-order cognitive skills and abilities (Downing & Haladyna, 2006), such as problem-solving, synthesis, and evaluation; and they are more effective on improving learning (Haynie, 1994; Smith & Karpicke, 2014). The multiple-choice questions for this study have been carefully constructed in line with the

context relevant recommendations on writing good multiple-choice items provided by the authors Downing & Haladyna (2006), Frey et al. (2005), Gierl et al. (2017) and Reynolds et al. (2009). In addition, two experts with experience in assessing computational thinking reviewed these test questions. Each item in this multiple-choice test has four response options, with one correct answer and three distractors. There are 50-multiple-choice questions (set of 5 quizzes with 10 questions each) in this test with maximum score of 50. It is conducted online with a duration of 100 minutes.

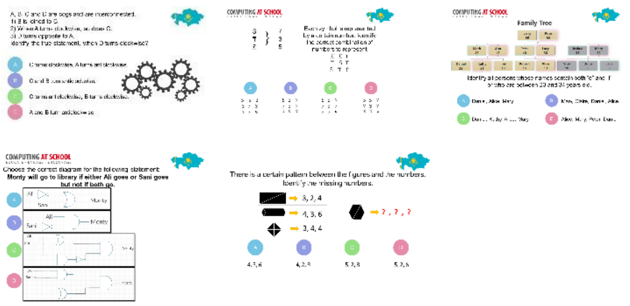


Figure 1. Sample questions.

Note: The sample questions can be accessed through: http://bit.ly/SampleQs

2.2. Item Response Theory

Item Response Theory (IRT) is a paradigm for the design analysis and scoring of test instruments that measures attitudes, abilities and other variables. This theory is based on the relationship between person's performance on a test item and the person's performance level on an overall measure of the ability the item was constructed to measure. IRT is based on a mathematical model, which describes in probabilistic terms, how a test taking person with a higher standing on a trait is likely to respond in a different response category to a person with a low standing on the trait (Ostini & Nering, 2006). IRT has several advantages over traditional test theory, such as, sample independency, measurement of range of different abilities, accounting item difficulty, accounting for guessing, and supporting adaptive testing (Thissen & Wainer, 2001).

3. DATA ANALYSIS

The responses for multiple-choice questions were converted into dichotomous items, 0s for wrong responses and 1s for correct responses. These data from 775 13-14 year old participants are tested according to 2-parameter and 3-parameter IRT models. These data are collected from 775 (549 boys, 226 girls) 8th grade students aged 13-14 years from Kazakhstan. The relationship between the probability of correct response to an item and the ability scale is described by the item characteristic curve (Baker & Kim, 2017). The item difficulty is a location index that shows where the item is located along the ability scale. An easy item functions among the low-ability students, a hard item functions among the high-ability students. The discrimination of an item, tells how well an item can differentiate between students with the abilities below the item location and those with the abilities above the item location. The item discrimination shows the steepness of the item characteristic curve in its middle section of the plot. The steeper the curve the better the item can discriminate; the flatter the curve the less the item can discriminate (Baker & Kim, 2017). The item discrimination parameter is "a". The item difficulty parameter is "b". The guessing parameter is "c". A 2-parameter IRT model suits better in this study, as the guessing parameter "c" is found as non-significant in 3-parameter model. The coefficients of item difficulty and item discrimination are presented in tables 1 and 2. The item characteristic curve plots are presented for each quiz in figures 2-6. The Cronbach Alpha is calculated for the items based on the responses from the sample size of 775. For the IRT analysis, the "mirt" and "ltm" libraries were used in RStudio.

4. RESULTS

The difficulty coefficients of majority of items are between the range of -0.8 and 1.3. All item characteristic curves for items fit well except for three items, item1(X1-black curve in Figure 4) and item6(X6-pink curve in Figure 4) in Abstraction quiz (Q3), item7(X7-yellow curve in Figure 5) in Pattern Numbers (Q4) quiz with the difficulty coefficients of 3.0, 1.8 and 2.0 respectively as shown in Table 1. The Cronbach Alpha (Field, 2013) coefficient for all 50 items is 0.87 (>0.7).

Table 1. The coefficients of item difficulty.

	Q1	Q2	Q3	Q4	Q5
Item1	-0.5	-0.8	**3.0**	0.9	-0.4
Item2	-0.2	-0.4	-0.8	0.1	0.2
Item3	0.2	0.1	-0.6	0.2	0.5
Item4	-0.3	-0.1	-0.1	0.0	0.4
Item5	0.2	-0.4	0.3	0.1	-0.1
Item6	0.0	0.0	**1.8**	-0.4	0.4
Item7	-0.2	1.0	-0.2	**2.0**	0.4
Item8	0.6	0.0	0.3	-0.1	0.4
Item9	1.3	0.2	0.5	0.2	1.1
Item10	1.2	0.5	0.2	0.0	0.3

The discrimination coefficients are between the range of 0.3 and 2.3 as shown in Table 2. The test information functions for each quiz show that the average ability respondent is tested the best.

Table 2. The coefficients of item discrimination.

	Q1	Q2	Q3	Q4	Q5
Item1	0.9	1.1	0.3	1.2	0.8
Item2	1.3	1.2	1.0	1.6	0.9
Item3	1.2	1.6	1.5	2.1	1.1
Item4	1.8	1.9	1.4	1.9	1.1

Item5	1.2	2.1	1.0	2.3	1.2
Item6	1.9	1.5	0.6	1.7	1.2
Item7	2.3	1.4	1.6	0.6	1.5
Item8	1.4	1.9	1.4	1.4	1.3
Item9	0.8	1.7	1.0	2.2	1.4
Item10	1.1	1.3	1.2	1.6	1.1

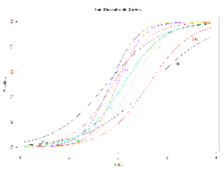


Figure 2. Logic quiz (Q1). http://bit.ly/Q1Logic

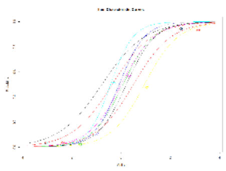


Figure 3. Logic quiz (Q2). http://bit.ly/Q2Logic

Figure 2 shows the individual item characteristic curves for the 10 items of the Logic Narrative quiz (Q1). Figure 3 shows the individual item characteristic curves for the 10 item of the Logic quiz (Q2). All lines ascend steeply showing a good discrimination coefficient.

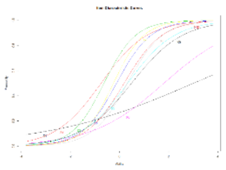


Figure 4. Abstraction quiz (Q3). http://bit.ly/Q3Abstraction

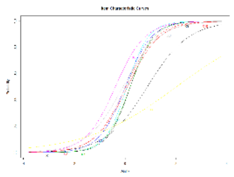


Figure 5. Pattern quiz (Q4). http://bit.ly/Q4Pattern

Figure 4 shows the individual item characteristic curves for the 10 items of the Abstraction quiz (Q3). The 2 outlier questions (items 1 and 6) being clearly identified by their more horizontal nature. Figure 5 shows the individual item characteristic curves for the 10 items of the Pattern quiz (Q4). The more difficult item being clearly identified by its displacement.

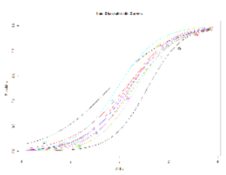


Figure 6. Pattern Figures quiz (Q5). http://bit.ly/Q5Pattern

5. DISCUSSION AND CONCLUSION

In this research, to measure students' computational thinking performance, 50 multiple-choice questions were specially designed with the focus on the concepts: logics, abstraction and generalisation. For the validity and reliability of the measurement 2-parameter IRT model and Cronbach Alpha test were used. Out of 50 items, 3 items were outliners as they were found difficult and were less informative in measurement. No too easy items were found. Test information functions for each quiz show that the most information is obtained for the average ability. The Cronbach Alpha result, the item difficulty and discrimination coefficients, the test information functions and the item characteristic curves are indications to justify the establishment of the validity and reliability of the multiple-choice questions to measure computational thinking performance of students.

6. REFERENCES

Ambrosio, A. P., Almeida, L. da S., Franco, A., & Macedo, J. (2014). Exploring Core Cognitive Skills of Computational Thinking. *Proceedings of PPIG 2014 - 25th Annual Workshop*. Sussex: University of Sussex.

Athreya, B. H., & Mouza, C. (2017). *Thinking Skills for the Digital Generation: The Development of Thinking and Learning in the Age of Information*. Cham: Springer International Publishing.

Baker, F. B., & Kim, S. H. (2017). *The Basics of Item Response Theory Using R*. Cham: Springer International Publishing.

Becker, W. E., & Johnston, C. G. (1999). The Relationship between Multiple Choice and Essay Response Questions in Assessing Economics Understanding. *Economic Record*, *75*(231), 348-357.

Bell, T., Witten, I. H., & Fellows, M. (2015). *CS Unplugged*. *Computer Science Unplugged*. Retrieved March 5, 2017, from https://www.csunplugged.org.

Downing, S. M., & Haladyna, T. M. (2006). *Handbook of Test Development* (1st ed.). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Dufresne, R. J., Leonard, W. J., & Gerace, W. J. (2002). Marking Sense of Students' Answers to Multiple-choice Questions. *The Physics Teacher*, *40*(3), 174-180.

Durak, H. Y., & Saritepeci, M. (2017). Analysis of the Relation between Computational Thinking Skills and Various Variables with the Structural Equation Model. *Computers & Education*, *116*, 191-202.

Field, A. (2013). *Discovering Statistics Using IBM SPSS Statistics (3rd ed.).* London: SAGE Publications Ltd.

Frey, B. B., Petersen, S., Edwards, L. M., Pedrotti, J. T., & Peyton, V. (2005). Item-writing Rules: Collective Wisdom. *Teaching and Teacher Education*, *21*(4), 357-364.

Gierl, M. J., Bulut, O., Guo, Q., & Zhang, X. (2017). Developing, Analyzing, and Using Distractors for Multiple-choice Tests in Education: A Comprehensive Review. *Review of Educational Research*, *87*(6), 1082-1116.

Gouws, L., Bradshaw, K., & Wentworth, P. (2013). First Year Student Performance in a Test for Computational Thinking. *Proceedings of SAICSIT '13*. ACM, 271-277.

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

Haladyna, T. M., & Steven, M. (1989). Validity of a Taxonomy of Multiple-choice Item-writing Rules.

Applied Measurement in Education, *7347*(January 2015), 37-41.

Ham, D. A. (2018). Spatial Thinking as a Path Towards Computational Thinking. In *Computational Thinking in Primary Education*, 103-122. Hershey: IGI Global.

Haynie, W. (1994). Effects of Multiple-choice and Short-answer Tests on Delayed Retention Learning. *Journal of Technology Education*, *6*(1), 1-7.

Holder, W. W., & Mills, C. N. (2001). Pencils Down, Computer Up: The New CPA Exam. *Journal of Accountancy*, *191*(3), 57-60.

Jamil, H. M. (2017). Smart Assessment of and Tutoring for Computational Thinking MOOC Assignments using MindReader. Retrieved October 17, 2017, from http://arxiv.org/abs/1705.00959

National Research Council (2010). *Report of a Workshop on The Scope and Nature of Computational Thinking*. National Academies Press.

Moreno-Leon, J., & Robles, G. (2015). Dr. Scratch: A Web Tool to Automatically Evaluate Scratch Projects. *Proceedings of the Workshop in Primary and Secondary Computing Education.* ACM, 132-133.

Oluk, A., & Korkmaz, Ö. (2016). Comparing Students' Scratch Skills with Their Computational Thinking Skills in Terms of Different Variables. *International Journal of Modern Education and Computer Science*, *8*(11), 1-7.

Ostini, R., & Nering, M. (2006). *Polytomous Item Response Theory Models*. California: SAGE Publications Ltd.

Reynolds, C. R., Livingston, R. B., & Willson, V. (2009). *Measurement and Assessment in Education (2nd ed.)*. New Jersey: Pearson.

Roberts, T. S. (2006). The Use of Multiple Choice Tests for Formative and Summative Assessment. *Proceedings of Conferences in Research and Practice in Information Technology Series*. Australian Computer Society, Inc., 175-180.

Román-González, M., Pérez-González, J.C., & Jiménez-Fernández, C. (2016). Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test. *Computers in Human Behavior*, *72,* 678-691.

Seiter, L. (2015). Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 540-545.

Selby, C. (2014). *How can the Teaching of Programming be Used to Enhance Computational Thinking Skills?* The United Kingdom: University of Southampton.

Shaniyev, Y., Gesen, I., Aidarbayev, N., Akhmetov, N., & Yerzhanov, E. (2017). *Informatics - A Bilingual Textbook - Grade 8* (1st ed.). Astana: Astana Kitap.

Simkin, M. G., & Kuechler, W. L. (2005). Multiple-choice Tests and Student Understanding: What is the Connection? *Decision Sciences Journal of Innovative Education*, *3*(1), 73-98.

Smith, M. A., & Karpicke, J. D. (2014). Retrieval Practice with Short-answer, Multiple-choice, and Hybrid Tests. *Memory*, *22*(7), 784-802.

Thissen, D., & Wainer, H. (2001). *Test Scoring*. New Jersey: Lawrence Erlbaum Associates.

Weese, J. L. (2016). Mixed Methods for the Assessment and Incorporation of Computational Thinking in K-12 and Higher Education. *Proceedings of the 2016 ACM Conference on International Computing Education Research.* ACM, 279-280.

Zeidner, M. (1987). Essay Versus Multiple-choice Type Classroom Exams: The Student's Perspective. *Journal of Educational Research*, *80*(6), 352-358.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2015). An Exploration of Three-dimensional Integrated Assessment for Computational Thinking. *Journal of Educational Computing Research*, *53*(4), 562-590.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Research on the Construction of App Inventor Program Evaluation Indicators based on Computational Thinking

Yue LIANG[1*], Jinbao ZHANG[2]
[12] Beijing Normal University, Faculty of Education, China
lybnu@mail.bnu.edu.cn, zhangjb@bnu.edu.cn

ABSTRACT

Based on the current status of Computational thinking research and practical problems of App Inventor courses, this study constructs the App Inventor program evaluation indicators based on Computational Thinking. This study refers to the existing research and related standards, established an evaluation indicators system initially. Delphi method was used to invite 9 experts in the field of computational thinking and 8 experts in App Inventor to establish an expert consultation group to consult on the rationality of evaluation indicators and to modify and improve the indicators according to expert opinions. In this study, three rounds of expert opinion consultations were conducted. After each round of expert consultations, statistical analysis and indicator revisions were conducted. After three rounds of expert consultation, the rationality of each indicator was greatly improved, and the opinions of experts tended to be consistent. The App Inventor program evaluation indicators (secondary school) that contained three first-level indicators, nine secondary indicators, and 27 third-level indicators were formed based on computational thinking.

KEYWORDS

computational thinking, App Inventor, evaluation indicators

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

基于计算思维的 App Inventor 程序评价指标构建研究

梁跃[1*]，张进宝[2]
[12]北京师范大学教育学部，中国
lybnu@mail.bnu.edu.cn，zhangjb@bnu.edu.cn

摘要

本研究基于计算思维领域研究中评价研究不足的现状及面向计算思维培养的 App Inventor 课程中亟待解决的评价问题，开展基于计算思维的 App Inventor 程序评价指标构建研究。本研究首先参考现有研究及相关标准，初步构建了评价指标体系，在此基础上采用德尔菲法邀请了 17 位领域研究及教学实践专家组建成专家咨询小组，对各指标的合理性进行咨询，经过三轮专家意见咨询，通过修改和完善最终形成了包含三级指标的基于计算思维的 App Inventor 程序评价指标（中学）。

关键字

计算思维评价；App Inventor；评价指标

1. 研究背景

近年来计算思维受到广泛重视，领域研究迅猛发展，现阶段主要集中在计算思维教育方向，呈现计算思维“扎根”教育实践的发展趋势。现有研究中计算思维课程实践形式丰富，教学工具多元化，创新性教学内容和方法不断涌现，对计算思维培养的必要性和重要性进行了全面、充分的验证。但关于计算思维的评价研究较为匮乏，在教学实践过程中缺乏对学生计算思维水平的科学评价，课程内容及教学的有效性无法得到证明。

在中学阶段的计算思维教育实践中，大多采用可视化编程教学的课程形式，其中 App Inventor 这一工具使用较多，基于该平台开展的项目研究十分丰富。通过现有研究已经充分证明了基于 App Inventor 课程实践来培养学习者计算思维的可行性，对于此类课程的内容、教学模式、相关资源建设等的探讨也已较为成熟，而课程评价研究则是目前较为缺乏的。本研究通过对实际教学的调研发现，在面向计算思维培养的 App Inventor 课程评价中，教师在评价学生的计算思维水平时存在传统评价方式和工具不适用于 App Inventor 课程、通用的程序设计课程评价模式无法对计算思维进行考察和评价等问题。为解决 App Inventor 课程中的计算思维评价问题，本研究首先构建了基于 App Inventor 课程的计算思维评价模型。根据评价模型，本研究发现目前缺乏可用于 App Inventor 程序的计算思维评价工具，因此本研究开发了基于计算思维的 App Inventor 程序评价指标体系以解决教学实践中的评价问题。

2. 文献综述

本研究以计算思维、计算思维教育等为关键词，通过 Web of Science、Springer、ProQuest 及中国知网、百度学术等在线数据库获取相关研究论文，主要对计算思维领域研究的发展及计算思维评价研究现状进行文献综述。

本研究通过文献分析发现，计算思维领域的研究从对其内涵及普遍性价值的探讨开始，逐渐聚焦于计算机及相关学科领域，最后发展为以教育教学为核心的计算思维教育研究。

从 2006 年周以真教授将计算思维重新提出开始，学者们首先对计算思维的内涵及其普遍性价值进行了讨论，主要强调计算思维对于生活中的问题解决和人的能力发展的重要性，并认为应该将其引入教育教学中，计算思维教育（Computational Thinking Education）这一提法便应运而生。自此计算思维领域研究的主要内容和核心方向开始转变为计算思维教育理论与实践。在计算思维教育理论研究中，研究者对计算思维教育的重要性、不同阶段学生计算思维能力标准、融入计算思维的课程框架、教学策略与方法等进行了研究。例如，VALERIE BARR 等人对在 K-12 教育中应如何引入计算思维以及相关教育组织、机构应该怎样提供支持进行了探讨（Barr & Stephenson, 2011）。现阶段计算思维领域研究逐渐发展为以教育实践为主。基于计算思维的教学实践囊括了从学前教育到高等教育不同教育阶段，面向不同的对象群体，并呈现普及化和常态化的趋势。

在计算思维评价研究方面，现有研究的内容和形式丰富，但评价工具类型较为集中。Werner 等人基于 Alice 平台，通过让学生完成提前设计好的不完整的程序作品来评价学生的理解、程序设计和调试等计算思维所包含的基本技能并证明了这种方法的有效性（Werner, Denner, Campe, et al., 2012）。Brennan 等人设计了三种基于 Scratch 平台的评价计算思维发展水平的方法，即所有项目文件的代码分析、学生访谈以及基于已有项目的完善和优化（Brennan, 2012）。Baichang Zhong 等开发了一个面向 Alice 平台的三维评价框架，通过收集学生的作品设计报告、问题反馈报告、访谈结果等从三个方面来对学生进行评价（Zhong, Wang, Chen, et al., 2016）。刘小燕等通过语义识别来提取和评价学生在可视化游戏开发中所体现的计算思维(刘小燕和陈艳丽，2014）。现有计算思维评价研究中主要采用的评价工具可分为三大类：计算思维概念及操作测试、情感态度量表和程序作品分析。这三类工具均适用于基于计算思维培养的 App Inventor 课程评价，其中前两类有较为成熟的工具可参考使用，但考虑到 App Inventor 课程的基本属性，程序作品作为教学的重点目标和学生课程学习的最终成果，间接体现着学生的计算思维水平，在 App Inventor 课程中从程序作品出发来开展计算思维评价是十分重要的。

3. 研究设计

3.1. 概念界定

为保证研究顺利开展、避免发生混淆，首先应对计算思维进行概念界定。本研究认为计算思维是个体自觉地运用计算机科学领域的思想方法来完成问题解决过程时进行的一系列思维活动。具体来说，这一问题解决过程主要包括问题界定、解决方案的构建和解决方案的实现三个环节，在这三个环节中主要运用计算机科学的思想、概念和方法，如问分解、抽象、编写算法等来达成解决问题的目标；当个体面临相关问题情境时，通常能够有意识地运用计算机科学领域的思想方法来完成问题解决过程并不断优化问题解决方案，且愿意与他人进行交流与合作。

3.2. 研究方法与步骤

由于现阶段可参考的评价指标体系较少，且现有评价指标信效度尚未被广泛验证。因此本研究拟采用德尔菲法来构建基于计算思维的 App Inventor 程序评价指标。

德尔菲法，也称专家咨询法或专家调查法，这种方法可以使意见收集更为可靠，其在各个领域的应用都极为广泛，通过德尔菲法得到的结果信效度较高。本研究主要使用较为常用的 SPSS 软件对专家咨询过程中形成的各指标合理性的评分结果进行数据处理，并根据各项系数的综合情况并结合专家提出的具体意见和建议来对指标进行删改。

本研究具体分为以下几个步骤：

第一步，初步构建评价指标；第二步，运用德尔菲法进行专家意见咨询；第三步，根据领域研究专家和经验丰富的教学实验者的意见和建议，对指标框架、内涵、具体描述等进行更改和完善；第四步，不断重复第二、三步，直至专家意见达成一致，最终形成完整的基于计算思维的 App Inventor 程序评价指标体系。

4. 初步构建评价指标

本研究在计算思维概念界定的基础上，考虑到程序作品作为解决方案实现的最终结果，可以通过其中的代码设计、算法运用及作品特征等进行计算思维评价。由于评价指标的评价目标群体为中学阶段学生，本研究主要根据中学阶段学生计算思维发展水平及表现性标准来获取和设计评价指标。为了使得评价指标的适用范围更广，本研究仅关注 App Inventor 程序中较为核心的组件和功能，突出程序作品中计算思维的核心表现。

本研究根据评价指标的构建思路和概念界定，为保证评价的逻辑性和合理性，采用了 3 级指标划分结构，一级指标为维度指标，主要反映计算思维的维度属性，也是主要的评价目标；二级指标为分类指标，选取维度下的计算思维核心要素，是一级指标的具体分类；三级指标为观测指标，是计算思维在 App Inventor 程序中的具体表现，主要根据这一级指标来进行评价。为获取各级指标项，本研究首先对不同计算思维定义及评价研究中的计算思维核心要素进行提取，并构建计算思维基本要素合集。本研究根据计算思维的概念界定并参考大量 App Inventor 程序，对合集中的要素进行分类整理，设计了三个不同的 App Inventor 程序评价维度，即一级指标，分别为基本概念、过程处理和程序设计，兼顾计算思维与程序设计两方面的评价。在明确评价指标的维度方向和基本要素集后，本研究对中学阶段学习者的计算思维水平或表现性标准进行了调研，并选取了在 App Inventor 程序中可以体现的、中学阶段学习者应当掌握的计算思维基本要素，包括抽象、问题分解、数据表征与处理、算法、逻辑表达、调试，并将其按照维度分类进行划分、命名，形成二级指标，另外从传统程序设计评价的角度出发，本研究还添加了完整性、复杂度和灵活性两个分类指标。最后，本研究根据 App Inventor 程序开发过程中不同阶段的计算思维体现，进一步明确了各观测指标项，即三级指标。最终初步拟定了由 3 个维度指标，9 个分类指标，28 个观测指标构成的评价指标框架。具体如下图所示（由于内容较多未包含观测指标）：

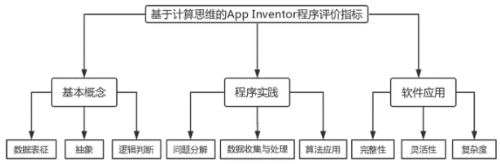


图 1 初步拟定的第一版评价指标结构图

在指标内涵方面，本研究维度指标建立在现有评价研究和计算思维概念界定的基础之上，从问题解决过程的三个环节出发，在进行问题的界定即初步构建 App Inventor 程序阶段，主要考察学习者对计算思维基本要素或者计算机科学基本技能的应用，因此第一个分类维度设计为概念运用，该维度下的三个分类指标分别为：问题表征，在程序开发过程中能够将要解决的问题形式化并选取正确的数据表征形式；数据组织，在程序设计中对程序运行中产生的数据进行了收集、传递和处理；逻辑判断，通过条件判断和逻辑分析设计了针对不同情况的程序操作。在问题解决方案构建和实现的环节中，即 App Inventor 程序开发、调试形成最终作品的过程，因此第二个维度指标为过程处理，旨在评价学习者在问题解决过程中（程序开发过程中）是否运用了合理、准去的方法来编写代码。该维度下三个分类指标为：模块化，程序体现了模块化问题解决的思想；抽象化，程序通过抽象特征、模型化代码设计方式进行问题解决；自动化，程序运用算法思想，通过创建包含一系列步骤或组件的过程等来自动化处理问题或完成操作。第三个维度指标为程序设计，旨在从程序作品的代码特征角度出发来进行评价。该维度下三个分类指标为：完整性，程序作品的功能、代码等的完整性及程序运行的正确性；灵活性，程序作品中代码设计具有代表性，可迁移性到其他相似的程序中；复杂度，程序作品组件数量、功能复杂程度等。

根据指标内涵，本研究阅读大量 App Inventor 程序，主要针对程序作品的组件、逻辑及界面、功能设计进行了观测指标的设计，由于篇幅有限不在此进行具体说明。

5. 德尔菲法完善评价指标

本研究在初步构建的基于计算思维的 App Inventor 程序评价指标的基础上，以提升评价指标的合理性和有效性为主要目标，采用德尔菲法对该领域专家和实践者进行咨询，完善指标的结构和表述。在运用德尔菲法完善指标的过程中，本研究主要经历了组建专家咨询小组、发放和回收第一、二轮专家咨询表并对指标进行修改、发放和回收第三轮专家咨询表并确定最终版本的评价指标这一系列研究过程。

本研究根据德尔菲法研究规范，将专家的选择集中在计算思维领域理论研究专家、App Inventor 教学实践专家和我国课程标准研究小组成员的范围内，选取了 10 名高校计算思维理论及实践研究专家，其中有 3 位为我国信息技术课程标准制定小组成员，大部分专家发表过较多计算思维研究论文且影响力较大，还有专家开展了丰富的基于计算思维教学实践或师资培训的项目研究；7 名 App Inventor 教学实践专家，均为中学阶段开展 App Inventor 教学的信息技术教师，其中 5 位教师积极参与了基于计算思维的 App Inventor 教学实践研究项目，另外 2 位教师在全国 App Inventor 教学研讨群中较为活跃，且对计算思维这一领域研究较为熟悉。

5.1. 第一轮专家意见咨询

在初步构建的评价指标的基础上，本研究编制了专家意见咨询表，并通过发送邮件、QQ 文件等方式向 17 位专家发送了第一轮专家咨询表。最后，第一轮德尔菲法专家意见咨询共回收 15 份有效的意见反馈表，回收率为 88%，共有 12 位专家除评分外还提出了十分具体的反馈意见，意见提出率为 80%，证明专家对本研究关心程度较高，并通过对相关数据分析得出专家权威程度系数较高，专家评分的信度较高。

在第一轮专家咨询中，专家对各指标项的合理性进行了评分，通过对评分结果的数据统计分析，可以通过平均值、标准差和差异系数了解专家的意见集中程度、离散程度和协调程度。第一轮专家咨询的具体数据统计结果如下表 1 所示（其中编号项为具体的观测指标，因篇幅有限以编号代替）：

表 1 第一轮专家咨询评分描述性统计分析结果

	N	平均数	标准差	变异系数
A 概念运用	15	4.27	.884	0.207
A1 问题表征	15	4.33	.816	0.184
A1-1	15	4.40	.828	0.188
A1-2	15	4.67	.488	0.104
A1-3	15	4.33	.900	0.208
A2 数据组织	15	4.40	.737	0.168
A2-1	15	4.47	.640	0.143
A2-2	15	4.33	.617	0.142
A2-3	15	4.27	1.033	0.242
A3 逻辑判断	15	4.47	.834	0.187
A3-1	15	4.60	.828	0.180
A3-2	15	4.60	.507	0.110
A3-3	15	4.33	.816	0.188
B 过程处理	15	4.33	.816	0.188
B1 模块化	15	4.27	.884	0.207
B1-1	15	4.53	.640	0.141
B1-2	15	4.27	.799	0.187
B1-3	15	4.00	.756	0.189
B2 抽象化	15	4.33	.900	0.208
B2-1	15	4.47	.834	0.187
B2-2	15	4.53	.834	0.184
B2-3	15	3.53	.990	0.280
B3 自动化	15	4.00	1.000	0.250
B3-1	15	4.20	.862	0.205
B3-2	15	4.53	.834	0.184
B3-3	15	4.00	.926	0.232
B3-4	15	3.07	1.033	0.336
C 程序设计	15	4.13	1.060	0.257
C1 完整性	15	4.60	.507	0.110
C1-1	15	4.13	.915	0.222
C1-2	15	4.67	.488	0.104
C1-3	15	4.27	.799	0.187
C2 灵活性	15	3.87	1.060	0.274
C2-1	15	3.73	1.033	0.277
C2-2	15	4.20	.775	0.185
C2-3	15	3.73	1.100	0.295
C3 复杂度	15	4.00	1.134	0.284
C3-1	15	3.73	1.033	0.277
C3-2	15	3.87	.834	0.216
C3-3	15	3.67	.976	0.266
有效的 N	15			

根据上表的数据结果可知，第一轮专家咨询中，平均得分小于 4 即不太合适的指标项的共有 8 个，主要集中在“程序设计”及其维度下的相关指标项中，跟专家的反馈意见结果较为一致；标准差大于 1 的指标项有 9 个，在前两个维度下分别有一个，剩下集中在第三个维度下，表明专家在进行评分时对这几个指标项的理解和认可度有偏差，离散程度较大；变异系数大于 0.3 的只有一项，且该项指标的平均值和标准差均不符合指标筛选的要求，应删除该指标项。另外几项评分结果不够理想的指标项的变异系数较低，可保留并根据专家反馈意见进行修改。对于三级指标，专家就指标的具体描述、指标内容的重复性、片面性以及指标所在分类的合理性等提出了很多意见，有部分指标专家未能理解指标的含义，应进一步指标的可读性。本研究将根据专家意见优化表述方式、增删指标项。

获取了第一轮专家反馈意见后，本研究根据专家提出的修改意见和参考建议对指标进行了修改，并形成了基于计算思维的 App Inventor 程序评价指标第二版。

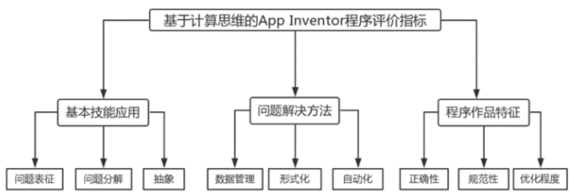


图 2 评价指标第二版结构图（一二级）

5.2. 第二、三轮专家意见咨询

经过第一轮专家意见咨询及指标修改后，本研究开展了第二轮专家意见咨询对修改后的指标合理性进行评分和意见收集。在第二轮专家咨询中，专家咨询表回

收率为93%，即专家积极系数为93%，证明专家积极程度较高。其中有 6 位专家提出了意见，意见提出率为43%，证明专家对本研究的关注度较高。与第一轮相同，本研究对第二轮专家咨询评分结果进行了统计分析，根据第二轮专家评分数据统计结果，经过修改后的各项指标已经满足了指标筛选的标准，专家意见也已经达成了一致。从整体来看存在问题的指标项较少，但由于第三维度指标改动较大，专家们对此部分指标提出了修改和完善意见，意见主要集中在维度指标的具体命名、观测指标的交叉、重复等方面，经修改形成了基于计算思维的 App Inventor 评价指标第三版，其一二级指标结构图如下图 3 所示。

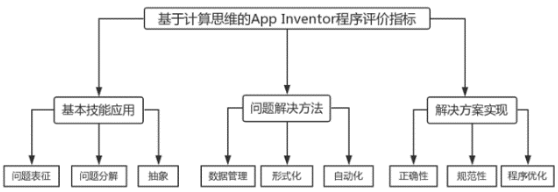


图 3 评价指标第三版结构图（一二级）

第三轮专家意见咨询主要针对上轮咨询中存在问题的指标项。此轮咨询共发放 14 份咨询表，回收 13 份，回收率 93%，家提供意见反馈的积极程度较高。第三轮与前两轮相同，主要对评分数据进行描述性统计。根据统计结果发现，修改后的各项指标的统计结果较第二次更好，证明在第三轮中很好地完成了指标的优化工作。

通过德尔菲法咨询领域专家意见，修改、完善各指标项，在专家小组的共同决策下形成了最终版本的评价指标，其具体结构图如图 3 所示。

6. 研究结果与讨论

经过三轮专家咨询评价指标逐步完善。从整体趋势来看，从第一轮到第三轮，指标项评分平均值逐渐变大，变异系数逐渐减小，证明指标项的描述越来越准确、专家意见也逐渐趋于一致。本研究组建的专家小组专家权威度较高，三轮咨询中的积极程度和对研究的关注度也较高，保证了最终结果的可信性。

在指标内容方面，第一轮专家咨询后，对评价指标的维度、结构和具体表述进行了大规模的修改，解决了一开始存在的问题，并形成了更加规范、科学、有效的指标体系；第二轮专家咨询后各指标项评分已达到较高值，仅对少部分指标进行了修改，以提升评价指标的严谨性和准确性；最后经过第三轮专家咨询形成了较为完善的评价指标体系。例如，在初步构建的评价指标中，本研究在“概念运用”一级指标中包含了“问题表征、数据组织、逻辑判断”三个二级指标，其中“问题表征”分类指标下的观测指标为“程序组件符合对象特征/功能定位、程序中正确地使用全局/局部变量来表示发生变化的数据、程序中正确地使用列表来表示多个数据项/实现从多个数据中进行选择”；经过三轮专家咨询，最终版评价指标将“概念运用”改为“基本技能应用”，共包含“问题表征、问题分解、抽象”三个分类指标，其中“问题表征”分类下包含“程序组件的选取符合对象的特征，与程序功能相对应、程序中正确地使用全局/局部变量来表示发生变化的数据、程序中正确地使用列表来表示多个数据项或实现从多个数据中进行数据选择”三个观测指标。对比最终版本和第一版评价指标，可以发现经过德尔菲法专家咨询，指标的维度更加合理，符合本研究对计算思维的界定及程序设计所能体现的计算思维要素和学习者思维活动水平；分类指标更加清晰，与实际教学更为贴合；观测指标描述更加具体、合理，更便于在实践中的具体操作。

本研究运用德尔菲法构建专家咨询小组，开展了三轮专家咨询，最终形成了较为完善的基于计算思维的 App Inventor 程序评价指标体系，指标结构清晰简明，与计算思维界定密切相关，同时又通过观测指标与程序表征相关联，本研究选用的从基本技能、编程方法到方案实现的维度分类与大部分计算思维教学实践的整体思路相吻合，同时也符合新课标等对计算思维的定义和培养标准。这不仅能够为一线教师的教学实践提供支持，同时对于计算思维评价研究也具有一定价值。在后续研究中，可继续本研究通过层次分析法计算指标权重，并在明确具体评分规则的基础上对评价指标的信效度进行检验。在此基础上，还可将评价指标用于教学评价中，通过不断实践来完善指标体系。

7. 参考文献

刘小燕和陈艳丽（2014）。可视化编程中识别计算思维。**计算机科学**，**41**（**s2**），403-407.

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 *AERA 2012*.

Folk, R., Lee, G., Michalenko, A., et al. (2015). GK-12 DISSECT: Incorporating Computational Thinking with K-12 Science Without Computer Access. *Proceedings of* 2015 *IEEE Frontiers in Education Conference (FIE)*. IEEE.

Werner, L., Denner, J., & Campe, S. (2012). The Fairy Performance Assessment: Measuring Computational Thinking in Middle school. *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education.* New York: ACM, 215-220.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Development of a Computational Thinking Scale for Programming

Yuan-kai CHU[1], Jyh-chong LIANG[2], Meng-jung TSAI[3*]
[123] National Taiwan Normal University, Taiwan
K082792@gmail.com, aljc@ntnu.edu.tw, mjtsai99@ntnu.edu.tw

ABSTRACT

The purpose of this study is to develop a scale to assess students' computational thinking for programming. A total of 427 middle school students were surveyed with a questionnaire about their computational thinking in a computer programing context. After an explorative factor analysis, a total of 20 items categorized in five factors were drawn in the final version of the scale. The five dimensions were abstraction, algorithm, decomposition, evaluation and generalization. In addition, the validity and the reliability of the scale were also examined in this study. The reliability of Cronbach' α was .91 for the overall scale. This suggests that the scale is good enough for examining students' computational thinking in a programming learning context.

KEYWORDS

computational thinking, programming, assessment tool

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

程式設計之運算思維量表

朱元楷[1*]，梁至中[2*]，蔡孟蓉[3*]
[123]國立台灣師範大學，臺灣
K082792@gmail.com，aljc@ntnu.edu.tw，mjtsai99@ntnu.edu.tw

摘要

本研究旨在開發檢測程式設計情境中的運算思維量表研究工具。透過問卷調查的方式，本研究收集 427 份國中學生樣本資料，經由探索性因素分析和信效度分析，確認此運算思維量表最包含抽象化、演算法、評估、分解與一般化五個構面，且總量表信度為.91，因此此量表具備良好之信效度，可作為未來程式設計教學者或研究者檢測學習者運算思維的工具之一。

關鍵字

運算思維；程式設計；評量工具

1. 前言

近年來，資訊科技的發展日新月異，廣泛的影響了人類的生活，然而教育也不例外。為了因應科技的快速變遷，各個國家的教育單位積極推動資訊教育的改革，不論是在美國、英國、德國、澳洲、韓國等國家不謀而合的皆對學生的資訊能力提出完善的課程導入政策，其中值得關注的是運算思維的發展，這些國家紛紛在國小開始設立資訊課程，目的為提早培養孩童的運算思維能力。舉例來說，美國在 2011 年針對電腦科學（Computing） 課程重新修訂以運算思維為主軸的課程系統；英國的電腦科學課程也同樣特別強調運算思維及創造力的培養（DOEE, 2013）；澳洲教育單位在中小學的課綱中強調學生對於數位系統的運用與運算思維的能力培養（ACARA, 2013），上述國家針對學生在運算思維能力的提升皆提出了明確的綱要與方案，目的皆希望能夠從小就讓學生具備二十一世紀重要的素養能力。

然而，我國教育部近年提出了 108 課綱的草案，內容提到針對國小、國中、高中學生資訊與科技課程的改革方案，以運算思維為主軸，透過學習電腦科學相關知能，培養邏輯、系統化思考等運算思維，並整合運算思維與資訊科技工具，實際產出資訊相關作品，最終增進學生運算思維的應用能力、問題解決、團隊合作以及創新思考等高層次思考的能力（十二年國民基本教育科技領域課程綱要草案，2016）。此外，教育部於 106 年也提出了「運算思維推動計畫」，其中提及運算思維的能力在國小階段的學生應能運用資訊科技工具處理生活的基礎事務。由上述可知，運算思維在近五到十年來已成為了國際教育間重要的發展方針，各個國家皆認為運算思維的培養必須從低學齡開始，透過完整的架構與系統化的課程，達到最終學生運算思維的養成。

2. 文獻探討

2.1. 運算思維的定義

運算思維（Computational Thinking）最早由電腦科學專長的學者 Jeannette M. Wing（2006）提出，他認為運算思維是探討人類的行為、系統設計與問題解決的思考流程，並且歸納了以下概念：（1）概念構思而非電腦編程：運算思維是將問題與需求進行多層次的構思，並非專指電腦的程式編寫。（2）基礎能力而非死板技能：強調運算思維是基礎能力的培養，而非一般死背呆板的技能。（3）人類思路而非電腦思維：強調是人類進行問題解決的方式與思維，而非模仿電腦的思考方式。（4）創意發想而非實體產出：強調運算思維是在生活中的人事物的概念發想思維，而非是針對軟硬體的產出。（5）適用於任何人任何場域：強調運算思維是不受限於個人與空間，是一種思維模式的養成。因此，許多學者也根據以上概念提出了運算思維的四大基石：問題拆解（Decomposition）、模式辨識（Pattern Recognition）、抽象化（Abstraction）與演算法（Algorithm Design），這四個步驟也成為了國際許多運算思維研究的參考依據。另外一方面，Selby 與 Woollard （2013）也參考 Wing（2008）所提出之概念，定義了運算思維的五個構面（Abstraction）、演算法（Algorithm）、評估（Evaluation）、分解（Decomposition）與一般化（Generalization）。

由於各國對於運算思維能力的重視日趨，資訊科技產業的龍頭「Google」2015 年在其教育的專欄中提到，運算思維是一種問題解決的過程，其中包含多樣的屬性與方針。此外，它更針對運算思維突出了 11 項重點：抽象化（Abstraction）、 演算法設計（Algorithm Design）、自動化（Automation）、資料分析（Data Analysis）、資料蒐集（Data Collection）、 資料表示（Data Representation）、分解（Decompositon）、平行計算 （Parallelization）、一般化（Pattern Generalization）、模式辨識（Pattern Recognition）、模擬（Simulation）。

綜合上述對於運算思維的定義與理解，可以發現在Google定義的11項概念中，有許多個構面與Wing, Selby, & Woollard所提出的概念完全相同，其他概念為Google根據自身對於資訊科技的理解進行拆解與衍生的概念，說明三者對於運算思維的概念具有高度的相關並且具有衍生性的關係。本研究最終採用Selby & Woollard（2013）所提出運算思維的概念進行問卷的開發。

2.2. 運算思維的現況與量表的發展

Kalelioglu, Gulbahar, & Kukul（2016）發表了一篇以運算思維為主題的文獻回顧，其提出了一個架構針對現今對於運算思維學術論文的目標對象、理論基礎、定義、範圍、類型和研究設計進行分析與探討。此研究整理了 125 篇以運算思維為主的研究，分析這些研究主要的目的，其中有 43 篇提出運算思維的課程與活動設計，34 篇研究提供專題與運算思維活動進行運算思維的教學；26 篇針對運算思維定義的探討與比較；24 篇說明一個教育的系統或平台用於提升學生運算思維的能力；13 篇提出一個教學策略的架構；4 篇進行構面的探討與分析。因此，此研究最後提及雖然目前有許多研究皆以運算思維為主題，教師依照現有對於運算思維的模式進行課程設計相對容易，但卻鮮少研究針對學生在運算思維能力的表現上進行評量，包含評量的方式與量表目前皆缺乏完整的系統架構，因此希望未來能有更多研究專於開發學生在運算思維能力的評鑑工具。

然而，近兩年來，開始有少數研究提出運算思維的量表用於檢驗學生在此部分能力的提升，因此 Korkmaz, Çakir, & Özden（2017） 開發了一個運算思維的量表，目的用於檢測學生對於運算思維能力的程度與發展狀況，並且認為此量表對於測量運算思維有極大的意義，因此針對大學生採用兩個階段的問卷蒐集，分別蒐集了 726 位與 580 位大學生樣本進行問卷開發。此外，本篇研究根據文獻內容將運算思維與二十一世紀關鍵能力進行整合與聚斂，最終提出了以下構面：創造力（Creativity）、演算思維（Algorithmic thinking）、合作能力（Cooperativity）、批判性思考（Critical thinking）、問題解決（Problem solving），以上五個構面透過量化的統計分析後皆呈現良好的信度與效度，因此為良好的評量工具。然而，此研究雖然蒐集的大量的樣本，但研究的對象局限於大學生，較難判斷是否適用於其他學齡段的學生。此外，此研究將運算思維與二十一世紀關鍵能力整合為一，雖然具備相關理論的闡述與良好的信效度，但相較於 Wing （2006）所提出運算思維的重要概念，僅僅只保留了演算思維（Algorithmic thinking）唯一構面，因此本研究認為此量表較難真正的評量學生在運算思維能力全方面的廣度與深度。

綜合上述文獻可以得知，目前在運算思維量表工具的開發較為缺乏，大多數的研究還是針對課程的融合與教學活動的設計，雖然有少數研究進行量表的開發與構面的分析，但卻未保留運算思維的核心概念。因此，本研究參考 Wing（2006）所提出了四個構面：問題拆解（Decomposition）、模式辨識（Pattern Recognition）、抽象化（Abstraction）與演算法（Algorithm Design），與 Selby & Woollard（2013）所提出的構面進行微調，並且針對學生在程式領域的學習，最終提出了五個構面：抽象化（Abstraction）、演算法（Algorithm）、評核（Evaluation）、問題拆解（Decomposition）與歸納（Generalization）。目的在於提供未來學生在程式學習中的運算思維能力評量工具，成為具有高度價值的運算思維量表。

3. *研究方法*

3.1. 研究對象

本研究對象為台灣私立中學共 427 人，透過實際的紙本問卷搜集，其中學生共有 216 人（50.6%）為國中二年級，211 人（49.4%）為國中三年級，年級分佈達到近各半數的比例。此外，這些學生有高達 96.1%的比例具備程式學習的經驗，且超過一半（72.7%）的學生具有一年以上的程式學習經驗，說明這些樣本適合用於運算思維的量測與量表開發，如表一所示。

表 1 研究對象背景資料

項目	屬性	N	%
性別	男	277	64.9
	女	148	34.7
學級	國中二年級	216	50.6
	國中三年級	211	49.4
曾學過的程式語言含正在學的，可複選	Scratch	371	86.9
	Scratch 和其他	39	9.20
	無	13	3.00
您是否有自己看書或上網學習程式設計的經驗?	是	153	35.8
	否	269	63.0
您是否曾經參加過程式設計的課外活動或營隊?	是	59	13.8
	否	365	85.5
父母或家中是否有成員從事資訊領域的相關工作?	是	99	23.2
	否	320	74.9
您接觸程式設計的時間總共大約?	不到 1 年	117	27.4
	1-3 年	225	52.7
	3-5 年	62	14.5
	5-10 年	21	4.90

3.2. 研究工具

本研究工具採用 Wing（2006）與 Selby & Woollard（2013）所提出對於運算思維的定義，進行修改延伸，最終提出了以下五個構面，抽象化（Abstraction）共有四題；演算法（Algorithm）共有五題；評估（Evaluation）共有四題；分解（Decomposition）共有三題與一般化（Generalization）共有四題，因此本研究工具為二十題量表，其選項採用五點量表包含，非常同意、同意、普通、不同意、非常不同意。例題如下：Abstraction1 我通常會試著思考程式問題與結果的呈現方式；Algorithm1 我通常會想出解決程式問題的詳細步驟。Evaluation1 我通常會試著找出正確的程式問題解決方案。Decomposition 1 我通常會試著思考程式問題被拆解的可能性。Generalization1 我通常會試著根據以往解決程式問題的經驗來解決新的問題。

3.3. 資料分析

本研究採用量化分析研究方法，透過 SPSS 統計軟體進行資料處理與分析，透過以下分析方法進行。（1）相關分析：探討各個變項間的關係。（2）信效度分析：探討量表內的正確性與一致性。（3）探索性因素分析：

進行各個變項與題目間的分析與判斷，最終篩選出完整的題目與向度。

4. 研究結果

4.1. 信效度與相關分析之結果

本研究透過信效度分析檢驗此量表是否具備足夠的穩定性與一致性，在表 2 的相關分析顯示總體向度的相關係數介於.42 至 61 之間且皆達顯著（*P<.01*），屬於中高度的相關，顯示各個向度具有一定程度的相互關係。此外，此量表呈現相關矩陣和每個構面數值的 AVE 平方根用於評估判別有效性，每個構面的 AVE 的平方根應該大於.50（Fornell & Larcker, 1981）並且大於其他構面的相關係數（Chin, 1998），表現如下：Abstraction 為.75；Algorithm 為.64；Evaluation 為.78；Decomposition 為.75；Generalization 為.69，說明各變項具有出獨特性與鑑別度。

表 2 相關分析與區別效度

	1	2	3	4	5
Abstraction	**.75**				
Algorithm	.61**	**.64**			
Evaluation	.48**	.54**	**.78**		
Decomposition	.42**	.55**	.46**	**.75**	
Generalization	.45**	.50**	.54**	.46**	**.69**

再者，如表 3 為本問卷各向度在的平均值介於 3.23 至 3.72 之間，在五點量表中呈現偏高的正向資訊，另外此量表總體的 Cronbach'α 為.91，其各別構面表現如下：Abstraction 的 Cronbach'α 為 .82 ； Algorithm 的 Cronbach'α 為.81；Evaluation 的 Cronbach'α 為.83；Decomposition 的 Cronbach'α 為.74；Generalization 的 Cronbach'α 為.75，因此可以得知所有向度信度皆大於.70 顯示此量表具有良好的一致性與穩定性。

表 3 信度分析

	M	SD	Cronbach' α
Abstraction	3.52	.75	.82
Algorithm	3.40	.72	.81
Evaluation	3.72	.75	.83
Decomposition	3.23	.83	.74
Generalization	3.62	.71	.75

4.2. 探索性因素分析之結果

本研究使用探索性因素分析檢驗各構面間題項的潛在因素，並且排除各變項間的共變程度，最終留下各構面間具有代表性的題項。本研究採用斜交轉軸法，用於檢驗此量表的各項構面，結果顯示最終的量表保留了原始的五個構面。此外，如表 4、5、6 所示，提供了斜交轉軸法的樣式係數（pattern coefficient）與結構係數（Structure coefficients），其中主要以樣式係數為判讀指標，代表每一變項對於因素之獨特貢獻量，其數值大於 0.4 題項具有參考價值（Lee, Johanson & Tsai, 2008），且透過表格可以得知各題項在各自構面的表現，factor1：Abstraction 的樣式係數介於.67 至.82 間；factor2：Algorithm 的樣式係數介於-.41 至-.77 間；factor3：Evaluation 的樣式係數介於.66 至.87 間；factor4：Decomposition 的樣式係數介於.68 至.80 間；factor5：Generalization 的樣式係數介於.51 至.82 間，可以得知，五個構面的樣式係數皆大於.4 且在其他因子的表現上小於 0.4（Stevens, 1996）。

表 4 Factor1 和 2 轉軸係數

	Factor 1		Factor 2	
	P	S	P	S
Abstraction 1	**.73**	**.76**	.02	-.34
Abstraction 2	**.82**	**.83**	-.09	-.42
Abstraction 3	**.78**	**.82**	.03	-.39
Abstraction 4	**.67**	**.74**	-.09	-.43
Algorithm 1	.36	.57	**-.41**	**-.61**
Algorithm 2	.22	.52	**-.56**	**-.72**
Algorithm 3	.05	.41	**-.77**	**-.80**
Algorithm 4	-.02	.31	**-.73**	**-.77**
Algorithm 5	.00	.37	**-.68**	**-.74**
Evaluation 1	.09	.39	.04	-.36
Evaluation 2	.07	.37	.02	-.36
Evaluation 3	.05	.39	-.07	-.42
Evaluation 4	-.11	.22	-.07	-.35
Decomposition1	.05	.30	.03	-.36
Decomposition2	.13	.39	.05	-.37
Decomposition3	-.08	.20	-.11	-.40
Generalization 1	-.06	.25	-.10	-.34
Generalization 2	.12	.36	.03	-.25
Generalization 3	-.02	.28	-.08	-.35
Generalization 4	.04	.32	.01	-.28

表 5 Factor3 和 4 轉軸係數

	Factor 3		Factor 4	
	P	S	P	S
Abstraction 1	.09	.38	-.10	.17
Abstraction 2	.11	.38	-.06	.22
Abstraction 3	.00	.35	.12	.36
Abstraction 4	-.05	.30	.14	.36
Algorithm 1	-.03	.34	.08	.37
Algorithm 2	.02	.39	.11	.42
Algorithm 3	.08	.42	-.12	.27
Algorithm 4	.00	.30	.26	.50
Algorithm 5	.10	.43	-.08	.29
Evaluation 1	**.72**	**.79**	.10	.36
Evaluation 2	**.87**	**.86**	.05	.33
Evaluation 3	**.84**	**.87**	-.04	.28
Evaluation 4	**.66**	**.71**	.02	.27
Decomposition1	.08	.35	**.80**	**.83**
Decomposition2	.02	.35	**.76**	**.82**
Decomposition3	.06	.32	**.68**	**.74**
Generalization1	-.06	.32	.11	.33
Generalization2	-.06	.32	-.01	.22
Generalization3	.16	.45	.13	.36
Generalization4	.22	.49	-.04	.22

表 6 Factor5 轉軸係數

	Factor 5

	P	S
Abstraction 1	.13	.37
Abstraction 2	-.14	.19
Abstraction 3	.07	.35
Abstraction 4	.04	.30
Algorithm 1	.08	.34
Algorithm 2	.06	.34
Algorithm 3	.10	.36
Algorithm 4	-.19	.11
Algorithm 5	.15	.39
Evaluation 1	.05	.40
Evaluation 2	-.09	.31
Evaluation 3	-.01	.38
Evaluation 4	.11	.39
Decomposition1	.00	.27
Decomposition2	.12	.37
Decomposition3	.04	.27
Generalization 1	**.75**	**.77**
Generalization 2	**.82**	**.82**
Generalization 3	**.51**	**.63**
Generalization 4	**.64**	**.73**

5. 結論與建議

本研究透過統計的分析與資料的彙整，最終開發了一個具備良好信度與效度的運算思維評量工具，用於檢測學生對於運算思維能力的程度與培養，其主要分為五個構面：抽象化（Abstraction）、演算法（Algorithm）、評估（Evaluation）、分解（Decomposition）與一般化（Generalization），透過統計的分析與判斷後最後完成二十題的五等第運算思維量表。此量表工具的開發能夠有效的呼應 Kalelioglu 等人（2016）所提出的目前缺乏具有完整系統與理論基礎的運算思維評量工具，協助教師評估與判斷學生在運算思維能力成長的良好依據。因此，面對目前龐大關於運算思維活動設計的研究，本研究所開發之運算思維量表能夠有效幫助教師進行評鑑，具有高度價值。

然而，本研究限制為研究樣本目前主要為國中二年級與國中三年級，年齡介於 14 至 15 歲，且皆屬於台灣北部地區單一學校所收集的學生，因此無法做更廣泛的推論於探究。

根據本研究之研究結果，建議未來能夠擴大樣本的豐富性，包含不同地區、不同學齡、不同國籍進行樣本的蒐集，此外，未來建議能夠針對背景變項例如：程式語言的學習經驗進行進一步的研究分析與探討。最後，希望未來針對運算思維的研究能夠開發出更多的評量工具與測驗工具，包含對於學生與老師，甚至是針對課程與活動設計的評量方式，都是在未來的研究可以進行發展的重要主題。

6. 致謝

本研究感謝以下科技部計畫編號之研究經費補助：MOST 106-2511-S-003-065-MY3、MOST 106-2511-S-003-064-MY3 與 MOST 105-2628-S-011-002 -MY3。

7. 參考文獻

Australian Curriculum, Assessment, Reporting Authority (ACARA). (2013). *Draft Australian Curriculum Technologies.* Retrieved January 21, 2019, from http://consultation.australiancurriculum.edu.au/Static/docs/Technologies/Draft%20Australian%20Curriculum%20Technologies%20-%20February%202013.pdf

Chin, W. W. (1998). The Partial Least Squares Approach for Structural Equation Modeling. *Modern Methods for Business Research, 295*(2), 295-336.

Department for Education in England (DOEE). (2013). *National Curriculum in England: Computing Programmes of Study*. Retrieved January 21, 2019, from https://www.gov.uk/government/publications/national-curriculum-inenglandcomputing-programmes-of-study/national-curriculum-inenglandcomputingprogrammes-of-study

Fornell, C., & Larcker, D. (1981). Structural Equation Models with Unobservable Variables and Measurement Error. *Journal of Marketing Research, 18*(1), 39-50.

Google. (2015). *Computational Thinking for Educators.* Retrieved January 21, 2019, from https://computationalthinkingcourse.withgoogle.com/

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking based on a Systematic Research Review. *Baltic Journal of Modern Computing, 4*(3), 583-596.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A Validity and Reliability Study of the Computational Thinking Scales (CTS). *Computers in Human Behavior, 72*, 558-569.

Lee, M. H., Johanson, R. E., & Tsai, C. C. (2008). Exploring Taiwanese High School Students' Conceptions of and Approaches to Learning Science through a Structural Equation Modeling Analysis. *Science Education, 92*, 191-220.

Stevens, J. (1996). *Applied Multivariate Statistics for the Social Science (3rd ed.)*. Mahwah, NJ: Erlbaum

Selby, C., & Woollard, J. (2013). Computational Thinking: the Developing Definition. *Proceedings of Special Interest Group on Computer Science Education (SIGCSE) 2014.* New York: ACM.

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

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Non-formal Learning

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

The Learning Effectiveness of Integrating Computational Thinking and English Oral Interaction

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

ABSTRACT

This study was aimed at exploring the effects on students' English learning and learning performance of computational thinking when the educational robots with computational thinking are integrated into the game-based learning of English oral interaction. On the other hand, the control group used the unplugged board games, and integrated computational thinking into the English oral interactive learning game. The experimental results indicated that through the integration of computational thinking and the game-based learning of English oral interaction, whether with the unplugged tools or the educational robots, the same computational thinking with English oral interactive game-based learning content can all effectively improved learners' learning performance on English and computational thinking. However, the learning effectiveness of the group adopting the educational robots was significantly higher than the group using the unplugged tools.

KEYWORDS

board game, educational robotics, computational thinking, game-based learning

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

整合運算思維與英語互動的成效分析

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

摘要

本研究旨在探討教育機器人整合運算思維融入英語口語互動遊戲式學習時，對於學生英語學習及運算思維學習的成效影響。對照的是使用不插電的桌上型遊戲，將運算思維融入英語口語互動遊戲式學習。實驗結果表明，透過整合運算思維與英語口語互動遊戲式學習，不論是採用不插電媒介或是教育機器人，相同的運算思維整合英語口語互動遊戲式學習內容，都可以有效提升學習者在英語學習及運算思維上的學習成效。然而，採用教育機器人的組別，學習成效顯著高於不插電的組別。

關鍵字

桌遊；教育機器人；運算思維；遊戲式學習

1. 前言

英語現在已經成為全球通用的語言，字彙對跨課程的理解和成果是不可或缺的；理解是一個可以有效幫助字彙學習的因素之一（Ardasheva, Carbonneau, Roo, & Wang, 2018）。但是在學習英語文法與記憶英語單字方面通常被認為是一件非常無趣的事，並且可能是一項會使學習者畏縮的作業（Tsai & Tsai, 2018），導致學習者在學習英語上會比較沒有動力學習（Wu, 2018）。在傳統的教學裡，學習者在學習英語上都是採獨自學習的方式，有些學習者在獨自學習的過程中遇到問題時，會希望可以馬上得到解答，或是在得知答案後了解這個解答是如何解出的，但是如果當下沒有人可以幫忙解答或是可以一起討論、一起研究問題該如何解答時，就會一直糾結在該問題上，也會不知道該如何解決，漸漸地會變得沒有自信，甚至覺得學習英語是一件無趣的事。Wang（2018）指出在英語教學領域上透過合作學習可以提升學習者對英語學習的動機，合作學習是採小組的教學方式，提供學習者與自己的組員共同合作並完成一個學習目標（Johnson & Johnson, 2018），通過合作可以擴增學習者們的思考能力同時擴增他們自我的想像力，所以如果學習者從獨自學習變成兩個人以上的合作學習，將可以透過溝通合作一起研究並釐清出自己所無法解答的問題，這將會使學習者在學習時，提升學習的動力同時增加他們對學習的有趣性。

不過也有研究指出，雖然合作學習可以促進小組間與整個教室的參與並提高他們的學習動機，但它不足以促進學習者提升他們的學習成績（Premo, Cavagnetto, Davis, & Brickman, 2018）。然而隨著科技的興起，遊戲已經被證明可以成為各種具有優勢和有效性的方式，幫助學習者學習字彙，它也可以激勵學習者練習和學習新的單字。遊戲融入學習還可以使教育相關領域挑戰並創新，包括教學方法和學習第二外語及外語的方法。未來應打破現今課堂單科學習模式，將運算思維的素養融入課程，例如模組化，自動化等，將運算思維使用在不同主題、不同教師的課程，以跨科整合的方式融入學習，運算思維（Computer Thinking, CT）是一個可以為未來幾年的學習者做準備的目標（García-Peñalvo & Mendes, 2018; Huyen & Nga, 2003; Smith et al., 2013），大多數 CT 教學活動是採用專題導向學習、問題導向學習、合作學習和遊戲式學習（Hsu, Chang, & Hung, 2018）。過去二十年中，遊戲式學習的環境已經發展成為強大的學習工具，遊戲式評估設計的附加作業也開始形成，且引起相關教育領域相當大的興趣（Groff, 2018），遊戲式學習最近使用的工具是透過提供有趣的場景和多媒體環境讓學生參與情境活動，可幫助學習者提升興趣和學習動機，研究結果也證明了將遊戲融入英語學習可以比非遊戲方法獲得更好的學習成果和學習動機（Liu & Chu, 2010; Malliarakis, Satratzemi, & Xinogalos, 2017），然而，將互動學習融入遊戲式學習的教學已逐漸崛起，互動學習在英語教學中提供學習者一個充滿溝通且形成解決談話能力的技能，談話能力被定義為現在教育組成能力的一個重要關鍵結果，包括語言、談話、社會語言學和社會文化能力，學習者的合作與容忍是社會與個人互動相關的組成要件（Razak, Saeed, & Ahmad, 2013）。

基於上述提到的合作、互動行為和遊戲式學習，由於合作學習僅僅提到透過合作可以增加學習者對學習的動機，而在提升學習者的學習成效中，互動學習融入遊戲式學習不僅可以提升學習者在學習上的動機還可以釐清學習上所遇到的問題及理解其中的知識。本研究將探討並調查以下研究問題：

1、利用運算思維桌遊與英語互動融入遊戲式學習的方法，提升學習者在英語學習與運算思維上的學習成效為何？

2、利用教育機器人整合運算思維與英語互動融入遊戲式學習的方法，提升學習者在英語學習與運算思維上的學習成效為何？

2. 文獻探討

2.1. 教育機器人與運算思維

根據科技的進步，越來越多具有社交技能的人形機器人已經應用在科學教育、特殊教育和外語教育等各個教育領域（Sisman, Gunay, & Kucuk, 2018）。教育機器人（Educational Robotics, ER）是以學術成就和科學概念聞名（Di Lieto et al., 2017），目前使用機器人的原因是因為教師與學生對機器人的印象。機器人技術是一種被視為可以創造許多科學教育方法的工具，例如探

究式學習及解決問題（Altin & Pedaste, 2013）。機器人在教學策略上是一種有效的活動，可以提高機器人技術的興趣、提高機器人教學的自我效能，培養學習者對科學概念的理解和促進運算思維（Computational Thinking, CT）的發展（Jaipal-Jamani & Angeli, 2017）。

在數位時代中，每個人都應該具備運算思維能力，運算思維可以被簡單定義為是一種能夠使用電腦解決生活問題的概念（Concept）、實作（Practice）和觀感（Perspectives）（Korkmaz, Çakir, & Özden, 2017），運算思維（CT）主要是一種思考和行動方法，可以透過特定技能展示成為 CT 技能的評估基礎（Shute, Sun, & Asbell-Clarke, 2017）。運算思維（CT）是評估教育的因素，也是學習者的基本技能（Zhong, Wang, Chen, & Li, 2016），運算思維（CT）是目前教育正在專注的思維，需要由新一輩的學習者在充滿數位軟體學習的軟體項目中成長，做出一套解決問題的技能（Román-González, Pérez-González, & Jiménez-Fernández, 2017）。

2.2. 遊戲式學習

隨著新流行的科技應用得出現，教育領域開始探索如何將在遊戲結合到教學和提供學習者學習（Godwin-Jones, 2016），許多研究者發現了這個現象，開始研究將遊戲融入教育當中，發現遊戲式學習（Game-Based Learning, GBL）中的遊戲功能及遊戲趨勢，是可能影響參與和學習的外部因素（Abdul Jabbar & Felicia, 2015）。

Rawendy, Ying, Arifin, & Rosalin （2017）認為學習結合遊戲可以避免學習者對學習感到枯燥，因為學習者可以藉此獲得新的學習環境，將遊戲式學習整合入教育環境，可以幫助學習者將遊戲中所學習到的知識融入教師所教授的知識（Barzilai & Blau, 2014），在設計教育遊戲方面，學習者可以透過遊戲的挑戰增加他們的學習和能力，所以遊戲式學習環境是可以在教育領域中進行（Hamari et al., 2016）。

3. 研究方法

3.1. 整合運算思維與英語互動遊戲式學習

本研究實驗組學習者使用教育機器人，使學習者體驗利用合作學習進行英語互動的練習，並透過掃描卡牌使機器人移動，在機器人移動的過程中同時驗證其運算思維的邏輯是否正確；對照組學習者使用運算思維桌遊（如圖 1），桌遊名稱為 Robot City，使學習者體驗利用合作學習進行英語互動的練習及移動角色至相對應位置並驗證雙方運算思維的整合型遊戲。

圖 1 運算思維桌遊：透過卡牌指令移動角色至相對應位置

3.2. 研究對象

本次實驗對象為台灣北部某國小三年級的學習者，實驗組實驗對象為 36 名尚未玩過運算思維桌遊的學習者利用教育機器人進行組內合作學習，對照組的部分為整個班級 28 名學習者利用運算思維桌遊進行組內合作學習，主要探討利用教育機器人整合運算思維融入遊戲式學習的方式，達到學習者在學習英語上的成效。

3.3. 評估工具

在這項研究中，測量工具為前、後測，測驗卷及量表，前後測皆為教學經驗豐富的教師設計而成。英語測驗涵蓋選擇題及配分題，總分 60 分，運算思維為 10 題選擇題，總分 40 分，測驗卷測量學習者的英語及運算思維基本知識。

運算思維量表使用的是 Korkmaz et al.（2017）開發的運算思維量表，它由 5 點李克特量表組成五個部分共 29 題：創造力（8 個項目）、演算法思維（6 個項目）、合作（4 個項目）、批判思考（5 個項目）、問題解決（6 個項目），量表 Cronbach 的總 a 值為 0.822。

最後再分別進行不同形式的教學後，採相同難易度的測驗卷題目對學習者進行測試，接著透過分析評估學習者在英語學習與運算思維上是否達到提升學習成績的成效。

3.4. 實驗流程

此實驗是在北部某國小三年級的課程上進行的，皆在評估學習者透過整合運算思維與英語互動融入遊戲式學習是否可以提升學習者在運算思維及英語上的成效。

如圖 2 表示，在實驗之前，學習者們進行前測，評估他們運算思維及英語的基本能力，同時測量他們的運算思維，此過程持續 40 分鐘。接著將兩組分開採用不同的教學活動進行實驗，此過程持續 60 分鐘，學習活動結束後，要求學習者再次填寫相同難易度的測驗卷當作後測及再次測量學習者在經過教學活動後的運算思維。

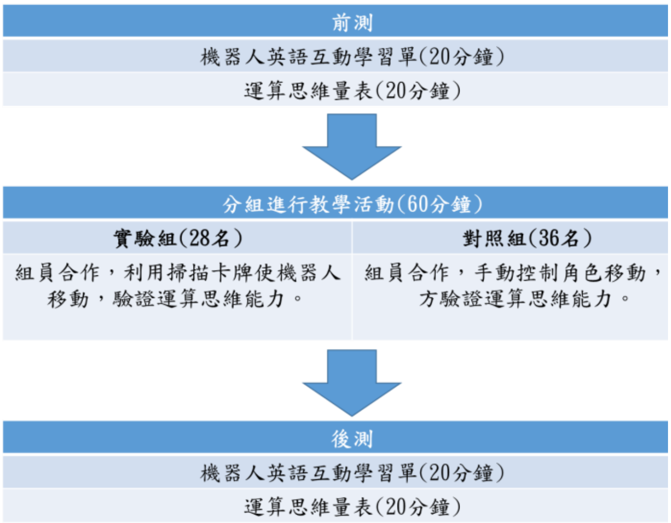


圖 2 評估活動的實驗流程

4. 研究結果

4.1. 整合運算思維桌遊與英語互動成效

本研究使用學習者的成對樣本 t 檢定評估學習者的學習成效，如*表 1* 所示，運算思維桌遊之整體學習成效是顯著提升的（$p<.05$），同時*表 1* 顯示，運算思維之學習成效與英語之學習成效都有顯著提升，這表示結合英語口語互動來使用運算思維桌遊不僅可以幫助學習者提升英語的學習成效，也可以同時提升運算思維的學習成效。

表 1 運算思維桌遊成對樣本 t 檢定

	t	df
整體（前）-整體（後）	-3.76^{**}	26
英語（前）-英語（後）	-2.92^{**}	26
運算思維（前）-運算思維（後）	-3.31^{**}	26

$^{**}p<.01$

4.2. 整合教育機器人運算思維桌遊與英語互動成效

如*表 2* 所示，教育機器人之整體學習成效是顯著提升的（$p<.05$），這表示教育機器人不僅可以提升學習者的英語能力，同時也可以提升學習者運算思維的能力。

表 2 教育機器人成對樣本 t 檢定

	t	df
整體（前）-整體（後）	-8.58^{***}	35
英語（前）-英語（後）	-6.66^{***}	35
運算思維（前）-運算思維（後）	-7.49^{***}	35

$^{***}p<.001$

4.3. 運算思維桌遊與教育機器人桌遊之成效比較

透過*表 3* 顯示，實驗組與對照組的運算思維前測不顯著（$p>0.05$），運算思維的後測有顯著差別（$p<0.05$），這表示實驗組與對照組在運算思維前測的測驗沒有差別，透過整合運算思維與英語互動融入遊戲式學習，教育機器人的桌遊成效顯著高於一般運算思維桌遊之運算思維學習成效。

表 3 運算思維學習之獨立樣本 t 檢定

	組別	N	Mean	SD	t
運算思維（前）	實驗組	36	13.44	8.88	1.60
	對照組	27	9.78	9.10	
運算思維（後）	實驗組	36	25.56	8.93	3.37^{**}
	對照組	27	17.33	12.79	

$^{**}p<.01$

有關學生英語的學習成效，本研究先對組組內迴歸係數同質性檢定，考驗結果 F=0.364，p >.05，未達顯著水準。從受試者間效應項檢定，也發現組別和前測沒有顯著交互作用（F=.077，P>.05），未達顯著水準。因此符合共變數分析中迴歸係數同質性之假設，得以繼續進行共變數分析。根據*表 4* 共變數分析結果（F=4.94*，p <.05）達顯著水準，顯示在排除前測成績的影響後，學習者在實驗組中使用教育機器人桌遊學習，其英語學習成效顯著優於使用桌遊學習的控制組。

表 4 實驗組與對照組英語學習成效共變數分析比較

組別	N	Mean	SD	Adjusted Mean	SE	F
實驗組	36	42.17	16.60	42.25	2.40	4.94^{*}
控制組	27	34.22	18.44	34.11	2.77	

$^{*}p<.05$

5. 結論與未來展望

近年來教育機器人已經被逐漸推廣，有學者發現學習者在使用機器人進行合作時可以產生積極的影響，也可以協助課堂教育者利用科學的策略有目的的設計機器人計畫，改善學習者對機器人體驗後的態度結果（Taylor & Baek, 2018）。在這項研究中，整合運算思維與英語互動融入遊戲式學習，讓學習者透過合作溝通的方式，提升運算思維與英語的學習成效。有學者發現學習者能夠透過遊戲式學習建立知識、並促進學習者積極參與學習同時增加他們的學習動機（Hwang, Sung, Hung, Yang, L. & Huang, 2013）。所以本研究主要是希望透過整合運算思維與英語互動融入遊戲式學習來幫助學習者提升學習上的動機，期望學習者能因為遊戲式學習提升學習成效。

有學者指出教育機器人有很大的潛力可以當作學習工具（Benitti, 2012），而研究者為了評估教育機器人整合運算思維與英語互動是否可以提升學習者在英語學習及運算思維上的成效，將合作互動學習融入遊戲式學習。實驗內容採用兩組不同的教學方式進行實驗，實驗組與對照組皆採用相同的教學模式進行教學，皆透過口說練習英語促進學習者開口說英語並記住過程中所需要的英語單子，同時利用卡牌排序使學習者可以增加他們的運算思維，最後實驗組採用掃描卡牌使機器人移動並驗證其運算思維是否正確，對照組則採

用手動角色，並由另一隊伍驗證其運算思維是否正確。

實驗結果表明，使用實驗組與對照組中的前測做成對樣本 t 檢定，結果發現沒有顯著差異。同時使用前測當做共變數，做共變異數分析，在前測同質性的部份為不顯著，可以表示兩組間在基本能力評估上是沒有差異的，而經過分組教學活動後，將兩組的後測做成對樣本 t 檢定，結果發現後測有顯著差異，實驗組及對照組在前後測的平均數皆有提升，但是實驗組在前後測的平均進步分數大於對照組的平均分數，故可以推測實驗組學習成效優於對照組。

最後雖然這兩組實驗皆為遊戲式學習，但本研究卻發現經過結果分析後，兩組的學習成效還是有些差異，故未來希望可以更加嚴謹的探討造成其中差異的因素。

6. 致謝

本研究感謝科技部研究計畫編號: MOST 105-2628-S-003-002-MY3 與 MOST 107-2511-H-003-031 出國會議補助，以及資策會、安譜國際股份有限公司和經濟部之實證產學合作案補助。

7. 參考文獻

Abdul Jabbar, A. I., & Felicia, P. (2015). Gameplay Engagement and Learning in Game-based Learning: A Systematic Review. *Review of Educational Research, 85*(4), 740-779.

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

Ardasheva, Y., Carbonneau, K. J., Roo, A. K., & Wang, Z. (2018). Relationships Among Prior Learning, Anxiety, Self-efficacy, and Science Vocabulary Learning of Middle School Students with Varied English Language Proficiency. *Learning and Individual Differences, 61*, 21-30.

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

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

Di Lieto, M. C., Inguaggiato, E., Castro, E., Cecchi, F., Cioni, G., Dell'Omo, M., & Dario, P. (2017). Educational Robotics Intervention on Executive Functions in Preschool children: A Pilot Study. *Computers in Human Behavior, 71*, 16-23.

García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the Computational Thinking Effects in Pre-university Education. In: *Elsevier*.

Gkonou, C. (2013). A Diary Study on the Causes of English Language Classroom Anxiety. *International Journal of English Studies, 13*(1), 51-68.

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

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

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

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

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

Huyen, N. T. T., & Nga, K. T. T. (2003). Learning Vocabulary through Games. *Asian EFL Journal*, *5*(4), 90-105.

Hwang, G. J., Sung, H. Y., Hung, C. M., Yang, L. H., & Huang, I. (2013). A Knowledge Engineering Approach to Developing Educational Computer Games for Improving Students' Differentiating Knowledge. *British Journal of Educational Technology, 44*(2), 183-196.

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.

Johnson, D., & Johnson, R. (2018). Cooperative Learning as the Foundation for Active Learning. In *Active Learning*: IntechOpen.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A Validity and Reliability Study of the Computational Thinking Scales (CTS). *Computers in Human Behavior, 72*, 558-569.

Liu, T.Y., & Chu, Y.L. (2010). Using Ubiquitous Games in an English Listening and Speaking Course: Impact on Learning Outcomes and Motivation. *Computers & Education, 55*(2), 630-643.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2017). CMX: the Effects of an Educational MMORPG on Learning and Teaching Computer Programming. *IEEE Transactions on Learning Technologies, 10*(2), 219-235.

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

Premo, J., Cavagnetto, A., Davis, W. B., & Brickman, P. (2018). Promoting Collaborative Classrooms: The Impacts of Interdependent Cooperative Learning on Undergraduate Interactions and Achievement. *CBE—Life Sciences Education, 17*(2), ar32.

Razak, N. A., Saeed, M., & Ahmad, Z. (2013). Adopting Social Networking Sites (SNSs) as Interactive

Communities among English Foreign Language (EFL) Learners in Writing: Opportunities and Challenges. *English Language Teaching, 6*(11), 187.

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.

Saranraj, L., & Meenakshi, K. (2016). The Influence of Anxiety in Second Language Learning: A Case Study with Reference to Engineering Students in Tamil Nadu, India. *Indian Journal of Science and Technology, 9*(42).

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

Sisman, B., Gunay, D., & Kucuk, S. (2018). Development and Validation of an Educational Robot Attitude Scale (ERAS) for Secondary School Students. *Interactive Learning Environments*, 1-12.

Smith, G. G., Li, M., Drobisz, J., Park, H.-R., Kim, D., & Smith, S. D. (2013). Play Games or Study? Computer Games in eBooks to Learn English Vocabulary. *Computers & Education, 69*, 274-286.

Sung, H.Y., Hwang, G. J., Lin, C. J., & Hong, T. W. (2017). Experiencing the Analects of Confucius: An Experiential Game-based Learning Approach to Promoting Students' Motivation and Conception of Learning. *Computers & Education, 110*, 143-153.

Taylor, K., & Baek, Y. (2018). Collaborative Robotics, More Than Just Working in Groups. *Journal of Educational Computing Research, 56*(7), 979-1004.

Tsai, Y. L., & Tsai, C. C. (2018). Digital Game-based Second-language Vocabulary Learning and Conditions of Research Designs: A Meta-analysis Study. *Computers & Education*, *125*, 345-357.

Wang, L. (2018). A Study of Students’ English Cooperative Learning Strategy in the Multimedia Environment. *Theory and Practice in Language Studies, 8*(6), 601-605.

Wu, T. T. (2018). Improving the Effectiveness of English Vocabulary Review by Integrating ARCS with Mobile Game-based Learning. *Journal of Computer Assisted Learning, 34*(3), 315-323.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Establishing Equitable Computing Programs in Informal Spaces:

Program Design, Implementation and Outcomes

Hui YANG[1]*, Chrystalla MOUZA[2], Lori POLLOCK[3]
[123] University of Delaware, Delaware
huiy@udel.edu, mouzac@udel.edu, pollock@udel.edu

ABSTRACT
This work examines the design, implementation and outcomes of a computer science (CS) informal learning program in a public library offered through a university-library partnership in the United States. The first purpose of this work focuses on dilemmas encountered by program providers when designing informal environments that focus on engaging diverse young students with CS concepts. For this study purpose, we analyzed over 80 reflection journals from program facilitators, illustrating both content and pedagogical decisions related to the design of the computing environment, along with program observations and interviews with children. Secondly, this work captures learning vignettes that exemplify children's growing understandings of computational concepts, practices and perspectives. Data collected to the second research purpose came from multiple sources, including children's computational artifacts, children's interviews and program observation fieldnotes. Findings of this study shed lights on the design, implementation and outcomes of informal computing programs for children from diverse backgrounds, as well as provide insights on understanding children's CT development outside the classroom.

KEYWORDS
computational thinking, informal learning program design, equitable practices

1. INTRODUCTION
Computational Thinking (CT) involves skills that help children analyze and solve real-world problems drawing on computer science (CS) principles (Wing, 2006). Many children, however, experience and use new technologies in their daily lives mostly as consumers while few have opportunities to become creators of computing innovations. Further, females and non-Asian minorities are under-represented in computing (Cuny, 2012). Partnerships with both formal and informal environments, where undergraduates with CS background assist local providers is one way of addressing this challenge. Libraries, in particular, are unique informal learning environments because in recent years they have reinvented themselves to offer a variety of low-tech and high-tech activities intended to improve visiting children's' computational skills (Myers, 2009). Nevertheless, research documenting the ways in which university-library partnerships help promote children' CT knowledge and skills is sparse (e.g., Bilandzic, 2016; Kafai et al., 2008; Myers, 2009).

This work is situated in a larger effort to improve the teaching of CS through a three- pronged approach in the United States: teacher professional development, a college field-experience course, and sustainable partnerships with formal/informal spaces. In this paper, we focus on the latter two strategies. The field-experience course combines college classroom meetings with field- experience in formal or informal settings. College meetings focus on identifying CS teaching resources, modeling CS classroom lessons, discussing CS pedagogy, and reflecting on the field-experiences. In the field, groups of undergraduates meet with educators to plan lessons, lead classroom activities, and facilitate programming events. In this work, we examine one such partnership between undergraduates and library staff members in a Scratch Technology Club ("STC") in a public library for three semesters.

Each semester, at least two undergraduates with computing background served as the STC program facilitators. These undergraduates worked closely with faculty in education and CS, as well as librarians, through weekly meetings to design and implement the STC program activities following equitable practices identified in learning sciences and CS education literature (e.g., Shah et al., 2013).

The STC was offered on Saturday mornings for 2 hours over a ten-week period each semester in three consecutive semesters. Any child interested in participating was permitted to attend. A total of 80 children between ages 7 - 15 attended the STC at least one semester. The ratio between male and female participants was about 1:1. On average, 5 - 14 children participated in each STC session. Most children had no prior experience with Scratch. Figure 1 shows the number of participants who attended the STC each semester.

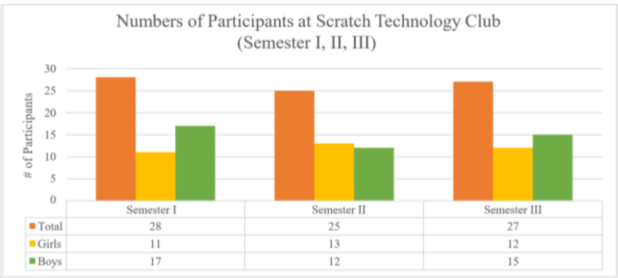


Figure 1. Numbers of Participants at STC.

For the purpose of this work, we examine the design of the program, focusing specifically on dilemmas encountered by facilitators, as well as their decisions on addressing these dilemmas when implementing the program. Additionally, we present learning vignettes to examine the role of the STC in fostering children's learning of fundamental CT concepts, practices and perspectives.

2. METHODS

2.1. Participants

Program facilitators included 7 undergraduates who were responsible for the design and implementation of the program in three consecutive semesters (Figure 2).

	Facilitator	Gender	Major/Year
Semester I			
	Ted	M	CS/2nd
	Dan	M	CS/2nd
Semester II			
	Justin	M	CS/2nd
	Sophie	F	Ed Tech/2nd
	Yvette	F	Ed Tech/2nd
Semester III			
	Jan	F	CS/2nd
	Aaron	M	CS/2nd

Figure 2. Program Facilitators' Background.

In addition, we selected 11 children participants (Figure 3) based on the following criteria: (i) regular participation in the STC, (ii) different levels of programming experience, and (iii) gender/ethnic diversity.

Participant	Age	Gender	Ethnicity	Prior Experience on Scratch	Attended Semesters	Notes
Becky	9	F	White	Limited	Semester I	-
Melissa	8	F	Turkish	No	Semester I	ELLs
Alex	9	M	Asian American	Yes	Semester I; Semester II; Semester III	-
Anna	9	F	White	Limited	Semester II; Semester III	-
Lily	10	F	White	Yes	Semester II; Semester III	-
Jim	9	M	White	No	Semester II; Semester III	-
Tim	8	M	Asian	No	Semester III	ELLs
Ella	9	F	White	No	Semester III	-
John	15	M	White	No	Semester III	Homeschool
Sophia	9	F	Asian	No	Semester III	ELLs
Kate	10	F	Asian	No	Semester III	ELLs

Figure 3. Children Participant Demographics.

2.2. Data Sources

Data were collected from multiple sources including: (a) weekly reflection journals completed by program facilitators on content and pedagogical decisions (N=80); (b) collection of children's Scratch projects at different stages of their participation in STC (N=22); and (c) interviews with children on their experience at the STC (N=11). Additionally, detailed field observations of STC sessions were collected to provide an accurate description of the activities.

2.3. Data Analyses

Program facilitators' weekly journals were analyzed using the constant comparative method to identify dilemmas faced by instructors and decisions to address those dilemmas (Hatch, 2002). Through this analysis a coding scheme was developed. Figure 4 shows an example.

Coding Categories	Sub-Categories	Definitions	Excerpts
Dilemmas	Diverse Learners	Refer to learners' diverse background with programming, skills, interests and so forth.	"*The girls chose to look at the games that seemed more feminine like a sweet donut maker that you would see in a commercial like EZ-Bake, While the boys went to see Minecraft and Pokémon clones.*" – (Tim, Semester I)
Decisions	Addressing Personal Factors	Decisions relate to personal characteristics which devote for a successful learning experience, such as prior knowledge, background, expectations, motivations and so forth	"*We used Scratch to code Finch Robots and learn some advanced features. We took what they learned last week, and implemented it this week with more advanced concepts. Instead of going through a maze, we went through an obstacle course, and had to use sensing.*" – Jan (Semester III)

Figure 4. Coding Scheme and Excerpts Example.

Interview data were analyzed qualitatively using a combination of a priori themes related to the study's questions and themes that emerged during the interviews. The six themes included in the coding scheme are: a) background information; b) motivations and interest; c) surprises; d) enjoyable learning experience, e) challenges and f) reflections. Additionally, field notes and other qualitative data were analyzed and coded into two categories, CT practices and CT perspectives. CT practices refer to how children learn about CT knowledge and skills which included "testing and debugging", while CT perspectives refer to children' reflections or attitudes towards computing (Brennan & Resnick, 2012).

Children's Scratch projects were analyzed through an automatic analytical system called "Dr. Scratch". Dr. Scratch automatically analyzes Scratch projects and produces scores in seven domains related to programming: a) Flow Control; b) Data Representation; c) Abstraction; d) User Interactivity; e) Synchronization; f) Parallelism and g) Logic. Dr. Scratch also provides an overall score, ranging from 0 to 21, and assigns the project a level ranging from "Basic", "Developing" and "Master". "Basic" (scored between 0 to 7) means that the project uses introductory programming features; "Developing" (scored between 8 to 14) means the project includes intermediate programming functions; and "Master" (scored between 15 to 21) means the project was at the developed staged with advanced programming features (Moreno-León et al., 2015).

3. FINDINGS

3.1. Dilemmas of the Design of Informal Learning Environment (Program Design)

Four types of dilemmas were discussed by program facilitators as they considered the design of the STC. The first dilemma focused on how to design a learning environment that helped all children, independent of their background knowledge develop CT knowledge and skills. Participating children's backgrounds were diverse, including their prior knowledge in programming, ages, interests, and goals. As Ted, one of the two undergraduates facilitating the STC in Semester I noticed after the second week of the club: "*the girls chose to look at the games that seemed more feminine like a sweet donut maker that you would see in a commercial like EZ- Bake, while the boys went to see Minecraft and Pokémon clones.*"

The second dilemma focused on participation rates among children, which varied from week to week anywhere from 1 to 10. For instance, at any given week, the facilitators were not aware whether the participating children would attend or not: “*We were ready to start our Scratch Club in the library. However, no kid came to the club*” (Yvette, Semester II). Moreover, every week new children joined: “*There were 13 kids who came to the Scratch Club today, some of them came for the first time*” (Yvette, Semester II). This transitional participation made it difficult for facilitators to plan activities and prepare equipment to meet the needs of children.

The third dilemma related to the resources provided by the library. In fact, the space and resources from the library were limited, and therefore planning based on number of participants attending was important. Thus, the fourth dilemma is the culmination of the first three which is about how to balance and maximize the effects of the numbers of children and the resources to design a learning environment that engages children develop CT knowledge and skills in short segments of time.

3.2. Addressing Dilemmas in the Design of Informal Learning Environment (Program Implementation)

When Ted and Dan first started to design the STC in Spring 2016, they sought out advice from the university faculty leading the field experience course and the librarians. Ted and Dan decided to have children freely explore Scratch and design a project of interest to them. After their first try, however, they recognized that free exploration without some initial guidance proved challenging to children. As a result, they re-defined “free-choice” deciding to first provide children with some foundational skills on navigating Scratch and subsequently offering “free-choice” on what to program. Once children acquired a basic understanding of Scratch they again introduced free choice. In the following semesters, other facilitators followed the same format of instruction; providing initial instruction and subsequently just scaffolding as children created computational artifacts of interest to them.

Throughout the program, the roles of the facilitators also shifted from instructors to facilitators. Their decisions, which included both content and pedagogical considerations, were based on three main factors: personal, socio-cultural and physical factors (Falk & Storksdieck, 2005).

3.2.1. Addressing personal factors to make learning CS approachable and engaging

All facilitators collected children’s feedback on what they wanted to learn either through observations or conversations. In Semester III, as Jan and Aaron took over the club, they firstly took action to understand the children’s needs and interests: “*we had the children get started on their stories, while I went around the room with my notebook, asking them individually what concepts they want to learn, or need to brush up on*”. Moreover, the facilitators would modify their planning based on children’s engagement and feedback from the previous week.

Considering most children did not have prior knowledge in computing, facilitators linked CS concepts through an engaging way. They provided children with knowledge and skills to construct personal meaningful artifacts and helped them establish a linkage between CS concept and its applications. Figure 5 illustrates designed CT activities provided by the program facilitators throughout three semesters that demonstrate connections between CS, children’s interests and daily life.

Week	Semester I	Semester II	Semester III
1	Exploring Scratch	Introduction to Scratch-- *conditional statement* CS Unplugged Activity: Harold the Robot	Introduction to Computer Science
2	Introduction to Scratch (Basic Scratch Concepts on Sprite, Scripts etc.)	Simple Animation on Scratch -- *x-y coordinate; operator; sensing*.	Programming on Scratch – Storyboarding Strategy
3	Learn To Code	Scratch & Math (I) – *loop function; pen*. CS Unplugged Activities: Image Representation	Creating your own Volleyball Game
4	Blocks on Loops	Making Scratch Game: Underwater Love	Introduction to 3D Printing – Basic
5	Variables and Operators	Scratch & Math (II) CS Unplugged Activity: Variables	Learning more about 3D Printing – Intermediate
6	CS unplugged and Problem Solving	Making Scratch Game: Pong Game	Robotics – Finch Bots – Introductory
7	Conditionals and Loops Creation of Individual Project	Finch Robots (I) – *movement; sounds.*	Robotics – Finch Bots – More Features
8	Creativity -- Music Instrument Creation of Individual Project	Programming Technique with Finch Robots (II) – *loop function; variables*.	Creative Programming with Makey Makey
9	Motion Sensing Blocks Creation of Individual Project	Makey Makey: Create your piano with bananas Creating Scratch Project	Advanced Features in Scratch – Cloning and more
10	Creation of Individual Project Project Presentation	Arduino Scratch Activity -- *Broadcasting* Scratch Project Presentation	Making your final project with Scratch/ Demo Day

Figure 5. Implemented CT Activities at Scratch Technology Club.

3.2.2. Addressing sociocultural factors to promote an interactive and collaborative dynamics

Since attendees at the STC came from diverse backgrounds, the program facilitators aimed at promoting a social and collaborative environment that allowed children to communicate with peers, share personal meanings, and construct learning together. To accomplish these goals, facilitators employed different approaches, including peer-programing. Moreover, children were observed to bring new friends or family members to the club. In this social and interactive environment, children were observed to talk, share and help each other: “*a couple of our students were helping each other out more when we were both in one- on-one sessions and the kids seem to listen to each other a lot. They also like to show off to their pals or new friends. It seems that even if they had no idea what scratch was before, they are still enjoying when in the company of others*” (Justin, Semester II).

3.2.3. Addressing physical factors to strengthen an effective learning environment

Program facilitators often needed to rearrange the physical settings to ensure that children have enough space to test their artifacts: “*We have to move the tables at convenient positions to allow maximum room for the robot's movement; set up individual laptops, chargers, and mouse; tether the Finch robot and launch the application before the class begins*” (Sophia, Semester II).

3.3. Capturing Learning Vignettes in Informal Learning Environment (Program Outcomes)

3.3.1. CT Concepts

Children demonstrated growth on CT concepts throughout their participation in the STC as illustrated by Tim's case (Figure 6). Tim entered the STC without any programming experience and with limited English language skills. His early project used only foundational blocks (left). In contrast, the project he created at the end of the program demonstrated a variety of computational concepts including synchronization and logic (right).

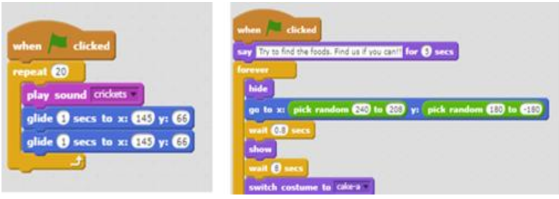


Figure 6. Tim's Midterm Project (Left) and Final Project (Right).

3.3.2. CT Practices

Interview and observation data revealed that children developed computational practices, particularly around problem-solving and ideation. Additionally, interview data indicated that participants were engaged in the process and development of personal meaningful artifacts, including computer games and robotics. One of the children who came to the STC without any prior experience in programming notably summarized his semester-long learning experience: "*I didn't really know how to use Scratch before but when I come to the Scratch club, I know how to do it. I just know many things about it, I can create many things that can be used (with) Scratch.*"

3.3.3. CT Perspectives

Observation notes and other qualitative data indicated that children developed CT concepts and practices through a social learning environment with more access to new people and resources. As Anna said: "*I enjoyed making things with my friend*." Moreover, children also liked to learn through access to other resources. Lily commented: *"(Learning Scratch) it was pretty fun and I enjoyed playing on other people's projects and then make my own.*"

4. SIGNIFICANCE

In this paper, we addressed an issue that received little attention in the literature, related to the design of informal learning environments that focus on engaging young students with CS concepts (Maloney et al., 2008). We focused on providing evidence and analysis on how program facilitators, with support from university faculty and librarians, regulated and adapted the design of the STC. The STC provided opportunities to children aged between 7 - 15 from different cultural backgrounds to explore CT knowledge and practices. The program facilitators considered the personal, sociocultural and physical factors as they decided on how to better facilitate children's free exploration of CS through Scratch programming. Throughout the program, children were found to be interested and confident in exploring resources about programming through social interactions with their peers. To become well-educated citizens in the 21st century, children must develop a deeper understanding of the fundamentals of CS and develop analytical CT skills (Wilson et al., 2010; Wing, 2006). Findings of this study provided insights related to the design, implementation and outcomes of informal computing programs for children from diverse backgrounds.

5. REFERENCES

Bilandzic, M. (2016). Connected Learning in the Library as a Product of Hacking, Making, Social Diversity and Messiness. *Interactive Learning Environments, 24*(1), 158-177.

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.* AERA, 1-25.

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

Falk, J., & Storksdieck, M. (2005). Using the Contextual Model of Learning to Understand Visitor Learning from a Science Center Exhibition. *Science Education*, *89*(5), 744-778.

Hatch, J. A. (2002). *Doing Qualitative Research in Education Settings.* New York: Suny University Press.

Kafai, Y. B., Desai, S., Peppler, K. A., Chiu, G. M., & Moya, J. (2008). Mentoring Partnerships in a Community Technology Center: A Constructionist Approach for Fostering Equitable Service Learning. *Mentoring & Tutoring: Partnership in Learning, 16*(2), 191-205.

Maloney, J., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by Choice: Urban Youth Learning Programming with Scratch. In *39th SIGCSE Technical Symposium on Computer Science Education*. ACM, 367-371.

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.

Myers, B. (2009). Imagine, Invent, Program, Share: A Library-hosted Computer Club Promotes 21st Century Skills. *Computers in Libraries*, *29*(3), 6.

Shah, N., Lewis, C. M., Caires, R., Khan, N., Qureshi, A., Ehsanipour, D., & Gupta, N. (2013). Building Equitable Computer Science Classrooms: Elements of a Teaching Approach. Proceedings of the *44th ACM Technical Symposium on Computer Science Education*. ACM, 263-268.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Implementing Computational Thinking through Non-formal Learning in after School Activities at Students Society Club

Poh-tin LEE[1*], Xin-rui LEE[2], Chee-wah LOW[3], Athinamilagi KOKILA[4]
[1234] Bukit View Secondary School, Singapore
lee_poh_tin@moe.edu.sg, lee_xin_rui@moe.edu.sg, low_chee_wah@moe.edu.sg, athinamilagi_kokila@moe.edu.sg

ABSTRACT

This paper shares a secondary school's implementation for students to acquire Computational Thinking skills through non-formal learning in after school activities at students society club. These activities are also known as school-based Co-Curricular Activities (CCAs) under the Singapore education system. In the school, a team of teachers manage the Infocomm Club where students develop coding competency in information and communications technology through non-formal learning beyond the school curriculum.

KEYWORDS

non-formal learning, coding, computational thinking, co-curricular activities (CCA), implementation.

1. INTRODUCTION

Under the Singapore education system, students spend between 4 to 5 years in the secondary school. Students are admitted to the Year 1 of the secondary school when they are 12 to 13 years old. It is mandatory for these students to select Co-Curricular Activities (CCA) from one of the following categories: Uniform Groups, Sports & Games, Performing Arts and Society Clubs.

At Bukit View Secondary School, the Infocomm Club is made available to the students under the Society Clubs category. The club aims to excite students about the possibilities of Information and Communications through a structured curriculum with Computational Thinking skills (Wing, 2006) and expand students' creative and entrepreneurial spirit. The students spend 3 hours per week at the Infocomm Club.

2. TRAINING FRAMEWORK OF THE INFOCOMM CLUB

The teachers of the Infocomm Club at our school have designed a training framework for the students to be developed in 3 areas: software, hardware and heart-ware.

Figure 1 shows the skills which the Infocomm Club students will acquire at the end of their Year 4.

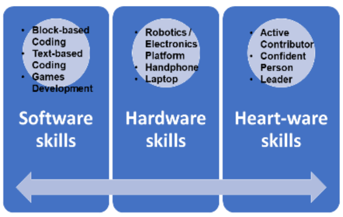


Figure 1. Infocomm Club Training Framework.

Under this framework, the Infocomm Club students are taught software skills in both blocked-based coding and text-based coding. The student will learn Scratch programming (Maloney et al., 2010), Python programming (Rashed & Ahsan, 2012) and MIT's App Inventor (Wagner et al, 2013). They also learn hardware skills on robotics / electronics platform such as the Arduino and Raspberry Pi boards.

3. IMPLEMENTATION OF THE INFOCOMM CLUB CURRICULUM

The students of Infocomm Club go through a rigorous 4-year curriculum in after school activities focusing on the following areas: Robotics, Apps Development, Games Development and Network Security.

Table 1 shows the implementation of the Infocomm Club Curriculum with the various activities.

Table 1. Infocomm Club Curriculum.

Area of focus	Description of activities
Robotics	Development of applications on electronics platform such as Arduino or Raspberry Pi boards capable of reading inputs from sensors and turning on devices such as motors and LEDs.
Apps Development	Building of mobile applications with MIT's App Inventor on Android phones.
Games Development	Conceptualisation of game, storyboarding, prototyping and game creation in Scratch or Python programming.
Network Security	Cyber security, encryption / decryption algorithms and cryptanalysis techniques.

4. HOLISTIC DEVELOPMENT OF STUDENTS

In addition to acquiring software and hardware skills, the Infocomm Club students also acquire heart-ware skills with the following outcomes:

a. *Active Contributor:* The students are involved in community service work such as providing computer training to senior citizens in the neighborhood as well as sharing on cyber wellness regarding online behaviour and risks in the cyberspace.

b. *Confident Person:* The students gain confidence through participation in Infocomm competitions where they develop creative ideas, give presentations to judging panels and hone their public speaking skills.

c. *Leader*: The upper secondary students have the opportunity to coach and mentor the junior members in both technology and presentation skills.

The Infocomm Club teachers identify various competitions for the students to participate. Through these competitions, students are developed in higher level of software and hardware skills. Our students have performed well in these competitions and won many awards. Table 2 shows the students achievements in the year 2018.

Table 2. Infocomm Club Achievements at key competitions in the year 2018.

Infocomm Club Competitions	Student Achievements
RoboCup Junior Cospace Pei Hwa Challenge	Championship Award
National Digital Storytelling Competition	Championship Award
IDE Arduino Maker Competition	Engineering Award
Singapore Games Creation Competition	Finalist Award
iCode Competition	Finalist Award

5. SURVEY ON STUDENTS INTEREST IN CODING AND ITS EFFECTS

A survey was conducted for 27 Infocomm Club students from Year 1 to Year 3 levels in October 2018. 89% of the students have expressed they like Coding / Programming activities as shown in Figure 2. 74% of the students feel that Computational Thinking through coding helps them to develop cognitive skills and solve real-life problems (Liao & Bright, 1991) as shown in Figure 3.

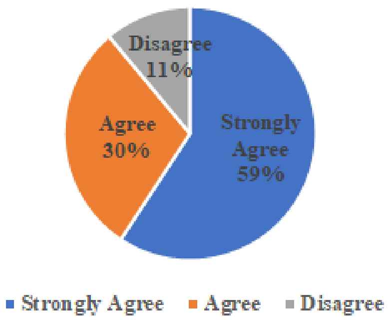


Figure 2. Survey Question: I like Coding / Programming activities.

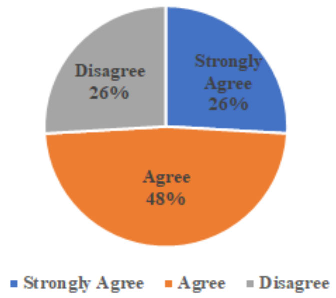


Figure 3. Survey Question: Coding helps me to develop cognitive skills and solve real-life problems.

6. FUTURE PLAN FOR THE CLUB

In the survey, the students were also asked whether they prefer to learn more on computer animation, games development or electronic platforms such as the BBC micro:bit, Raspberry Pi and Arduino. 89% of the students have indicated that they prefer to learn more on coding with games development. Hence, the teachers will explore training students in other games development platforms such as GameMaker, Unity and PyGame.

7. CONCLUSION

This paper shares the framework, curriculum and implementation of non-formal learning activities at the students society club for students to acquire Computational Thinking in a fun environment. In addition to acquiring software and hardware skills at the Infocomm Club, the students are also enriched in heart-ware skills to develop their leadership ability through participation in community projects related to information and communications technology. We hope that sharing the students experience at the Infocomm Club would provide some understanding in how schools can develop their students in Computational Thinking through non-formal learning.

8. REFERENCES

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.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*, *10*(4), 16.

Rashed, M. G., & Ahsan, R. (2012). Python in Computational Science: Applications and Possibilities. *International Journal of Computer Applications*, *46*(20), 26-30.

Wagner, A., Gray, J., Corley, J., & Wolber, D. (2013). Using App Inventor in a K-12 Summer Camp. *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*. ACM, 621-626.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Computational Thinking and Psychological Studies

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

What Underlies Computational Thinking: Exploring its Cognitive Mechanism and Educational Implications

Rina Pak-ying LAI
Faculty of Education, University of Cambridge, The United Kingdom
pyrl3@cam.ac.uk

ABSTRACT

The effectiveness of a pedagogical approach is directly related to whether it takes the characteristics of cognition into account (Sweller 2007). As such, a fundamental question that has strong implications for effective teaching and learning—in the context of computational thinking (CT)—is what is its nature. Is it a domain-specific or domain-general cognitive capacity? The main objective of this article is to provide an initial introduction of generality/specificity issues to the CT literature from a cognitive psychology perspective. Specifically, it focuses on a critical appraisal of the domain-specific account of CT, which is subscribed by a considerable amount of existing conceptualisations and teaching practices. The article discusses educational implications under this account of CT, provides suggestions informed by learning theories, and presents a learning model for CT education.

KEYWORDS

computational thinking, cognitive psychology, computer science education, learning model, cognitive mechanism

1. INTRODUCTION

In the past decade, interest in computational thinking (CT) has been burgeoning in both research and education sectors. Despite the infancy of the field, there is a significant amount of studies investigating the association between CT and coding/programming (e.g., Brennan and Resnick, 2012; Kong 2016) as well as related disciplines (Hambrusch et al., 2009; Kanaki and Kalogiannakis, 2018). As Voogt et al. (2015) note, the predominant focus in these specific areas can lead to an unwarranted bias in the question of whether CT equates to or is a prerequisite to programming. Yaşar (2018) further elaborates that the entanglement between CT, programming, and related educational tools may shift researchers' attention away from investigating the cognitive underpinnings of CT. Hence, there is a conceptual and theoretical reason to distinguish its core characteristics from its correlates, or peripheral skills, as the overemphasis on the latter could lead to more ambiguities regarding its nature and reduce the unity of CT. Given that the current state of research has little knowledge regarding its cognitive characteristics, pedagogical decisions become a difficult task.

Indeed, Wing (2008) suggests that an important challenge for learning and teaching—when including CT in the repertoire of thinking abilities—is deciding on how and when learners should be taught. Implicit in this concern is the question of how CT operates at a cognitive level, which is currently unknown. Merging these research gaps identified in the literature, a ramification on a fundamental question regarding the underlying mechanism of CT is whether it is a domain-specific (e.g., programming) or domain-general capacity (e.g., general problem-solving). The implications of this question bring us to the heart of efficient pedagogy, curriculum design, and guidance for future research. However, addressing these issues requires bridging cognitive/psychological theories and CT research, which is yet to be fulfilled. This paper aims to fill in this gap by providing an initial account of generality/specificity mechanism from cognitive psychology into the CT literature.

2. DOMAIN-SPECIFIC AND DOMAIN-GENERAL CT

In the tradition of cognitive psychological studies, domain-specific and domain-general mechanisms have been used to theorise processes that underlie learning, reasoning, and knowledge. For example, they have been used to understand problem-solving, statistical learning, and scientific reasoning. Consistent with the work of Rakison and Yermolayeva (2011), *domain-specific* mechanisms are processes that target learning in a particular area of knowledge that may pose certain constraints specific to that area, such as computer science. In contrast, *domain-general* mechanisms are processes that are not associated with a particular area of knowledge. That is, mechanisms that are "knowledge and modality universal" (Rakison & Yermolayeva, 2011), context-independent, and may be adapted to diverse areas. A domain-general CT would be perceived as cognitive processes useful for problem-solving across subjects, as suggested by Wing (2008), Yadav, Hong, and Stephenson (2016), as well as Labusch (2018), among others.

CT, as cognitive capacity, can be represented by and promoted as a domain-specific or domain-general construct. However, given the lack of empirical evidence, the aim of this article is not to propose for a definite mechanism that explains the nature of CT. Rather, its objective is to explore, in particular, the domain-specific account of CT that the majority of the existing conceptualisations and education practices subscribe to. It should be noted that there are emerging views that endorse a domain-general account, viewing CT as an approach to problem-solving without relating it to a specific area such as computer science, its related skills or tools. However, domain-specificity is focused in this article as it has larger influences on current educational policies and practices (Bocconi et al., 2016).

3. DOMAIN-SPECIFIC ACCOUNT OF CT

The domain-specific account of CT conceptualises it as a skill that is particularly important to computer science and often requires the support of technology. Indeed, many early computer science scholars and pioneers perceived CT as an important skill set for individuals who design and execute computations with the support of computers (Tedre and Denning, 2016). One important figure is Seymour Papert, the co-developer of the LOGO programming language and educational robot, Turtle, who purportedly coined CT. Influenced by constructionism, Papert's notion of CT situates in a context that emphasises learner-computer interactions as well as the thinking skills developed through such relations. He speculates that children could be manipulators of computers and that interacting with machinery is crucial to the enhancement of thinking skills (Papert, 1980). This idea is extended to include the importance of engaging with artefacts, tools, and media. Through these, learners are not only prompted to think computationally but to also generate ideas and construct meaningful products (Harel & Papert, 1991).

Papert's early ideas have a significant influence on the modern understanding of CT, with a particular focus on learner-computer interactions, computer science concepts, and programming. Indeed, extending from his perspectives, some researchers view CT in a context-dependent manner. For example, a well-referenced theoretical framework proposed by Brennan and Resnick (2012) identifies three dimensions of CT, consisting of CT concepts, practices, and perspectives—all of which are embedded in programming languages and artefacts. While CT concepts refer to programming concepts (e.g., conditionals), CT practices point to processes learners develop as they program (e.g., abstraction), and CT perspectives are how learners relate to the technological world (e.g., questioning) (Brennan & Resnick, 2012). Likewise, considering "design thinking components" (e.g., exploration, empathy, and creation) and hardware aspect of CT solutions (e.g., robotics), other researchers define CT as cognitive tools for creative programming and digital literacy (Romero, Lepage, and Lille, 2017). Other organisations such as The International Society for Technology in Education (ISTE, 2011) highlights the importance of formulating problems such that a computer and other tools can help. As such, there is a supposition that CT cannot be divorced from the context of computer science or technology.

Educational practices that emerge from the domain-specific account of CT tend to promote it through programming, with the hope to develop what Yaşar (2018) refers to as "electronic CT skills". While recent proposed curriculum frameworks consider general problem-solving and real-world problems to some extent (Kong, 2016; Angeli et al., 2016), current educational policies remain predominantly focused on teaching CT through coding/ programming lessons with an emphasis on artefacts (de Paula et al., 2018). This is evident at an international level. To name a few: The English National Curriculum for Computing Education (2013) established the priority to prepare learners to use CT to understand and contribute to their environments; Italy aims to teach programming as a means to introduce CT to learners; Hong Kong has launched a 4-year coding program to teach CT in pilot schools; and Singapore has introduced CT-based coding enrichment lessons for learners. Emerging from these initiatives are applications that support the creation of artefacts in the classrooms through child-friendly graphical programming languages (e.g., Scratch, Alice, Snap, Blockly) and robotic kits (e.g., Mindstorm, LEGO® WeDo® 2.0).

4. RETHINKING THE DOMAIN-SPECIFIC LEARNING APPROACH OF CT

4.1. Programming Languages

It is undeniable that a computing curriculum, programming activities, and educational applications support the development of CT skills and provide means for it to be expressed. However, it is less known whether the knowledge structure or schemas in these areas encompass CT entirely or merely aspects of CT skills such as programming. Ambrosio et al. (2014) define a schema as the cognitive structure that helps learners to recognise a specific class of problems which necessitate particular strategies or processes. This, in turn, contributes to the construction of mental models. Despite programming languages allowing learners to demonstrate CT skills, promoting CT through programming alone may not be sufficient enough for learners to develop a coherent mental model of CT. This model becomes crucial when learners face novel (non-programming) problems in which they need to identify the most appropriate CT process(es) so as to devise a successful solution. That is, a coherent mental model of CT should reflect conceptual clarity and is flexible to different problem contexts. Whether programming education can contribute to such a mental model of CT is an area to elucidate in future research.

On the other hand, educators following this procedure ought to question whether learning transfer is an objective. If domain-specific CT skills operate without any general conceptual knowledge or problem-solving strategies, they could be diminished to rudimentary skills that are only useful in standard or routine problems. That is, it would be, at best, procedural skills in which learners memorise steps of operations without a clear understanding of underlying meanings (Arslan, 2010). This is different from conceptual knowledge in which learners are able to carefully analyse new scenarios and generate novel ideas. For example, a learner coding an algorithm that was previously taught is different from being able to write his or her own code to tackle a new problem situation. This ability is rather important, especially in the 21st century where many new and complex problems arise in situations never encountered by learners. Yet, this ability surpasses routine problems that domain-specific CT skills can solve. In this regard, it may be of interest to investigate how domain-general CT skills could be integrated into or support the teaching of domain-specific CT skills.

4.2. Educational Robotics

On the other hand, influenced by constructionism, there is emerging popularity in promoting CT through educational tools, such as educational robotics. These tools have shown to provide rewarding learning experiences that motivate learners (Penmetcha 2012). However, overemphasising the usage of these tools without delivering fundamental conceptual knowledge alongside, the development of CT, as a thinking skill, could be undermined. Indeed, giving the example of using a calculator in contrast to understanding arithmetic, Wing (2008) cautions the possibility of learners confusing conceptual understanding with mastery of applications. Moreover, constructionism may only be successful if learners have sufficient prior content knowledge (Yaşar, 2018). Yet, the link between CT principles and robotics are often not made explicit to learners. As such, it requires learners to reason and reflect on the underlying logic and processes during a robotics activity. However, introspection is not always initiated by learners. In this regard, the conceptual understanding of CT may not be delivered to the extent that educators and policy-makers intend to. To ensure effective learning outcomes, there is a necessity to integrate and consider other aspects of CT education. To extend from the current practices, suggestions based on learning theories are given below to guide future directions in curriculum design and pedagogical decisions. A learning model is presented to summarise these suggestions.

5. FUTURE DIRECTIONS IN CT EDUCATION

5.1. A Hybrid Teaching Approach to Foster CT

Promoting domain-general CT entails the development of cognitive skills that are relevant for cross-curricular contexts, such as problem-solving strategies, information processing, self-regulation, and metacognition (Greiff et al., 2014). This is the foundation in which educators can support learning transfer in CT. On the other hand, fostering domain-specific CT could provide programming knowledge and concepts such that learners can apply CT to solve computational problems and developed relevant skills within the scope of computer science or technology-related areas. For example, the use of the computer may provide an environment for learners to apply CT effectively and creatively to solve complex network problems in computer science, biology, and physics. Despite targeting different learning goals and outcomes, the two domains are non-exclusive: domain-specific expertise can emerge from domain-general knowledge given adequate experience (Rakison and Yermolayeva, 2011). To this end, a well-designed curriculum ought to integrate both models. That is, a hybrid teaching approach.

The hybrid approach goes beyond solely integrating CT across the curriculum, which has already started to emerge in both research and practice. Rather, successful teaching ought to also consider the nature of the various components, such as abstraction, algorithmic thinking, problem-decomposition, and debugging, that CT is composed of. As Yaşar (2018) observes, abstraction and problem-decomposition are often expressed in a more generalised form than other components that may have a stronger reliance on computer science or technology. He notes that abstraction is an inductive reasoning process that simplifies, categorises and processes important information for faster retrieval. On the other hand, decomposition is a deductive reasoning process that manages complexity by solving smaller problems. Both of which are practiced in our everyday lives albeit we all possess a varying degree of understanding and usage. This suggests some components of CT may be better taught as generalised skills while others within a domain subject— with or without the support of technology. For example, the concept of abstraction can be taught in map reading, a rather generic and real-life scenario that learners can relate to. Implicit in this idea is the need for a dynamical and flexible pedagogy that encompass the nature of CT and its related components.

5.2. Teaching Toward Transfer

Transfer of learning is essential to building lifelong learners and should be an explicit goal in the promotion of CT. However, there is a paucity of studies and discussions surrounding the topic of learning transfer in CT. Transfer occurs when learning in one context influences performance in another context and is demonstrated in two ways: low-road transfer (to similar contexts, problems, and performance) and high-road transfer (to dissimilar contexts, problems, and performance) (Perkins & Salomon, 1992). To apply CT in the most effective and meaningful manner, learners should thoughtfully abstract, generalise, and apply CT principles in novel problems. These problems should share similarities in structure but differ in surface features as previously solved problems (Mestre, 2006). According to Witherspoon et al. (2017), transfer may be most efficiently accomplished through instructional and teaching approaches along with metacognitive strategies. Building upon the authors' recommendation and informed by learning theories, six suggestions are given to optimise learning transfer in CT:

Motivation: provoke learners' interests to learn and apply CT. Educational robotics and fun unplugged activities can be used to introduce basic concepts while increasing learners’ motivation.

Learn how to learn: integrate metacognitive strategies that support learning transfer in lessons

Build foundational knowledge: make explicit connections between CT principles to real-world problems that learners can relate to; use analogies and metaphors to assist teaching

Reflective learning: create an environment for transfer by having learners deliberately analyse how they can apply CT in their everyday lives and have them present their ideas

Embed CT into a specific area: allow learners to apply their CT knowledge to a specific area such as a programming environment to further develop CT skills

Use formative assessments to test transfer: focus on whether students can think conceptually and thoughtfully when applying CT in a novel and unfamiliar problem. Gives feedback regarding learners’ strengths and weaknesses to enhance learning.

5.3. Future Curriculum Design

Implied by the previous two points is a necessity to move beyond a context-bound understanding and pedagogy of CT. This includes the need to consider how to help learners construct a coherent mental model of CT as the foundation of learning. What is of interest to educators are not merely CT skills demonstrated in a rudimentary form such as programming, but of learners' insight into *how* and *when* to use CT in diverse contexts that are relevant—the ability to discern when CT is the best approach to problem-solving. This is largely dependent on learners' mental models, which could be developed through conceptual, theoretical discussions and demonstrations. For example, using worked examples to encourage exploration of similarities and patterns that commonly arise when tackling similar families of problems. While the former reduces cognitive load, the latter facilitates mindful abstraction. It is then, and only then, that domain-specific teaching approaches that integrate programming and robotic tools become good "objects to think with", as Papert (1980) puts it.

5.4. A Summary: A Three-Layer Learning Model of CT

To demonstrate the above suggestions explicitly, a three-layer learning model is illustrated in Figure 1 to represent three aspects of CT education: learning procedure, learning goals, and learning outcomes. The model comprises learning process in domain-general and domain-specific procedure, with a consideration of transition between the two. Whereas domain-general procedure aims to help learners develop a coherent mental model of CT as the foundation of learning, domain-specific procedure seeks to incorporate and contribute to the model by advancing CT skills in a specialised subject (e.g., computing education and STEM). The transition stage prepares for the success of learning transfer between the two procedures. Learning outcomes and goals are different and depend on the given procedure but together fulfil the purpose of the hybrid teaching approach.

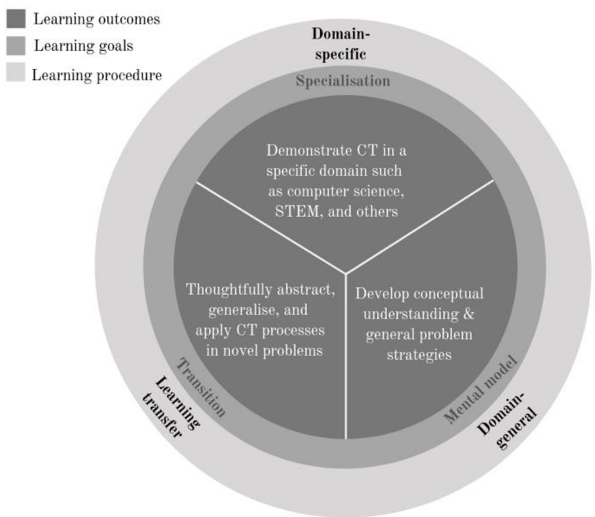


Figure 1. A Three-layer Learning Model of CT.

6. CONCLUSION

This article gives the first discussion on the generality-specificity issue of CT from a psychological perspective. It is hoped that through this a dialogue is opened for cognitive scientists, educational researchers, and curriculum developers on the cognitive mechanism that underpins CT. Addressing this issue is not only significant from a theoretical point of view, but also from a practical perspective. At a theoretical level, this article paves the path for cognitive psychology to enter CT research thereby to further elucidate on its underlying cognitive mechanisms. It is believed that such effort could continue to shed light on the nature of CT by addressing, for example: the developmental trajectory of CT; cognitive functions that support it; and ways to enhance conceptual learning of CT. In this regard, contributions from cognitive scientists and educational psychologists are appreciated and demanded. On the other hand, at a more practical level, this article calls for a revisit of the current pedagogy and curriculum design; and to argue for the importance of a hybrid teaching approach. This is a call for promoting CT in a dynamic manner such that we are developing learners' computational minds rather than rudimentary CT skills.

7. REFERENCES

Ambrosio, A. P., Almeida, L. D. S., Macedo, J., & Franco, A. (2014). Exploring Core Cognitive Skills of Computational Thinking. *Proceedings of Psychology of Programming Interest Group Annual Conference 2014*. University of Sussex, 25-34.

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. *Educational Technology & Society, 19*(3), 47-57

Arslan, S. (2010). Traditional Instruction of Differential Equations and Conceptual Learning. *Teaching Mathematics and its Applications*, *29*(2), 94-107.

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*, 1-25.

De Paula, B. H., Burn, A., Noss, R., & Valente, J. A. (2018). Playing Beowulf: Bridging Computational Thinking, Arts and Literature through Game-making. *International Journal of Child-Computer Interaction, 16,* 39-46.

Greiff, S., Wüstenberg, S., Csapó, B., Demetriou, A., Hautamäki, J., Graesser, A. C., & Martin, R. (2014). Domain-general Problem Solving Skills and Education in the 21st Century. *Educational Research Review*, *13*, 74-83.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A Multidisciplinary Approach towards Computational Thinking for Science Majors. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*. New York: ACM, 183-187.

Harel, I., & Papert, S. (Eds.). (1991). *Constructionism*. Westport, CT: Ablex Publishing.

International Society for Technology in Education (2011). *NETS for Coaches*. Retrieved January 18, 2019, from www.iste.org/standards/nets-for-coaches.aspx

Kanaki, K., & Kalogiannakis, M. (2018). Introducing Fundamental Object-oriented Programming Concepts in

Preschool Education within the Context of Physical Science Courses. *Education & Information Technologies*, *23*(6), 2673-2698.

Kong, S. C. (2016). A Framework of Curriculum Design for Computational Thinking Development in K-12 Education. *Journal of Computers in Education*, *3*(4), 377-394.

Labusch, A. (2018). Fostering Computational Thinking through Problem-solving at School. *Proceedings of the 2018 ACM Conference on International Computing Education Research*. New York: ACM, 276–277.

Mestre, J. P. (2006). *Transfer of Learning from a Modern Multidisciplinary Perspective*. North Carolina: Information Age Publishing.

Papert, S. (1980*). Mindstorms: Children, Computers, and Powerful Ideas, iv-17*. New York: Basic Books, Inc.

Rakison, D. H., & Yermolayeva, Y. (2011). How to Identify a Domain-general Learning Mechanism When You See One. *Journal of Cognition and Development*, *12*(2), 134-153.

Romero, M., Lepage, A., & Lille, B. (2017). Computational Thinking Development through Creative Programming in Higher Education. *International Journal of Educational Technology in Higher Education*, *14*(1), 42.

Perkins, D. N., & Salomon, G. (1992). Transfer of Learning. *Contribution to the International Encyclopedia of Education (2nd ed.)*. Oxford: Pergamon Press.

Sweller, J. (2012). Human Cognitive Architecture: Why Some Instructional Procedures Work and Others Do Not. In K. R. Harris, S. Graham, T. Urdan, C. B. McCormick, G. M. Sinatra, & J. Sweller (Eds.), *APA Educational Psychology Handbook, Vol. 1. Theories, Constructs, and Critical Issues*, 295-325. Washington, DC: American Psychological Association.

Tedre, M., & Denning, P. J. (2016). The Long Quest for Computational Thinking. *Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16*. Koli: ACM Press, 120-129.

Department for Education (2013). *The National Curriculum in England: Key Stages 1 and 2 Framework Document.* Retrieved January 4, 2019, from https://www.gov.uk/government/publications/national-curriculum-in-england-primary-curriculum.

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

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

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing Computational Thinking through a Virtual Robotics Programming Curriculum. *ACM Transactions on Computing Education*, *18*(1), 1-20.

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.

Yaşar, O. (2018). A New Perspective on Computational Thinking. *Communications of the ACM*, *61*, 33-39.

Computational Thinking in Educational Policy

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Implementing Computational Thinking in the Dutch Curriculum

an Exploratory Group Concept Mapping Study

Marcus SPECHT[1], Marinka COENDERS[2], Slavi STOYANOV[3]
[1] Delft University of Technology, Netherlands
[123] Open University, Netherlands
m.m.specht@tudelft.nl, Slavi.Stoyanov@ou.nl

ABSTRACT
The Dutch ministry of education aims to include the development of computational thinking skills in the core curriculum of primary and secondary education. However, in order to establish a successful implementation of computational thinking into the core curriculum it is important to gain an insight into the necessary measures for this cause. In a group concept mapping study 39 participants from different stakeholder groups identified a total of five grander measures which were deemed necessary to successfully implement CT: (1) determine the content of a CT programme, (2) guard the quality of the implementation of CT, (3) define the term ‘CT’, (4) enhance the skills of teachers in this field and, finally, (5) make sure there is enough time and money available.

KEYWORDS
Curriculum, computational thinking, implementation

1. INTRODUCTION
In an effort to prepare young students to an ever-changing, digitalized society, 21st-century-skills, such as “digital literacy”, are indispensable. Other evident skills in the digitalized society are basic ICT skills, media knowledge, information literacy and computational thinking (CT). This is why the Dutch ministry of education aims to include the development of these skills in the core curriculum of primary and secondary education. However, in order to establish a successful implementation of these new components into the core curriculum, it is important to gain an insight into the necessary measures for this cause. Therefore, this study aims to gain an overview of the necessary means for a successful, long-term implementation of CT into primary and secondary education in the Netherlands.

In order to collect and classify the measures that could be used for successful implementation of CT this study uses the group concept mapping method. This is a form of mixed-method data collection and handling which brings the shared ideas and opinions of a group to the fore. A total of 70 participants initially registered in the group concept mapping web-environment of which 39 participants filled out enough information to be able to give a valid contribution to the analyses. Most of the participants were recruited through the network of the researcher herself and through snowball-sampling, and mainly consist of teachers in primary or secondary education, members of the school direction and other field-experts. Furthermore there 2 researchers, 3 educational managers, s also teachers from secondary education have been participating.

This study is relevant for the implementation of CT in primary and secondary education. It implies the necessity of a critical view of the current implementation measures and may lead to changes in the implementation process so teachers may be supported in their effort of realizing CT in daily teachings.

2. COMPUTATIONAL THINKING IN THE DUTCH CURRICULUM
Although the importance of CT is widely shared, defining CT is still a big challenge. For example, it appears to be difficult to distinguish between the characteristics of CT and the more peripheral conditional cases (Voogt, Fisser, Good, Mishra, & Yadav, 2015). To frame the further discussion the International Society for Technology in Education (ISTE), together with the Computer Science Teachers Association (CSTA), has drawn up an operational definition (ISTE & CSTA, 2011). According to them, CT is "A problem-solving process with the following (not limited) characteristics: (a) formulating problems in a way that makes it possible to solve them with a computer or other instrument; (b) organize and analyze information (data) logically; (c) represent data by abstractions such as models and simulations; (d) automate solutions by algorithmic thinking (a series of ordered steps); (e) identify, analyze and implement possible solutions with the aim of finding the most effective and efficient combination of steps and sources; and (f) generalize the problem-solving process for a wide range of problems so that transfer occurs to other areas "(ISTE & CSTA, 2011).

In addition, the ISTE & CSTA (2011) emphasize in their definition of CT that: "These skills are promoted and supported by a number of views that are essential to CT such as: (1) deal with complexity with confidence; (2) working with perseverance on difficult problems; (3) tolerate lack of clarity; (4) the ability to deal with problems of indefinite duration; (5) the ability to communicate and collaborate with others to achieve a common goal or solution "(ISTE & CSTA, 2011). Some authors, however, claim that the core of CT is an abstraction process; abstraction (reducing unnecessary details) algorithmic thinking (step-by-step plan to solve a problem); automation (repeated tasks); decomposition (split up problem); debugging (testing and tracing of errors) and generalization (pattern recognition) (Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016; Tedre & Denning, 2016; Voogt et al., 2015). Wing (2006) emphasizes that CT is focused on learning to see through underlying concepts of computing. Moreover, Wing was one of the first to note that the research field computing sciences is unique in that it is driven in combination from three disciplines; (a) science (research questions); (b) technology (innovations); (c) society (expectations). That is why Wing pleaded early on to teach CT fundamental concepts in formal educational learning situations as early as possible. (Wing, 2008).

The Curriculum Development Foundation (Stichting Leerplan Ontwikkeling, SLO) recently integrated a definition of computational thinking in their approach. According to SLO CT combines; (a) process (re) formulating problems in such a way that it becomes possible to solve the problem with computer technology; (b) the thinking processes using problem formulation, data organization, analysis and representation to solve problems using ICT techniques and tools (SLO, 2018).

Although programming, CT and computer science are intertwined, there is agreement that CT is much broader than programming (Voogt et al., 2015). In this view, CT represents an attitude (way of thinking) that makes it possible to formulate a problem and the possible solution in computer terms (Grover & Pea, 2013; Kennisnet, 2016; Voogt et al., 2015; Wing, 2006; 2008; Yaşar, 2017). Wing says: "We do not want the computer in the way of understanding the underlying concept. But we also do not want people alone to be able to use the computer without having learned to understand the concept "(Wing, 2008, 3721). She and other authors claim that through 'unplugged' activities for children, CT can be taught at a young age (Lee & Recker, 2018, Wing, 2006, 2008, 2016).

In the Netherlands, the current, broadly based, public debate also seems to lead to a mandatory central role of CT in education (Platform Onderwijs2032, 2016). The first signs of this are visible in the Technology Pact ("National Technology Pact 2020," 2013). This Technology Pact stipulates that from 2020 all primary schools in the Netherlands Science & Technology (S & T) must have structurally introduced their education program ("National Technology Pact 2020," 2013). Although this social aim is primarily intended to compensate for the imminent shortage of technicians by attracting more children to technology, today the emphasis is on value for daily life and future society (Casu & Veer, 2016; Jaipal-Jamani & Angeli , 2017).

CT is a compulsory part of the curriculum in a number of countries (Bocconi et al., 2016). Moreover, the trend is that CT is both obligatory integrated into the curriculum of secondary education and also in basic education. Heintz, Mannilla, and Farnqvist (2016) studied how 10 different countries, including the Netherlands, introduce CT in basic education. Their conclusion is that the emphasis in the approach lies on a narrow perspective of learning ICT basic skills together with programming, or the broader theme computing or computer science. The researchers note that although the term computational thinking is hardly mentioned explicitly, the ideas have been included in the approach in one way or another (Heintz et al., 2016). Incidentally, in countries such as England, where CT has been part of the obligatory curriculum of public education since 2014, programming is seen as "the locomotive that involves all other aspects of the digital world" (Department for Education, 2013; Kennisnet, 2016; et al., 2015). In many other European countries, pupils, according to European Schoolnet (Pijpers, 2017), therefore already learn to program young.

In practice, implementing and reshaping an existing curriculum is a major challenge for designers (Kahle quoted in Ryder, 2015). This is because in practice the curriculum, as intended by the government and curriculum developers, is reinterpreted by teachers through reinterpretations (Ryder, 2015; Valcke, 2010). As a result, the distance between the optimal curriculum (the hypothetically optimal) and the operational curriculum (the instruction activities) is large (Valcke, 2010). Datnow et al (quoted in Sleegers & Ledoux, 2006) formulate some important generic success factors in the implementation of external designs such as: (a) a careful selection process of a program in which teachers participate; (b) adequate training for all teachers; (c) sufficient financial means for introduction; (d) strong leadership at different levels and some key teachers who help coordinate innovation; (e) the extent to which the design contributes to the expectations of the environment. Moreover, the question of what effective strategies are for a successful implementation depends on the content and purpose of the innovation or change (Sleegers & Ledoux, 2006). Nevertheless, a few more abstract points can be formulated at a somewhat more abstract level (Leithwood, Jantzi, & Steinbach, 1999). For example, every change or innovation requires attention for: (1) professionalisation of the teacher; (2) vision on learning and teaching; (3) cooperation between teachers; (4) organization and method of school management. Moreover, a change or innovation will be more likely to succeed if it is clearly aimed at learning children; has broad support, and is in line with important social expectations (Ryder, 2015; Sleegers & Ledoux, 2006).

Several researchers describe the effects of the implementations of CT in basic education (Bocconi et al., 2016, Brown et al., 2013; Pijpers, 2017). From this a number of recommendations and issues can be distilled. The question is which place CT should occupy in the curriculum. As a separate subject or just connecting boxes? In addition, it is necessary that room is made in teacher training, so that teachers have at least a basic understanding of CT (Voogt et al., 2015). Research shows that teachers who learn the basics of CT by linking them to everyday examples, develop a better understanding of CT and the application possibilities (Adler & Kim, 2018; Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011). In addition, this changes the attitude of teachers towards CT, and they are also more willing to integrate CT in daily practice (Yadav et al., 2011, page 468). Further recommendations for the implementation of CT concern four areas: (a) a consolidated understanding of CT (what is CT, what are its overlaps and differences with digital literacy); (b) a coherent integration with clear goals and guidelines to integrate CT into curriculum; (c) a systematic roll-out with assessment tools for CT and adequate support from teachers; (d) support policy by informing stakeholders about the meaning and educational benefits of CT (Bocconi et al., 2016). Other researchers emphasize the importance of adequate assessment tools as a prerequisite for the successful integration of CT in the curriculum (Chen et al., 2017, Grover & Pea, 2013; Román-González, Pérez-González, & Jiménez-Fernández, 2017). . In addition, attention is needed for the role that extracurricular activities can play to get CT in the classroom, for example by designing games (Repenning, Webb, & Ioannidou, 2010)

3. A GCM STUDY

The aim of this research is to investigate how CT, according to professional field experts such as experts and teachers, teachers, teachers in training, administrators, staff and management, should be implemented in basic education. The main question in this research is: "What measures are necessary to implement computational thinking in the curriculum of education?" In addition to the central research question, four sub-questions can be distinguished: (a) which measure should be given the highest priority in the short term implementation; (b) which measure should be given priority in the longer term; (c) to what extent are the measures easy to implement; (d) to what extent are the measures important?

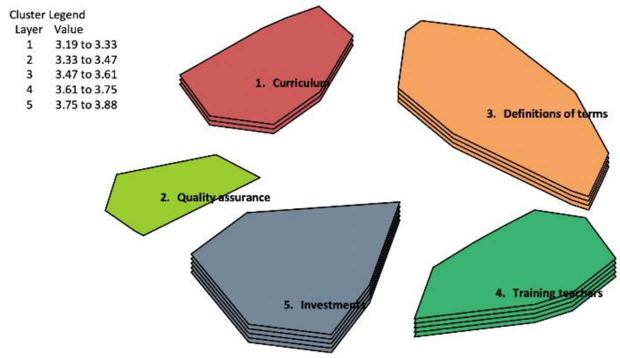
This research had an orienting character. Therefore, the central question was investigated with the research approach of Group Concept Mapping (GCM). This is a participatory mixed-method methodology, which distinguishes itself from an important component of traditional concept mapping approaches. Namely because of the objective way in which a group of stakeholders comes to a shared point of view in relation to a specific issue (Jackson & Trochim, 2002; Kane & Trochim, 2007; Scheffel, Drachsler, Stoyanov, & Specht, 2014; Trochim & McLinden, 2017). GCM is "a structured process in which various participants paint a picture of their ideas and concepts, with the focus on a particular area of interest, and how they are related to each other" (Trochim & McLinden, 2017, p.166). Because in our research both the ideas of the various stakeholders, as well as the prioritization and feasibility were mapped out, GCM showed a suitable methodology for this.

The participants have been involved in various activities such as: generating ideas; sorting ideas into categories; the ranking of the ideas on the basis of certain values (for example: the importance of an idea and the ease with which this can be implemented). After participants anonymously and independently generated the ideas, sorted and assessed, the individual contributions were analyzed by two multivariate statistical analyzes - multidimensional scaling (MDS) and hierarchical cluster (HCA) analysis - for patterns in the data. As a result, a common understanding of the researched issue arose in the research population (Jackson & Trochim, 2002). GCM showed how ideas are related to each other, how they were grouped into more general categories, how much emphasis each idea and category received, and how stakeholders differed in their perspectives. Moreover, GCM suggested which actions could be suitable for the short and which for the long term. In addition, GCM visualized the results of the research using concept maps, pattern matches and go-zone, so that the user could easily include the meaning of the findings (Kane & Trochim, 2007). GCM thus facilitated a bottom-up approach.

There were six process steps to be distinguished: (a) the preparation (selecting participants and developing focus); (b) generating statements (brainstorming); (c) structuring statements (sorting and valuing); (d) the representation of the statements (calculating the maps); (e) the interpretation of the maps; (f) using the maps for planning or evaluation. In summary, there were three moments when participants made their contribution. The first was the brainstorming phase that took place online and took between 15 and 30 minutes per participant. The second moment was sorting and valuing the statements generated in the brainstorming session. This took place in the online environment and took on average 45 minutes of the participant's time.

4. RESULTS

Seventy people initially registered for the online data collection. Thirty of the 70 (43%) people who registered were contributing to the brainstorming session. Twenty-one of the 30 (70%) participants who participated in the brainstorming session made an effective contribution to the sorting and valuation phase. Twenty-six participants grouped the statements, in the way as instructed, into clusters. 39 participants took part in the valuation of the statements on importance. Thirty-two participants appreciated the statements on feasibility. Eight people took part in the interpretation workshop, which was intended to validate the results. The majority (67.5%) used ICT applications in education more often than once a week (Table 3). The 30 participants in the brainstorm initially generated 182 statements of which 97 statements remained after synthesis.



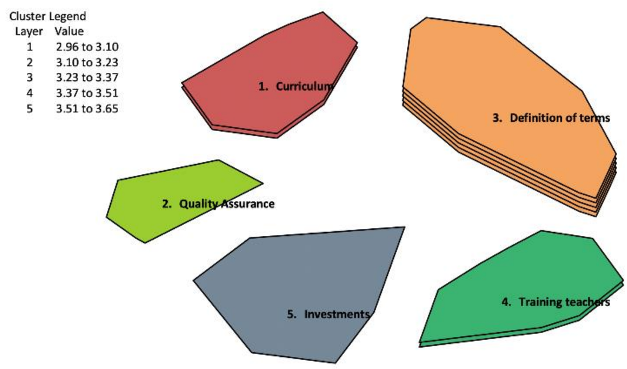


Figure 1 and 2. The resulting clusters of the study.

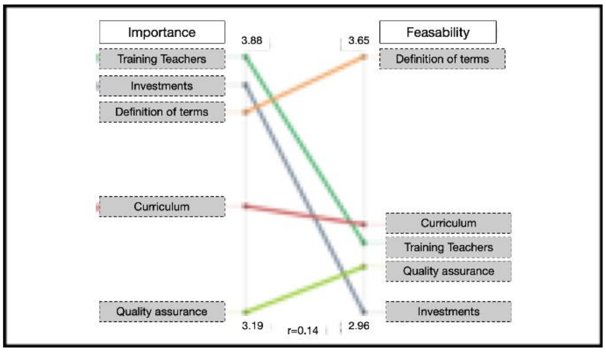


Figure 3. The relation between importance (left) and feasibility (right) of different factors.

By dividing each cluster based on the average value of two assessments into four quadrants (Go zone), suggestions for short-term actions arose. For example, the ideas found in the upper right quadrant were seen by the stakeholder groups as very important and very easy to implement and therefore seemed suitable as short-term action. The quadrant below it suggested the long-term actions; important but less easy to implement (Kane & Trochim, 2007, Scheffel et al., 2014). As a result, determining the necessary actions for planning to implement CT in education became easier {Formatting Citation}. The folder with Gozones showed that the green box contained the statements that were considered both important and feasible. It was about this; (a) 'It is emphasized that CT is not a goal but a means that must be seen separately from the use of computers. It is a way of thinking in cause-effect and analyzing action-response. "(Important M = 3.95 and feasible M = 3.75); (b) That school managers support and facilitate the process of CT implementation (important M = 3.95 and feasible M = 3.75); (c) That time is created to prepare good lessons enriched with technology (important M = 4.64 and feasible M = 3.28); (d) That the school has a flexible team that can handle the changing demands of the digital world (important M = 4.08 and achievable M = 2.59); (e) Ensure continuous learning path with clear goals and guidelines that are shaped by policy makers and practice (important M = 3.79 and feasible M = 3.16)

5. DISSCUSSION AND CONCLUSION

The central question in this research was: "What measures are necessary to implement computational thinking in the curriculum of basic education?" To answer this question, a GCM study was carried out that shows the ideas of teachers, teachers, trainee teachers, directors, managers, directors, staff, and other field experts in education have been mapped. Based on the results of the brainstorming group and cluster activities, five themes were distinguished: 'Determine program content'; 'Guard quality'; 'Define concept CT'; 'Increase teachers' own skills'; 'Reserve time and money'. There was great agreement about the content of the themes / clusters, with the exception of the cluster 'Quality guard'. The themes that were agreed upon are largely in line with the subjects that are mentioned in the literature as conditional to introduce an educational change. These measures could be subdivided into (a) more generic, which are necessary for every educational innovation, and (b) more specific measures that are particularly necessary for the implementation of CT in education. There were two clusters of this nature in general. The first cluster 'Increase own teacher skills' was similar to the generic condition mentioned by Datnow et al (quoted in Sleegers & Ledoux, 2006) that training and professionalization of the teacher is necessary. The second cluster 'Reserve time and money' corresponded to what the literature reported about conditions that are generally considered necessary, such as 'ensuring sufficient financial means for introduction' (Sleegers & Ledoux, 2006). In addition to these generic themes, the group also generated clusters that had more similarities with specific CT implementation measures. For example, the cluster 'Define the concept CT' was very similar to the specific implementation measures for CT as formulated in Bocconi et al. (2016) and Voogt et al. (2015) namely that 'a consolidated and basic understanding of CT' is being worked on. Although the themes 'Determine content program' and 'Quality guard' were found to be less important by the group than the other clusters, they largely corresponded to other CT-specific implementation measures. For example, the condition mentioned in the literature 'a coherent integration with clear goals and guidelines for the curriculum' (Bocconi et al., 2016) had similarities with the cluster 'Determine content program' generated by the group. For example by linking statements such as 'CT to existing subjects so that it does not become something extra' and 'That educational sciences and computing sciences (CS) together develop a learning line with concepts from CS that are appropriate for the different age groups'. The statements in the cluster 'Guard quality' generated by the group showed similarities with a 'systematic rollout in which assessment instruments for CT are discussed' (Bocconi et al., 2016, Chen et al., 2017). An example was the statement 'That the Inspectorate assesses schools on this aspect with a' rubric '(rating scale).'.

- It is emphasized that CT is not a goal but a means that must be seen separately from the use of computers. It is a way of thinking in cause-effect and analyzing action-reaction.
- That school administrators support and facilitate the process of CT implementation.
- The ideas that were seen by the group as very important but less feasible were the implementation measures for the longer term, namely:
    - That time is created to prepare good lessons enriched with technology.
    - That the school has a flexible team that can handle the changing demands of the digital world.
    - Ensuring a continuous learning path with clear goals and guidelines that are shaped by policy makers and practice.

The picture that emerged from this research was that the group of stakeholders thought that it first had to be clear what CT meant before time, money and energy were put into it. Statements from the participants in the interpretation workshop also pointed in this direction. This seemed to correspond with the obstacles identified by Mueller et al (cited in Alenezi, 2017) in the integration of technological innovations in educational practice and the related critical role of teachers, including the necessary support and support (Alenezi, 2017; Bocconi et al., 2016).

This research showed that, according to stakeholders, it was important to pay more attention to increasing their own (teacher) skills and to reserve sufficient time and money for implementation. But that defining what the concept of CT encompassed is the only aspect that was also feasible to implement in practice. This research was relevant for the implementation of CT in education, because it gave reason to take a critical look at the current implementation measures. The research results gave direction to what was needed to support teachers so that they could implement CT in their daily teaching practice.

6. REFERENCES

Adler, R. F., & Kim, H. (2018). Enhancing Future K-8 Teachers' Computational Thinking Skills through Modeling and Simulations. *Education and Information Technologies*, *23*(4), 1501-1514.

Alenezi, A. (2017). Obstacles for Teachers to Integrate Technology with Instruction. *Education and Information Technologies*, *22*(4), 1797-1816.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing Computational Thinking in Compulsory Education. Implications for Policy and Practice. *European Commission, Joint Research Centre*. Luxembourg: Publications Office of the European Union.

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing Computer Science Back into Schools: Lessons from the UK. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*. Denver, Colorado: ACM, 269-274

Casu, C., & Veer, van 't R. (2016). Curriculum Pabo: Make STEAM. *Jeugd in School En Wereld*, *7*. Retrieved January 15, 2019, from http://www.jsw-online.nl/homepage/deze-maand-in-jsw/in-de-nieuwe-jsw/curriculum-pabo-make-steam/

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 and Education*, *109*, 162-175.

Coördinatiegroep. (2018, June). Curriculum.nu. Retrieved January 15, 2019, from https://curriculum.nu/wp-content/uploads/2018/06/Tweede-tussenproduct-ontwikkelteam-Digitale-geletterdheid-Curriculum.nu.pdf

Creswell, J. W. (2014). *Educational Research: Planning, Conducting and Evaluating Quantitative and Qualitative Research*. (4th ed.). Essex, Engeland: Pearson.

Department for Education. (2013). *National Curriculum in England: Computing Programmes of Study*. London. Retrieved January 15, 2019, from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

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., & Farnqvist, T. (2016). A Review of Models for Introducing Computational Thinking, Computer Science and Computing in K-12 Education. In *Frontiers in education conference (FIE)*. Erie, PA: IEEE, 1-9

ISTE, & CSTA. (2011). *Operational Definition of Computational Thinking for K-12 Education*. Retrieved January 15, 2019, from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Jackson, K. M., & Trochim, W. M. K. (2002). Concept Mapping as an Alternative Approach for the Analysis of Open-ended Survey Responses. *Organizational Research Methods*, *5*(4), 307-336.

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.

Kane, M., & Trochim, W. M. K. (2007). *Concept Mapping for Planning and Evaluation*. Thousands Oaks, CA: SAGE Publications.

Kennisnet. (2016). *Computational Thinking in Het Nederlandse Onderwijs. De Stand van Zaken aan de Hand van Interviews*. Retrieved January 15, 2019, from https://www.kennisnet.nl/fileadmin/kennisnet/publicatie/Computational_Thinking_in_het_Nederlandse_onderwijs.pdf

Kirschner, P. A., & Stoyanov, S. (in press). Educating Youth for Non-existent/ Not Yet Existing Professions. *Educational Policy*.

KNAW. (2012). *Digitale Geletterdheid in Het Voortgezet Onderwijs*. Retrieved January 15, 2019, from https://www.knaw.nl/nl/actueel/publicaties/digitale-geletterdheid-in-het-voortgezet-onderwijs

Lee, V. R., & Recker, M. (2018). Paper Circuits: A Tangible, Low Threshold, Low Cost Entry to Computational Thinking. *TechTrends*, 1-7.

Leithwood, K., Jantzi, D., & Steinbach, R. (1999). Changing Leadership for Changing Times. *Changing Education Series*. Philadelphia, PA: Open University Press.

Nationaal Techniekpact 2020. (2013). Retrieved January 15 ,2019, from https://www.rijksoverheid.nl/documenten/convenanten/2013/05/13/nationaal-techniekpact-2020

Onderwijsraad. (2017). *Doordacht digitaal. Onderwijs in het Digitale Tijdperk*. Retrieved January 15, 2019, from https://www.onderwijsraad.nl/upload/documents/publicaties/volledig/Doordacht-digitaal-a.pdf

Pijpers, R. (2017). *Computing Onderwijs in de Praktijk. Wat Kunnen we Leren van de Britten? Kennisnet, (16 November 2017)*. Retrieved January 15, 2019, from https://www.kennisnet.nl/artikel/wat-wij-kunnen-leren-van-computing-onderwijs-in-de-britse-praktijk/

Platform Onderwijs2032. (2016). *Ons Onderwijs2032 Eindadvies*. Retrieved June 2, 2018, from http://onsonderwijs2032.nl/wp-content/uploads/2016/01/Ons-Onderwijs2032-Eindadvies-januari-2016.pdf

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable Game Design and the Development of a Checklist for Getting Computational Thinking into Public Schools. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education*. New York, NY: ACM. 265-269.

Rijksoverheid. (2017). *Herziening van het Curriculum in het Primair en voortgezet onderwijs*. Retrieved May 9, 2018, from https://www.rijksoverheid.nl/documenten/rapporten/2017/02/13/herziening-van-het-curriculum-in-het-primair-en-voortgezet-onderwijs

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.

Rosas, S. R., & Kane, M. (2012). Quality and Rigor of the Concept Mapping Methodology: A Pooled Study Analysis. *Evaluation and Program Planning*, *35*(2), 236-245. doi.: 10.1016/j.evalprogplan.2011.10.003

Ryan, D. (2016). Using Tablet Technology for Personalising Learning. *Journal of Research in Special Educational Needs*, *16*, 1071-1077.

Ryder, J. (2015). Being Professional: Accountability and Zuthority in Teachers' Responses to Science Curriculum Reform. *Studies in Science Education*, *51*(1), 87-120.

Scheffel, M., Drachsler, H., Stoyanov, S., & Specht, M. (2014). Quality Indicators for Learning Analytics. *Journal of Educational Technology & Society*, *17*(4), 117-132.

Sleegers, P., & Ledoux, G. (2006). *Innovatie in het Primair Onderwijs: Strategieën, Ervaringen en Aanbevelingen. Een Literatuurstudie Naar Werkzame Principes*. Retrieved January 15, 2019, from http://expeditieleerkracht.nl/wp-content/uploads/2017/04/innovatie-in-het-primair-onderwijs.pdf

SLO. (2018). *Curriculum van de Toekomst*. Retrieved April 30, 2018, from http://curriculumvandetoekomst.slo.nl/21e-eeuwse-vaardigheden/digitale-geletterdheid/ict-basisvaardigheden

Tedre, M., & Denning, P. J. (2016). The Long Quest for Computational Thinking. *Proceedings of the 16th International Conference on Computing Education Research*. New York, NY: ACM, 120-129.

The Concept System® Global MAX™ (Build 2016.046.12) [Web-based Platform]. (2016). Ithaca, NY. Retrieved January 15, 2019, from http://www.conceptsystemsglobal.com

Thijs, A., Fisser, P., & Hoeven, M. van der. (2014). *21e Eeuwse Vaardigheden in het Curriculum van het Funderend onderwijs*. *SLO, 2017*. Enschede: SLO Nationaal expertisecentrum.

Trilling, B., & Fadel, C. (2009). *21st Century Skills: Learning for Life in Our Times.* San Francisco, CA: Jossey-Bass.

Trochim, W. M. K. (1989). An Introduction to Concept Mapping for Planning and Evaluation. *Evaluation and Program Planning*, *12*(1), 1-16.

Trochim, W. M. K., & McLinden, D. (2017). Introduction to a Special Issue on Concept Mapping. *Evaluation and Program Planning*, *60*, 166-175.

Vakvereniging Informatica & Digitale Geletterdheid. (2017). *Visie van de Vakvereniging i&i op Digitale Geletterdheid*. Retrieved January 15, 2019, from http://www.platformvvvo.nl/2017/12/07/visie-vakvereniging-ii-op-digitale-geletterdheid/

Valcke, M. (2010). *Onderwijskunde als Ontwerpwetenschap. Een Inleiding voor Ontwikkelaars van Instructie en Voor Toekomstige Leerkrachten.* Gent, België: Academia Press.

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.

VSNU. (2016). *De Digitale Samenleving. Nederland en zijn Universiteiten: Internationale Pioniers in Mensgerichte Informatietechnologie*. Retrieved January 15, 2019, from https://www.vsnu.nl/files/documenten/Publicaties/VSNU_De_Digitale_Samenleving.pdf

Wing, J. M. (2006). Computational Thinking. *Communications of the Association for Computing Machinery (ACM)*, *49*, 33-35.

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

Wing, J. M. (2016). *What is Computational Thinking? Computer Science Teachers Association Video Interview Series*. Retrieved January 15, 2019, from https://www.youtube.com/watch?v=fSoknljUI4Q

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing Computational Thinking in Education Courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*. New York, NY: ACM, 465-470.

Yaşar, O. (2017). The Essence of Computational Thinking. *Computing in Science and Engineering*, *19*(4), 74-82.

General Submission to Computational Thinking Education

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019*. Hong Kong: The Education University of Hong Kong.

Exploring the Role of Algorithm in Elementary School Students' Computational Thinking Skills from a Robotic Game

Hsin-yin HUANG[1], Shu-hsien HUANG[2], Ju-ling SHIH[3]*, Meng-jung TSAI[4], Jyh-chong LIANG[5]
[13] Department of Information and Learning Technology, National University of Tainan, Taiwan
[2]Department of Information Management, National Chin-Yi University of Technology, Taiwan
[45] Program of Learning Sciences, National Taiwan Normal University, Taiwan
d10655004@gm2.nutn.edu.tw, shushein@gm.ncut.edu.tw, juling@mail.nutn.edu.tw, mjtsai99@ntnu.edu.tw, aljc@ntnu.edu.tw

ABSTRACT

This study explored the role of elementary school students' computational thinking skills and how they were related to the performance in a self-designed robotic game <STEM Port>. This game required the operation of robots on the large world map in which the players carried out competitive tasks with block coding. Five computational thinking dimensions in terms of problem-solving skills were examined which includes Algorithm, Evaluation, Decomposition, Abstraction, and Generalization. Among total of 99 six-graders, there were 66 and 33 students with high and low algorithm skill respectively. The regression results showed that students with high algorithm skill could have overall better performance in the game than those with lower algorithm skill, especially in decomposition and generalization dimensions in the four rounds of the game. It was also found that the high algorithm students' performances in the beginning and end round were predicted by their perceptions of computational thinking skills, however, none of low algorithm students' perceptions of computational thinking skills were associated with their performances. It implied the importance of CT education to the students which would lead them to have better performance in the complex problem-solving situations such as the fast advancing world lies before them.

KEYWORDS

game-based learning, computational thinking, robotic game, algorithm skill

1. INTRODUCTION

With the rapid development of science and technology, software program computing knowledge and skills are indispensable, thus the demand for computational talents is global. Therefore, the computational thinking education has become the world trend.

Computational thinking (CT) is regarded as one of the basic keyskills in the 21st century, which can be narrowly defined as computer programming skills or widely defined as problem-solving skills. Most countries aim to develop students' logical thinking skills and problem-solving skills through CT in curriculum reform (Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016). The concept includes five basic dimensions, such as algorithm, evaluation, decomposition, abstraction and generalization. In order to enhance students' learning motivation and to observe their CT performances, this study designs a strategic robotic game which requires the students to apply both their coding skills and problem-solving skills in order to win the game. The game is designed for the students to use block coding to give commands to the robots to make directive actions of moving forward and making turns so that the robots can navigate on the map to the designated locations. Students discuss how to make the robot to the right location.

In the real world, competition is a common social phenomenon, and in the teaching environment, teachers often use competitive psychology to stimulate students' learning motivation in order to enhance their learning effectiveness (Lin, Huang, Shih, Covaci, & Ghinea, 2017). Therefore, the game adopts the strategic mechanism, which allows learners to cooperate and compete with other players in order to achieve the goals.

2. RELATED WORK

2.1. Game-based Learning

Children are happy in the games since they are interesting, exciting, and sometimes challenging. Nowadays, many teachers conduct game-based learning (GBL) to enhance students' learning motivation, and most importantly, to embed learning with carefully designed curriculum. From the experiential activities, students are more immersed in the learning scenarios so that they enhance learning effectiveness and get wider and deeper knowledge and skills (Yien, Hung, Hwang, & Lin, 2011).

GBL encompasses interesting, interactive and pleasant features of games with the teaching content, which stimulate learners' motivation to learn spontaneously and repeatedly (Perini, Luglietti, Margoudi, Oliveira, & Taisch, 2018). The spirit of Game-based learning is to allow learning happen in the fun process (Perrotta, Featherstone, Aston, & Houghton, 2013). Children gradually learn and construct knowledge in the process. Passive learning becomes more active and engaged in GBL. Students actively explore the issues assigned by teachers from various perspectives, work with peers to find answers, and then develop the skill to communicate, coordinate, and do creative thinking and problem solving. But GBL is different with game. This distinction is that the design process of games for learning involves balancing the need to cover the subject matter with the desire to prioritize game play (Plass, Homer, & Kinzer, 2015).

Game becomes an indispensable auxiliary element in the learning activities. While implement game mechanisms and elements in activities, such as scoring, ranking, getting badges, doing competition and interaction, can turn the entire teaching activity into a gamified activity (Curzon, Dorling, Ng, Selby, & Woollard, 2014; Perrotta et al., 2013). The game used in this research was designed from a board game, which makes the daily classroom lecture content interesting and challenging.

2.2. Computational Thinking

Computational thinking originates from the concept of programming in computer science. The concept is to use computer language to manipulate computers to solve daily life problems. It is a model and a process of thinking that uses the basic concepts of computer science to solve problems. This is an skill everyone should have (Wing, 2006).

Computational thinking influences everyone in the works of all fields that its vision poses a new educational challenge for our society, especially for our children (Wing, 2008). The skill of abstraction and automation is a way to accelerate the efficacy of thinking, analyzing, and taking actions. Problem solutions can be produced through analyzing problems, making judgments and decisions, and integrating tools and resources to solve problems. The purpose is to help students to solve problems by assessing the appropriate tools and strategies to be used in specific situations. It is also one of the most important skills in the 21st Century (Mohaghegh & McCauley, 2016). Computational thinking has been studied by many scholars since Wing put forward it, and Selby, Dorling, and Woollard (2014) defined the five core concepts as follows.

1. **Algorithm** is to develop rules that can solve similar problems step by step and be implemented repeatedly.
2. **Evaluation** is the process of ensuring an algorithmic solution is a good one. Various properties of algorithms need to be evaluated including whether they are correct, are fast enough, are economic in the use of resources, and are easy for people to use.
3. **Decomposition** is a way of thinking about problems, algorithms, artefacts, processes, and systems in terms of their parts. The separate parts can then be understood, solved, developed, and evaluated separately. This makes complex problems easier to solve and large systems easier to design.
4. **Abstraction** is another way to make problems or systems easier to think about. It simply involves hiding details and removing unnecessary complexities. The skill is to choose the right details to hide so that the problem becomes easier to handle without losing anything important.
5. **Generalization** is a way of quickly solving new problems based on previous problems solved. It is to take an algorithm that solves specific problems and adapts the algorithm to solve a whole class of similar problems.

Computational thinking is an important concept in the new curriculum plan to be implemented in August 2019 in Taiwan. It is also the main axis of information technology courses to cultivate students' logical thinking and systematic thinking. With appropriate curriculum design, students can use information technology tools in formal learning and to develop the skill of communication, coordination, innovative thinking, independent thinking, problem solving, and so forth. A high quality computing education equips students to use computational thinking and creativity to understand and change the world (Department for Education, 2013, p. 188).

3. RESEARCH DESIGN

3.1. Game Design of <STEM Port>

The game used in this study is a game called <STEM Port> which was self-designed robotic game based on the historical context of Great Voyage. It was oriented from a board game <Fragrance Channel> which was also a self-made game. <STEM Port> was designed to embed STEM educational movement and interdisciplinary concepts. In the game, a big map in the size of 600 x 400 cm showed the geographic area covered in the Age of Discovery in the 17th century (Figure 1). Students were divided into five groups, and role-play one of the five countries including England, Netherland, Portugal, Spain, and France. Robots represented ships of respectively countries by colored lights. Each country chose ship parts such as hull, oar, mast, and weapons which would influence their total ship power, including propulsion power, cargo capacity, deceleration, firing distance, arm force, and sailing duration. With these parameters, all groups started up with different strengths and weaknesses. Then, the players took turns to move their ships by block coding to go to designated colonies to trade for spices. Whichever country completed its spice task won.

Figure 1. Game design of <STEM Port>.

From the problem-solving point of view, the students have to apply their CT skills in order to complete the tasks of the game. They have to “decompose” the game requirements and limitations of the tasks, and try to reach the goal in limited rounds. Then, they apply “algorithm” skills to calculate the distance, angle, speed of the robots, and do “abstraction” to turn the navigation route into coding blocks. The students “evaluate” the differences between the predict path and the actual path, and make adjustments to their actions in the next round. As the students solicit the main strategies for the game, they can “generalize” the patterns to different rounds and quickly use the resources around them to solve the problems.

This robotic game mechanism required the students to use block coding (Figure 2), in this case was mBlock, to control robot ships to move forward or make turns. They had to estimate the distance to go to their destinations, and then used the limited game points to move the robots. In the navigation process, they had to decide whether they would do trading or going into battles. By using simple and basic commands, the students would focus on using the coding skills to solve the game problems and to complete their tasks. Thus, a coding-based CT and problem-solving-oriented CT were functioned at the same time in the game. This programming environment can cultivate students' CT abilities during programming activities by enabling them to concentrate on the problem solving process as they learn (Kong, Chiu, & Lai, 2018).

Figure 2. mBlock coding program.

3.2. Computational Thinking Questionnaire

In this study, four 5th grade classes of students in an elementary school in southern Taiwan were invited. There were 65 boys and 34 girls with a total of 99 students participated in the game-based learning. Each class played an individual game in four different days. This study used mBot robots and navigation route prediction records as well as computational thinking questionnaires as research tools to assess learners' CT performances in the robotic game. The research process is as Figure 3.

Before the start of the game, the CT questionnaire was distributed to the students as the pre-test. Then, the students played the game <STEM Port> for about 60 minutes. After the game was finished, post-test CT questionnaire was conducted. The results of the questionnaires were used to cross-analyzed with the students' gaming outcomes with regressions.

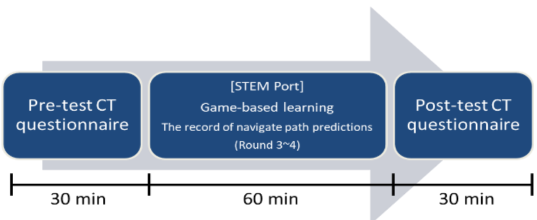


Figure 3. Research process.

The CT questionnaire used in this study is newly designed based on the relevant literature (Atmatzidou, Demetriadis, & Systems, 2016; Curzon et al., 2014; Dagienė, Sentance, & Stupurienė, 2017; Selby, Dorling, & Woollard, 2014) and taking the principles of International Challenge on Informatics and Computational Thinking as the main reference.

To construct a valid and reliable questionnaire for computational thinking, five dimensions were designed, including items designed specifically for the participants. In total, two faculty members specializing in education validated the items twice (Chu, Liang, & Tsai, 2019).

The questionnaire includes five dimensions: Algorithm, Evaluation, Decomposition, Abstraction, and Generalization. Each dimension has 5 questions with total of 25 questions. For example, "I will try to dissect the big problems into small parts" is to test out the students' perception to the Decomposition skills; "I will try to think of the most efficient way to solve the problems" is to test out their Evaluation skills; "I will figure out the detailed steps for problem-solving" is for the Algorithm skills; "I will try to find out the key factor of the problem" and "I will try to use previous experience to solve new problems" is to test their Abstraction and Generalization skills respectively. The total correlation analysis showed that the correlation coefficients of the overall divergence ranged from 0.42 to 0.61 and both reached significant ($p<.01$), which was a medium-high correlation, indicating that each dimension has a certain degree of correlation. The reliability Cronbach' α of this scale is 0.91. The reliabilities for the five dimensions ranged from 0.74 to 0.83. The pattern coefficient of all dimensions is above 0.4. It shows that the reliability and validity of questionnaire is good.

Questionnaire was distributed to total of 99 students, and two invalid ones were excluded from the returned questionnaire, which ended up with 97 copies for analysis.

3.3. Navigation Route Analysis

The students in each of the four classes were divided into 5 groups of 4 to 5 members. Each group need to draw the navigation route prediction record in every round while completing the spice trade mission. The navigation route is shown in Figure 4. The student's predicted path and actual path were documented to assess students' spatial concepts, judgements of distance and angle, and programming skills.

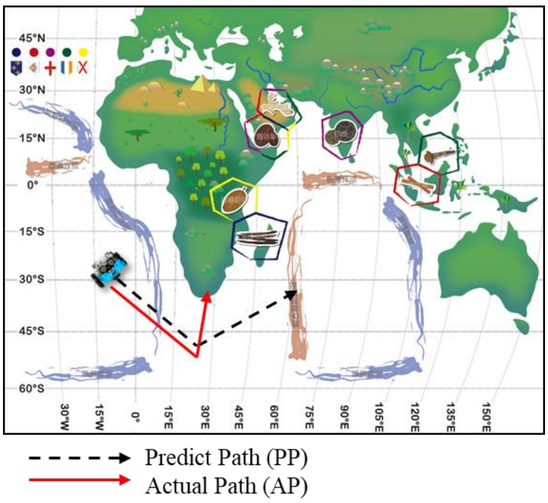


Figure 4. Prediction of the navigation route of mBot.

The formula of the Pythagorean theorem is used to calculate the distance between the student's predicted location and the actual location. It is to measure their CT performance through coding. The formula is as Figure 5.

The distance between predict path to actual path:

$$D = \sqrt{|PA^2 - PP^2|}$$

D = Distance, AP = Actual Path, PP = Predict Path

Figure 5. The distance between predict path to actual path

4. RESULTS AND DISCUSSION

4.1. Navigation Route Analysis Results

In order to understand the changes of the students' learning in the game activity, all of their predict path and actual path of the robot navigation were documented. Then the distance between the two paths were calculated.

Since the student number is lower than 100, the p value set 0.1 as the benchmark. The results were shown as Table 1. For the total average of the distances of the four classes, Round 1 was 22.74, Round 2 was 28.73, Round 3 was 13.17, and Round 4 was 12.09. The sampled t-test results show that the distance between Round 1 and 2 had increased (t=-2.011, p=0.047) since Round 1 involved only straight line Round 2 involved making turns. Calculation to angles had added complexity. Round 2 and 3 has significant difference (t=7.68, p=0.000) as the distance had become much smaller at this stage which means the students had been familiarized with the process, calculations, and predictions. Round 3 and 4 had no significant difference (t=0.91, p=0.364) showing that students had reached the peak in Round 3, and Round 4 had remained stable.

Table 1. Predicted distance difference value paired sample t-test.

Distance	N	Mean	SD	*t*
distance1	97	22.74	13.65	-2.01^{*}
distance2	97	28.73	21.55	
distance2	97	28.73	21.55	7.68^{**}
distance3	97	13.17	9.53	
distance3	74	13.90	10.72	.91
distance4	74	12.09	11.92	

**p*<.05, ***p*<.01

Taking a class as an example (as shown in Figure 6) to look at the details of their navigations. Y axis is the distance difference, and X axis is the rounds per country. Thus, the changes of each group can be seen. All of the groups had obvious improvements along the game that at the end of the game, they had all reached the expected learning outcome.

Two of the five groups had more than 30 cm differences in the first round, including France (37.34 cm) and Portugal (33.00 cm). Surprisingly, Netherlands' difference was 0 cm; their performances in other rounds were also good (5.83 cm in Round 3, and 10 cm in Round 4). Their robot had made a wrong turn (left from right) in Round 2 (75.66 cm) so they had very big difference then. However, they made up the mistakes in Round 3 nicely. It is worth mentioning that although Portugal was not the most accurate group, their gap from the first round to the fourth round was in a smooth gradual descending signifying the students' skill had improved along the way.

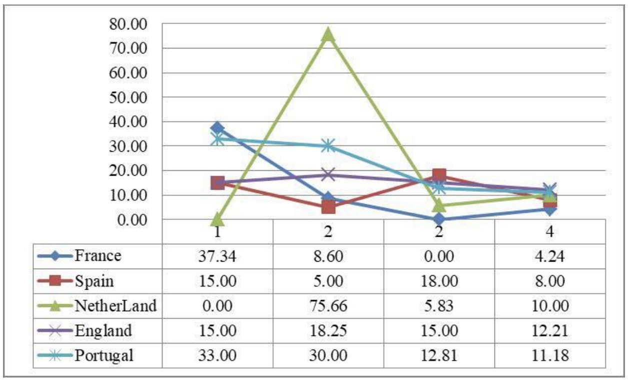


Figure 6. The distance between the predict path and the actual path.

Because the content of the game was directly related to the Algorithm dimension in the computational thinking, the students were divided into two groups based on the results of the cluster analysis of the pre-test scores of algorithm. The distribution of the two groups: high algorithm group (HA) (n=66; mean=4.57) and low algorithm group (LA) (n=31; mean=3.26). The quasi-experimental approach is used to understand the differences between the two groups, and to explore the relationship between them. The differences between the predict path and actual path of the four rounds were presented using paired samples *t*-test (as shown in Table 2). Nevertheless, three out of the four classes played the game for four rounds; one class didn't complete the tasks for the fourth round so the number of students in the statistics dropped by 23 students.

Table 2. HA group and LA group Predicted distance difference value paired sample t-test.

Group	Distance	N	Mean	SD	*t*
HA	distance1	66	24.26	12.87	-.34
	distance2	66	25.29	17.19	
	distance2	66	25.29	17.19	7.77^{***}
	distance3	66	11.32	6.23	
	distance3	54	11.40	6.80	-.39
	distance4	54	12.19	12.48	
LA	distance1	31	19.50	14.89	-2.57^{*}
	distance2	31	36.05	27.65	
	distance2	31	36.05	27.65	3.77^{*}
	distance3	31	17.13	13.52	
	distance3	20	20.62	15.72	1.89
	distance4	20	11.80	10.55	

**p*<.05, ****p*<.001

In HA group, Round 1 and Round 2 had no significant difference (*t*=-0.34, *p*=0.735), indicating that students were still getting familiar with game in Round 2. To Round 3, it began to be significant (*t*=7.77, *p*<0.001), indicating that the distance was greatly reduced, and the students had done well at this stage. From Round 3 to Round 4, it was not significant (*t*=-0.391, *p*=0.697), indicating that the student's performance became consistent and stable after Round 3. The distance of LA from Round 1 to Round 2 (*t*=-2.57, *p*=0.015) and to Round 3 (*t*=3.74, *p*<0.01) were significant. From Round 3 to Round 4 was (*t*=1.893, *p*=0.074), indicating that the game had allow the students to continually improve their CT skills to the last round.

4.2. Computational Thinking Skills

In order to explore how the students' CT skills influence their gaming outcomes, regression analysis was conducted using the five dimensions of the CT skills as predictors. Overall speaking, the CT skills of LA group were not related to the outcome, therefore, only the CT skills of HA group were discussed in the following explanations.

In Round 1 (Table 3), the analysis result of the HA group's Decomposition skill was positive (*t*=2.96, *p*=.004), indicating that if the students know how to dissect the problem into small parts, they can have better performance. The analysis result of the Generalization skill was negative (t=-1.94, p=.057<.1), indicating that making reference of their current strategies to the new round was not what the students should do at this stage.

Table 3. The First Round Coefficients of each CT dimension with regression analysis.

Model		Unstandardized Coefficients		Standardized Coefficients	*t*
		B	Std. Error	Beta	
HA	Algorithm	-6.09	4.379	-.21	-1.39
	Evaluation	-3.28	3.377	-.14	-.97
	Decomposition	7.99	2.700	.45	2.96**
	Generalization	-5.81	2.990	-.30	-1.94
	Abstraction	.391	3.572	.02	.11
LA	Algorithm	-9.29	8.34	-.24	-1.11
	Evaluation	.313	6.94	.01	.05
	Decomposition	1.58	4.53	.08	.35
	Generalization	-3.51	4.49	-.18	-.78
	Abstraction	4.29	5.37	.22	.79

**p<.01

In Round 2, the analysis result of the HA group's Generalization skill was negative (t=-1.64, p=0.106), indicating that the students were still familiarizing with the game and programming skills. In Round 3, the analysis results of both group's skills were unpredictable to the outcome. Statistically did not reach significance. However, students in the HA group had reached their peak performance. In Round 4 (Table 4), the analysis result of the HA group's Decomposition skill was negative (t=-3.46, p<0.001), which is different from Round 1, indicating that the Decomposition skill was not as important at the end stage since they were supposed to be very familiar with the game mechanism and programming. However, the result of Evaluation skill was positive (t=2.25, p=0.029), indicating that being able to know what strategies were good or bad for their victory, and to apply correct strategies became more important at the end of the game.

Table 4. The Fourth Round Coefficients of each CT dimension with regression analysis.

Model		Unstandardized Coefficients		Standardized Coefficients	*t*
		B	Std. Error	Beta	
HA	Algorithm	3.82	4.39	.14	.87
	Evaluation	7.60	3.39	.35	2.25*
	Decomposition	-9.18	2.65	-.54	-3.46**
	Generalization	3.09	3.19	.17	.97
	Abstraction	-4.12	3.51	-.19	-1.18
LA	Algorithm	-5.15	7.48	-.18	-.69
	Evaluation	-.38	5.99	-.02	-.06
	Decomposition	6.58	4.26	.47	1.54
	Generalization	-5.96	4.63	-.43	-1.29
	Abstraction	3.53	5.21	.28	.68

*p<.05, **p<.01

The results of this study showed that this activity was helpful to explore the functions of the CT skill dimensions of the students. For the HA group, and the students' skill of Decomposition and Evaluation were closely correlated to their gaming outcomes. Generally speaking, students with high algorithm skill performed better than those with lower algorithm skill. Algorithmic thinking is the core element of CT, and is difficult for the LA group. It is our aim to plan curriculum that would increase students' algorithmic thinking thus better fill up the gap between the LA and HA students.

5. CONCLUSION

In this study, the students can obtain and practice the spatial concept, measure the angles, distances, and speed, as well as solving the navigation problems which increased students' interests in learning and CT skills. From other research (DomíNguez et al., 2013), students who completed the gamified experience got better scores in practical assignments and in overall performances. Since this study was short, it would be interesting to know whether the students can have better performance in their formal classes after playing the game in the future study. Students were excited and immersed in playing the <STEM Port>. The students worked together in group, discussed how to get the highest points to win. The game had received many positive feedbacks from the students. It is likely to reduce distractions, thereby improving the quality of learning beyond what is provided in this activity.

The students need to establish spatial concept, and use their CT skills to complete the tasks. Students in HA group used the Decomposition skill the most in the first round, since they had to try out to dissect the tasks and transformed the route into codes. In Round 2 and 3, they were familiarizing the game mechanism and the coding skills, so their performances tend to be more stable. Until the last round, Evaluation skill started to take effects since they started to use their experiences, resources, and strategies to apply their successful experience to the end. That also indicated that the game was appropriately designed to require the students to apply different CT skills in the game. Reversely, from students' CT skills, it could even predict how the students might perform in the game since the predictors were elicited from the statistics.

In this study, games helped students to get more practices and to reinforce existing knowledge and skills. It allowed students to integrate and apply their knowledge outside of the school context (Plass, Homer, & Kinzer, 2015). It also transformed the lecture-type teaching into interesting learning scenario. Although this study concluded with specific CT dimensions that can better predict the students' gaming outcomes, implications were twofold. For one, the students with low Algorithm skills cannot achieve as much as those with high Algorithm. It is necessary for us to help the students to have better Algorithm skills so that they can accomplish more in the strategic game and problem-solving tasks, and can have better performance in general. For two, more dimensions of CT skills should be reinforced in our pre-activity training. CT courses should be diagnosed with the five dimensions, and make sure students were educated in a more well-rounded CT skills so that they can have better performance in the complex problem-solving situations such as the fast advancing world lies before them (Chen et al., 2017).

6. ACKNOWLEDGEMENTS

This study is supported in part by the Ministry of Science and Technology of the Taiwan, under MOST 104-2628-S-024 -002 -MY4.

7. REFERENCES

Atmatzidou, S., Demetriadis, S. J. R., & Systems, A. (2016). Advancing Students' Computational Thinking Skills through Educational Robotics: A Study on Age and Gender Relevant Differences. *Robotics and Autonomous Systems, 75*, 661-670.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education-Implications for Policy and Practice* (No. JRC104188). Joint Research Centre (Seville site).

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing Elementary Students' Computational Thinking in Everyday Reasoning and Robotics Programming. *Computers & Education, 109*, 162-175.

Chu, Y. K., Liang, J. C., & Tsai, M. J. *The Computational Thinking Scale for Programming.* Paper Presented at the 3th International Conference on Computational Thinking Education (CTE 2019), Hong Kong.

Curzon, P., Dorling, M., Ng, T., Selby, C., & Woollard, J. (2014). *Developing Computational Thinking in the Classroom: A Framework.* Retrieved October 7, 2014 from https://eprints.soton.ac.uk/369594/.

Dagienė, V., Sentance, S., & Stupurienė, G. J. I. (2017). Developing a Two-dimensional Categorization System for Educational Tasks in Informatics. *Informatica, 28*(1), 23-44.

Department for Education (2013). *The National Curriculum in England, Framework Document*. Retrieved September 11, 2013 from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study.

DomíNguez, A., Saenz-De-Navarrete, J., De-Marcos, L., FernáNdez-Sanz, L., PagéS, C., MartíNez-HerráIz, J. J. J. C., & Education. (2013). Gamifying Learning Experiences: Practical Implications and Outcomes. *Computers & Education, 63*, 380-392.

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A Study of Primary School Students' Interest, Collaboration Attitude, and Programming Empowerment in Computational Thinking Education. *Computers & Education, 127*, 178-189.

Lin, C. H., Huang, S. H., Shih, J. L., Covaci, A., & Ghinea, G. (2017). Game-based Learning Effectiveness and Motivation Study between Competitive and Cooperative Modes. *Proceedings of the 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT).* IEEE, 123-127.

Mohaghegh, D. M., & McCauley, M. (2016). Computational Thinking: The Skill Set of the 21st Century. *International Journal of Computer Science and Information Technologies, 7*(3), 1524-1530.

Perrotta, C., Featherstone, G., Aston, H., & Houghton, E. J. S. N. (2013). *Game-based Learning: Latest Evidence and Future Directions.* Slough: NFER.

Plass, J. L., Homer, B. D., & Kinzer, C. K. J. E. P. (2015). Foundations of Game-based Learning. *Educational Psychologist, 50*(4), 258-283.

Selby, C., Dorling, M., & Woollard, J. (2014). *Evidence of Assessing Computational Thinking*. Retrieved December 10, 2014 from https://eprints.soton.ac.uk/id/eprint/372409.

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

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

Yien, J. M., Hung, C. M., Hwang, G. J., & Lin, Y. C. (2011). A Game-based Learning Approach to Improving Students' Learning Achievements in a Nutrition Course. *Turkish Online Journal of Educational Technology-TOJET, 10*(2), 1-10.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Developing Computational Thinking Practices through Digital Fabrication Activities

Megumi IWATA[1*], Kati PITKÄNEN[2*], Jari LARU[3], Kati MÄKITALO[4]
[1234] University of Oulu, Finland
megumi.iwata@student.oulu.fi, kati.pitkanen@student.oulu.fi, jari.laru@oulu.fi, kati.makitalo@oulu.fi

ABSTRACT

This paper presents a study of developing computational thinking (CT) practices through digital fabrication activities, such as creating tangible artefacts with digital tools. The aim of the study was to explore the potential of digital fabrication activities for developing CT practices. We investigated three cases of school visits where the students engaged in digital fabrication activities in Fab Lab Oulu, northern Finland. Based on the perspectives of the teachers who participated in the activities and facilitators who ran the activities, we identified that digital fabrication activities have the potential to develop CT practices, especially formulating problems in order to use a computer for assistance, thinking logically, and implementing possible solutions efficiently and effectively. The findings suggested that the nature of digital fabrication activities, such as frequent use of computers and complex problem-solving, encouraged development of CT practices. However, we also uncovered the possibility that CT is not being adequately defined by the teachers and facilitators.

KEYWORDS

computational thinking, digital fabrication, fab lab, secondary school education

1. INTRODUCTION

The term *computational thinking* (CT) was coined by Seymour Papert in 1996 in his article on mathematics education (Papert, 1996). Along with the rapid development of digital technology, the needs for integrating CT in education are increasing to equip children with a fundamental skill that is necessary in modern society. Jeannette M. Wing provided a broader recognition towards CT by describing CT as “a fundamental skill for everyone, not just for computer scientists” (Wing, 2006). According to her later definition, “Computational thinking is 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).

CT is deeply rooted in computing. Wing (2008) explained that CT is built on the fundamental concepts of computing: *abstraction* (“mental” tool of computing) and *automation* (operation of abstractions). Denning (2009) claimed that CT itself is not a principle, but a practice. He reminded about the importance of keeping in mind the fundamental concepts of computing, not only the operational term: CT. In order to apply CT, it is crucial to understand the concepts of computing underlying CT.

In this study, we aimed to explore the potentials for developing CT practices through digital fabrication activities. *Digital fabrication* is commonly described as a process of making digital or physical artefacts with the help of computers, for example, using a laser cutter, a 3D printer or a CNC router. It emphasizes experiment-based, project-based and interest-driven knowledge construction (Blikstein & Krannich, 2013). Applying digital fabrication provides opportunities for innovation and invention for various groups of people, such as entrepreneurs and educational institutions.

We chose digital fabrication as a context of the study because of its following two characteristics. Firstly, digital fabrication is fundamentally a problem-solving activity. It is typically implemented as ill-structured open-ended problem-solving activities (Pitkänen, Iwata & Laru, in press). As CT is considered as a thought process to support understanding problems and formulating solutions effectively (Wing, 2010), digital fabrication may be an environment to apply abstract thought processes into practice.

Secondly, digital fabrication emphasizes STEM (Science, Technology, Engineering and Mathematics) areas. Blikstein (2013) suggested that “digital fabrication is typically associated with the learning and practice of STEM disciplines” (p. 13). STEM-rich digital fabrication activities take place around design and engineering practices, typically integrating digital tools (Bevan, 2017). Hu (2011) maintained that solving STEM problems may foster a person’s CT ability. As CT combines mathematical and engineering thinking (Wing, 2006), STEM-based digital fabrication activities may provide opportunities to apply CT into practice.

2. CT IN EDUCATIONAL CONTEXTS

Previous studies introduced several practical definitions of CT which can be implemented in school contexts (e.g., Astrachan & Briggs, 2012; Barr, Harrison, & Conery, 2011; Barr & Stephenson, 2011; Grover & Pea, 2013). These definitions commonly describe CT as a problem-solving skill that breaks down a problem into smaller manageable pieces and implements solutions to confront the problem.

Mannila et al. (2014) studied how CT is implemented in schools. The paper reviewed previous studies and national curricula in five different countries to seek a possibility to include CT in schools. They also conducted a survey for school teachers to investigate implementation of CT at schools. For the survey, they used nine CT aspects (data collection, data analysis, data representation, problem decomposition, abstraction, algorithms, automation, simulation and parallelization). They concluded that some school teachers had already begun applying CT practices in schools, even if CT had not been defined in the national curricula. However, among the nine aspects of CT, they did not find many relevant answers regarding abstraction and automation that are the fundamental concepts of CT.

In this study, we use the definition of CT practices introduced by Barr, Harrison and Conery (2011) based on the joint project by International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA). The project intended to make accessible concepts of CT for teachers by building operational definition of CT which goes along with educational objectives and classroom practices (Barr et al., 2011). Regarding CT as a problem-solving process in K-12 contexts, the definition consists of six CT practices: 1) Formulating problems in a way that enables us to use a computer and other tools to help solve them, 2) Logically organizing and analyzing data, 3) Representing data through abstractions, such as models and simulations, 4) Automating solutions through algorithmic thinking (a series of ordered steps), 5) Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, and 6) Generalizing and transferring this problem-solving process to a wide variety of problems (Barr et al., 2011).

3. DIGITAL FABRICATION AS A WAY TO DEVELOP CT PRACTICES

Kafai (2016) argued that CT in K-12 contexts is social and includes creative practices. She emphasized the importance of creating tangible and shareable applications, where children are able to use the skills of programming in a meaningful way. Previous studies indicated the possibilities of developing CT practices by making tangible artefacts with digital tools. *Making* is described as "a class of activities focused on designing, building, modifying, and/ or repurposing material objects, for playing or useful ends, oriented toward making a "product" of some sort that can be used, interacted with, or demonstrated" (Martin 2015).

Rode et al. (2015) indicated that CT can be taught in the creative process of making artefacts. Kafai et al. (2014) found the potential of using electronic textiles (e-textiles) to introduce CT for the high school computer science class. Montero (2018) stated that hands-on digital fabrication activities are beneficial to introduce CT, rather than through programming alone, since such activities reduce the negatively biased perception toward "programming" or "coding".

Current digital tools (e.g. for programming and electronics) which are affordable and intuitive, and community spaces (e.g., makerspaces) providing publicly accessible high-end manufacturing equipment have encouraged educators to apply digital fabrication in education. A motivation to apply digital fabrication in education is based on the learning theory of *constructionism* introduced by Papert (Blikstein, 2013). Constructionism emphasizes that an individual constructs knowledge effectively in interactions with the physical and social environments, such as building meaningful artefacts and publicly sharing objects with others (Blikstein, 2013; Papert & Harel, 1991). Digital fabrication activities motivate children to build personally meaningful artefacts in STEM-rich environments and may involve opportunities to develop CT practices in the process.

4. METHODS

4.1. Research Context and Subjects

Fab Lab Oulu, established in 2015 at the University of Oulu, northern Finland, has arranged digital fabrication activities for school visitor groups since 2016. *Fab Labs* are a type of makerspaces offering digital fabrication facilities (Cavalcanti, 2013). The original Fab Lab was conceived by Professor Neil Gershenfeld in collaboration with Bakhtiar Mikhak in the Massachusetts Institute of Technology (MIT) to provide a creative space for university students (Blikstein, Martinez, & Pang 2015). Fab Labs have spread all over the world resulting in a global network to share common tools and processes (Fab Foundation, n.d.). A Fab Lab is typically equipped with digital fabrication machines, such as 3D printers, laser cutters, high-resolution milling machines, computer numerical control (CNC) machines and vinyl cutters, as well as electronics workbenches for prototyping circuits and programming microcontrollers.

In this study, we focus on three cases of school visits where students from 7th to 9th grades engaged in digital fabrication activities by creating physical artefacts in Fab Lab Oulu in October and November 2016. Table 1 describes the participants and the duration of the activities in the three cases. All the projects in the three cases were implemented as collaborative projects, where two to five students worked together on one project as a group. In the case I (School A), the participating students had a total freedom of what to make with only a few requirements, such as using a microcontroller. In the case II (School B), the students decided what to make based on the theme of "100 years of Finnish independence" provided by the teachers. In the case III (School C), the group of students visited the Fab Lab as part of their ongoing project: designing models of a playhouse at the school.

In all the three cases, the activities were run by two facilitators working in Fab Lab Oulu. One of them is a specialist in electrical engineering, and the other one in ubiquitous computing and human computer interaction. The facilitators' main role was to provide instructions on the basic operations of the facilities and digital tools and to help the pupils when they had problems in the process. The school teachers' role was mainly to observe the activities and general schedule management.

Table 1. Participants and project details of the three cases.

Participants & project details	Case I: School A	Case II: School B	Case III: School C
Number of participants	12 students, 1 teacher	20 students, 2 teachers	9 students, 2 teachers
Students' grade	9th	7th - 8th	9th
Duration	5 days	3 days	5 days
Projects	Useless box, Rail for camera, Electronic controlled lock, Jukebox game, Music car	Finland 100 years calendar, Finland 100 years history wheel, Finland flag day clock	Two models of playhouse
Required conditions	Use Arduino Uno board and at least one actuator, fabricate mechanics using laser cutter or 3D printer, make functional artefact in 5 days	Use Arduino Uno board, fabricate mechanics using laser cutter	A competition between two teams designing a playhouse for the school community
Designing software	Inkscape, Autodesk TinkerCad	Inkscape	Inkscape, SketchUp
Machines used	Laser cutter, 3D printer Laser cutter	Laser cutter, vinyl cutter, sewing machine	
Electronics	Arduino Uno board, servos, buttons, piezoelectric buzzer	Arduino Uno board, servos	
Programming	Arduino Software (IDE)	Arduino Software (IDE)	

4.2. Data Collection and Analysis

We collected data through two semi-structured focus group interviews. A focus group interview is a special type of interviewing technique, which provides insights into how people think and gives a deeper understanding of the phenomena being studied (e.g., Morgan, 1997; Puchta & Potter, 2004). We chose the focus group interview as a data collection method 1) to gather individual's perspective and interpretation of experience and 2) to provoke participants' thoughts and enhance discussion by sharing ideas.

In the focus group interview I, the interviewees were three teachers from two schools (School A and School C), who participated in the activities with their students (see Table 1). One of them works as an advisory teacher, while other two teachers work as subject teachers including the fields of chemistry, physics, mathematics, ICT, biology and woodworking. The two facilitators who ran the activities in the three cases participated in the focus group interview I as observers. Discussion followed the questions regarding their observations and experiences of the digital fabrication activities, (e.g., *which aspects in Fab Lab activity were meaningful for the students' CT process?*).

For the focus group interview II, we invited the facilitators who ran the activities. The interview questions regarded their perspectives and experiences during the activities (e.g. *how was CT seen in the activities?*), as well as observation of the focus group interview I with the teachers (e.g., *what did you notice in teachers' answers, what are lacks/ shortcomings of educationalists?*). The interviews were recorded in video and audio.

For data analysis, we employed a theory-driven approach following the guidelines introduced by Krueger and Casey (2000). First, we transcribed the recorded two focus group interviews. Based on the predetermined codes, six CT practices defined by Barr et al. (2011), we examined the teachers' and facilitators' perspectives towards their experiences in digital fabrication activities. We used NVivo in the coding process and to test the reliability of coding. The score of overall inter-rater agreement, Cohen's Kappa coefficient (Cohen, 1960), was $k = 0.80$.

The unit of analysis in this study is the institutional level. According to Lewis-Beck, Bryman, and Liao (2004), the unit of analysis is the subject of the study about which an analyst may generalize. In this study, we intend to highlight the perspectives of groups of people, teachers and facilitators, rather than those of individual participants. Moreover, to avoid any possibility to identify individual participants, we refrain from specifying individual participants, referring to them in general terms such as "one of the teachers" and "one of the facilitators".

5. RESULTS

The school teachers and facilitators recognized CT practices defined by Barr et al. (2011) in the digital fabrication activities. Table 2 summarizes the identified CT practices in the two focus group interviews.

Table 2. CT practices identified in focus group interviews.

	Focus group interview I (n[a]=8,387)		Focus group interview II (n=6,328)	
CT practices (Barr et al. 2011)	CC[b]	n[c]	CC	n
1) Formulating problems in a way that computer and other tools can help solve them	45.8%	432	17.8%	147
2) Logically organizing and analyzing data	18.1%	171	28.9%	239
3) Representing data through abstractions	0.0%		0.0%	
4) Automating solutions through algorithmic thinking	7.1%	67	17.8%	147
5) Identifying, analyzing, and implementing possible solutions	29.0%	274	35.5%	293

with the most efficient and effective combination				
6) Generalizing and transferring this problem- solving process	0.0%		0.0%	
Total	100%	944	100%	826

[a] Total number of words in the focus group interview.

[b] Coding coverage: percentage of the number of words coded at the node.

[c] Number of words at the node.

5.1. Developing CT Practices through Frequent Use of Computers in Digital Fabrication

Among the six CT practices defined by Barr et al. (2011), the teachers and facilitators most frequently discussed formulating problems to allow computers and other digital tools to help solve them. In the digital fabrication activities, the students used computers for design and fabrication. One of the facilitators explained an example as follows:

"You have to use a vector graphics program to prepare your file in a certain way, in a form that the laser cutter can process…. it's an algorithm you have to follow and also do CT, you have to perform certain steps in a certain order to get the result."

The students designed artefacts on computers to fabricate them with digital fabrication machines, such as the laser cutter, the vinyl cutter and the 3D printer. They had to make the design files in a specific format to use digital fabrication machines. Also, the students in School A and School B converted the functionality of the artefacts, such as playing the various songs in a jukebox by pressing a button (a project in School A), into codes in Arduino software to allow the microcontrollers to execute those functions (see Table 1).

5.2. Developing CT Practices through Complex Problem-Solving in Digital Fabrication

In the digital fabrication activities, the students were engaged in complex problem-solving. The complexity they needed to confront involved the functions of the artefacts and the procedures of making. One of the facilitators explained,

"Computational thinking is best applied to a little bit larger design problems, when you really have to divide your work into pieces that you need to solve piece by piece."

Not only by using computers in the process, but the students also had opportunities to develop CT practices by thinking logically and implementing solutions efficiently and effectively. For instance, one group in School B needed to understand the complex mechanics of automatically raising the flag on national flag days in terms of separate steps, such as defining the national flag days, moving the servo in a specific direction to raise the flag, and moving the servo to locate the flag in the original position (see Table 1). One teacher explained how the students used logical thinking when considering the functions of the artefacts as follows:

"The whole project was pretty much about thinking logically, all the hardware staff, you had to think that when this part is moving that way, it moves the other one in the opposite direction, and stuff like that. So…. you had to think logically to get it working."

One group in School A had to redesign the artefacts, because they realized that the design of the external box was too small for the mechanics which were supposed to be placed inside the box. One of the facilitators illustrated as follows the way in which the students needed to consider what kind of procedure would be efficient in a complex fabrication process:

"You first need to think about the external design, then how to put in the mechanics, followed by how the mechanics is going to work…. So this entire process is a step by step thing, and I think in that sense it is about computational thinking. "

5.3. Teachers' and Facilitators' Perspectives on CT Practices

The facilitators indicated that digital fabrication involves the concept of CT in its process. One of them explained it as follows:

"Any process in Fab Lab requires this way of thinking [CT]: go through these logical steps. For example, if you want to make a printed circuit board or milled circuit board, you have to follow certain steps, and it is about computational thinking, it's an algorithm you have to follow."

However, the facilitators also indicated that the school teachers may not be familiar with the concept of CT. One of the facilitators said,

"Basically, engineering is about computational thinking. But at schools, I don't think teachers have this built-in curriculum to follow."

On the other hand, the facilitators mentioned their unclear understanding as regards the concept of CT. One of them said, "I don't know. We don't actually know, we guess they are learning CT. But we don't know they really are."

6. DISCUSSION

The findings suggested that the nature of STEM-based digital fabrication, frequent use of computers and complex problem-solving, enhances the development of CT practices described by Barr et al. (2011). Frequent use of computers in the process, which naturally occurs in the process of digital fabrication, allows the students to convert their problems into certain formats to employ digital fabrication methods. These problems included design problems (e.g., visualizing the designs in 2D and 3D modeling software) and functional problems (e.g., executing the functions with microcontrollers by writing code in Arduino software).

The complexity of digital fabrication activities may increase the potential for developing CT practices, especially thinking logically and implement solutions effectively. In digital fabrication activities where the students created the tangible artefacts, they faced complex problems with mechanics. The students had to decompose the complex mechanics problems into small manageable steps considering the possible tools (e.g., the microcontroller, servo and button) and the logical order to make the tools function. Also, the students applied CT practices in the process of planning and implementing efficient and effective procedures of fabrication. While repeating the design process including design, prototyping, analyzing and re-design, the students had to consider the whole process of fabrication and choose the constructive procedures, such as designing the mechanics first, and designing the external box.

However, we found that both the teachers and the facilitators could be vague in their definition of the concept of CT. The facilitators indicated that school teachers were perhaps not familiar with the concept of CT. In fact, in the focus group interview with the teachers, the discussion focused on visible actions in the use of computers, such as designing on the computer, rather than implementation of the fundamental concepts of computing: abstractions and automation, which, according to Wing (2008), underlie CT. One of the challenges for teachers to implement CT practices in classrooms might be to familiarize themselves with the fundamental concepts of computing that constitute the core of CT practices.

This finding is in line with the result of the survey among school teachers conducted by Mannila et al. (2014). They did not find many relevant answers in the teachers' survey about utilizing abstractions and automation in classrooms, suggesting that very few teachers connect abstractions and automation with their implementation of CT practices in classrooms. It may be inconsequent to implement CT practices without internalizing the fundamental concepts of computing on which CT is deeply rooted.

On the other hand, we uncovered that the facilitators might not have a clear definition of CT and CT practices, even though CT is a basic way of thinking in their fields of expertise: computer science and engineering. As CT is already part of their built-in thinking and working processes, the facilitators might not have perceived it as something that the students can develop through digital fabrication activities.

As a limitation of this study, we are aware that the sample size for the focus group interviews was small. In order to avoid disclosing the individual participants' opinions, we refrained from identifying individuals in the results section, which may affect the trustworthiness of the study. In addition, we could have described the concept of CT in a more detailed manner during the focus group interviews. Although we provided an explanation of CT before starting discussion in the focus group interviews, it is likely that the interviewees might not fully understand the concept to discuss it properly.

7. CONCLUSION

In this study, we aimed to explore the potential for developing CT practices through digital fabrication activities. We found that STEM-based digital fabrication activities may enhance developing CT practices. In agreement with the nature of digital fabrication, fabricating artefacts with digital technology, the students formulated the design problems and functional problems with digital tools so that computers and digital tools could be used to help to solve them. In addition, the complexity, such as the complex mechanics of the artefacts and complex procedures to fabricate, encouraged the students' skills in thinking logically and implementing solutions effectively. However, we also found that the teachers and facilitators could be lacking in their understanding of the definition of CT. To encourage the development of students' CT practices, both the school teachers' and facilitators' awareness of the concepts of CT is essential. Future research may examine the development of teachers' and facilitators' understanding and implementation of CT in more detail.

8. REFERENCES

Astrachan, O., & Briggs, A. (2012). The CS Principles Project. *ACM Inroads*, *3*(2), 38-42.

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.

Bevan, B. (2017). The Promise and the Promises of Making in Science Education. *Studies in Science Education*, *53*(1), 75-103.

Blikstein, P. (2013). Digital Fabrication and 'Making' in Education: The Democratization of Invention. In J. Walter-Herrmann & C. Büching (Eds.). *FabLabs: Of Machines, Makers and Inventors*, 203-222. Bielefeld: Transcript Publishers.

Blikstein, P., & Krannich, D. (2013). The Makers' Movement and FabLabs in Education. In *Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13)*, 613-616.

Blikstein, P., Martinez, S. L. & Pang, H. A. (2015). About the FabLearn Labs. In Blikstein, P., Martinez, S. L. & Pang, H. A. (Eds.). *Meaningful Making: Projects and Inspirations for FabLabs and Makerspaces*, viii-ix. Retrieved January 22, 2019, from http://fablearn.stanford.edu/fellows/sites/default/files/Blikstein_Martinez_Pang-Meaningful_Making_book.pdf

Cavalcanti, G (2013). *Is it a Hackerspace, Makerspace, TechShop, or FabLab*. Retrieved January 22, 2019, from https://makezine.com/2013/05/22/the-difference-between-hackerspaces-makerspaces-techshops-and-fablabs/

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, *20*(1), 37-46.

Denning, P. J. (2009). The Profession of IT Beyond Computational Thinking. *Communications of the ACM*, *52*(6), 28-30.

Fab Foundation (n.d.). *What Is A Fab Lab*. Retrieved January 22, 2019, from http://fabfoundation.org/index.php/what-is-a-fab-lab/index.html

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

Hu, C. (2011). Computational Thinking: What it Might Mean and What we Might Do about it. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education.* ACM, 223-227.

Kafai, Y. B. (2016). From Computational Thinking to Computational Participation in K-12 Education. *Communications of the ACM*, *59*(8), 26-27.

Kafai, Y. B., Lee, E., Searle, K., Fields, D., Kaplan, E., & Lui, D. (2014). A Crafts-oriented Approach to Computing in High School: Introducing Computational Concepts, Practices, and Perspectives with Electronic Textiles. *ACM Transactions on Computing Education (TOCE), 14*(1), 1.

Krueger, R. A. & Casey, M. A. (2000). *Focus Groups: A Practical Guide for Applied Research* (3. ed.). Thousand Oaks, CA: Sage.

Lewis-Beck, M. S., Bryman, A., & Liao, F. T. (2004). *The SAGE Encyclopedia of Social Science Research Methods.* Thousand Oaks, CA: Sage Publications, Inc.

Martin, L. (2015). The Promise of the Maker Movement for Education. *Journal of Pre-College Engineering Education Research (J-PEER), 5*(1), 30-39.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational Thinking in K-9 Education. In *Proceedings of the Working Group Reports of the 2014 on Innovation & Technology in Computer Science Education Conference*, 1-29.

Montero, C. S. (2018). Facilitating Computational Thinking through Digital Fabrication. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research.* ACM, 32

Morgan D., L. (1997). Focus Groups as Qualitative Research. *Qualitative Research Methods Series*, *16*(2).

Papert, S. (1996). An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Papert, S. & Harel, I. (1991). Situating Constructionism. In I. Harel & S. Papert (Eds.), *Constructionism*, 1-11. Westport, CT: Ablex Publishing Corporation.

Pitkänen, K., Iwata, M., & Laru, J. (in press). Supporting Fab Lab Facilitators to Develop Pedagogical Practices to Improve Learning in Digital Fabrication Activities. In *Proceedings of the Conference on Creativity and Making in Education.* ACM.

Puchta, C., & Potter, J. (2004). *Focus Group Practice*. London: SAGE Publications Ltd,

Rode, J. A., Weibert, A., Marshall, A., Aal, K., von Rekowski, T., El Mimouni, H., & Booker, J. (2015). From Computational Thinking to Computational Making. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 239-250

Wing, J. (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 of London A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

Wing, J. M. (2010). *Computational Thinking: What and Why*. Retrieved January 22, 2019, from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

A Goal Analysis of Computer Science Education:

Setting Institutional Goals for CS Ed

Stephanie B. WORTEL-LONDON[1], Leigh Ann DELYSER[2*], Lauren WRIGHT[3], Júlia Helena AGUIAR[4]
[1] Stony Brook University, New York, The United States
[1234] CSforALL, New York, The United States
stephanie@csforall.org, leighann@csforall.org, lauren@csforall.org, julia@csforall.org

ABSTRACT

Computer Science (CS) education efforts have primarily focused on the professional development of teachers as a mechanism to increase participation and broaden access. Although these efforts have produced many additional teachers with a preliminary knowledge of CS content and pedagogy, difficulties still exist when implementing new classes or encouraging peer teachers to also teach CS. In this paper, we detail a process piloted with teachers and administrators focused on goal setting around a framework of district implementation. Data is presented from workshops with 60 school systems. A text-based analysis of 1,023 goals illustrates district focal areas around goal setting and demonstrates a significant relationship between leadership strength and efforts to support teacher development. We offer examples of goals and considerations for further exploration around the ways in which districts structure goal setting statements as well as tactics employed by districts for accomplishing CS education implementation goals.

KEYWORDS

systems-change, district leadership, teacher development, SCRIPT, workshops

1. INTRODUCTION

The introduction of computer science (CS) education into K-12 classrooms in the United States is more than an exercise in teacher professional development and student learning, it is a systems-change endeavor as well. Over the last 5 years, cities and states in the United States have created plans for CS education; however, the work of individual school districts has largely been ignored (DeLyser & Preston, 2015; Ericson & Guzdial, 2014).

Large school districts such as New York, Chicago, Broward County, and San Francisco have either benefited from National Science Foundation funding or private donations that have allowed the districts to hire staff with a responsibility for CS (DeLyser & Preston, 2015). These staff members are able to strategize, create plans, and lead the execution of that vision with partners to increase teacher capacity for computer science instruction.

Smaller districts, however, may not have access to the same resources, but still need coordinated plans to implement CS across multiple grades, buildings, and classrooms. In the United States there are over 1,600 school districts, and 15,000 have fewer than 20 schools - creating a long tail of districts who will need to be supported to implement CS education.

School districts are a unique unit of change, as they serve multiple schools and grade levels, are of varying size (the authors have worked with districts as small as two schools and as large as 1,700), and are constrained by regional and state policies as well as set policies for themselves. In the ecosystem of CS education efforts in K-12, school districts are both a policy and fiscal center (McLaughlin, 1987). Often districts also set instructional priorities, dictate culture, and determine how to measure student and faculty success and growth over time.

The challenges and subsequent decisions made by districts can have a huge impact not only on whether CS is taught in a region, but also how CS is implemented, what teachers are selected to teach the subject, the frequency and depth of student instruction, and the multidisciplinary problem-solving activities that students engage in.

2. LITERATURE REVIEW

Computer science efforts in the United States have primarily focused on the development of curriculum, the development of teacher capacity through professional development, and the state level policy needed to support the expansion of CS as a new subject in schools. Although these efforts are crucial to the overall success of the goal of expanding CS education across the country, they do not fully represent the landscape of institutions that need to change in order to implement CS education equitably.

Equity in CS education cannot be implemented on a teacher-by-teacher basis. While the overall increase in the number of teachers prepared to teach CS in K-12 is one measure of progress, it is not a complete picture with regards to equitable implementation of CS education. While it is tempting to highlight schools with at least one teacher who has participated in professional development, one teacher alone cannot ensure CS literacy for all students within a school, let alone a district.

2.1. District Leadership and Education Implementation

General research into educational effectiveness and school reform highlights the importance of district leadership in reform efforts. Although teachers are a key component of CS education implementation, the professional isolation of teachers is a well-documented phenomenon (Flinders, 1988). From a position of isolation, it can be difficult for teachers to sometimes implement CS in their own classroom, let alone influence their peers to take on a new subject (Goode, 2008).

District leadership can play an important role in amplifying and multiplying the efforts of individual teachers. The involvement, and not just support, of district leaders in education reform initiatives can support the best practice of collaborative working groups of teachers, resulting in improved teacher practice and student learning. Dufour even goes so far as to state "No single person can improve student achievement in an entire district, school, or classroom" (DuFour & Marzano, 2011).

2.2. The Use of Goals

A key component in the sustainability of any initiative is the transfer of ownership from the directing organization to the organizations that will need to enact change (Coburn, Touré, & Yamashita, 2009). Goal setting activities promote the transfer of ownership, and the practice of goal setting is correlated with both high performing organizations in general, as well as school systems (Waters & Marzano, 2006). Using a goal-setting process is not only useful for encouraging ownership and problem-solving mindsets in school leadership, but also as a data collection methodology for research (Leithwood & Steinbach, 1995).

The analysis of goals is a long-held technique for identifying the needs of educational systems (Witkin, 1975). The goals can both be used to infer the focus of the organizational change as well as opportunities to impact the current system. In this paper, we use the goals set by districts to highlight targeted areas for improvement as well as provide a landscape for the CS education community for supportive efforts.

3. METHODOLOGY

The goals were collected as a part of an in-person workshop that was facilitated by the CSforALL team. The goal setting process was part of a larger agenda focused on expanding CS education efforts in the districts. The workshops were conducted in Yorktown Heights, NY; Ithaca, NY; Atlanta, GA; Nashville, TN; Boston, MA; Austin, TX; Phoenix, AZ; Detroit, MI, and St. Louis, MO.

3.1. District Team Profile

The workshops were advertised through CSforALL networks as well as through outreach to local influencers including prominent CS education community members in each region. Districts were encouraged bring a team of at least three, including a district level administrator, school building administrator, and instructional staff who have or anticipated having computer science teaching roles. This could be a stand-alone teacher from a high school, an elementary teacher who conducted CS or computational thinking activities in their classroom, or a library media specialist. Additionally, districts were encouraged to diversify the teams by grade band, including representatives from elementary, middle, and high schools.

Table 1. District Participation Profiles

Metric	Min	Max	Mean	Median
Students K-12	167	387,311	22,204	5,777
Schools	1	535	35.4	9.5
Team Size	1	12	4.9	5
Students on Free or Reduced Lunch	0 %	100%	49.9%	50.9%

Table 1 shows the characteristics of the districts who participated in the workshops.

During the workshop, districts participated in a larger process where they identified a vision for CS education for their particular student body, self-assessed their current implementation against rubrics, were prompted to set an unspecified number of goals for 3 months, 6 months, and long-term improvement plans, shared their work with the other districts present, and heard presentations about resources from the locally-, regionally- and nationally-based CS education community.

3.2. Rubric Data

In this paper, we focus explicitly on the goals set by the districts and therefore will describe the specific prompts leading to the goal setting activities. Depending on the length of the workshop as well as the developmental stage of the rubric components (due to the pilot year), district teams were asked to focus on 3-5 components of the organizational change tool, the SCRIPT Rubric (https://www.csforall.org/projects_and_programs/script/). Each component included rubrics that were constructed to specifically focus on CS education implementation and integrated models of best practice for educational leadership and teacher development. The areas were: Leadership, Materials and Curriculum Selection and Content Refinement, Teacher Capacity and Development, Partners, and Community. Each rubric area contains a series of sub-components for which districts rated themselves as Novice (Score of 1), Emerging (2), Developing (3), or Highly Developed (4).

Table 2. Average Rubric Scores

Area	Number of Districts	Min	Max	Mean
Leadership	58	1.00	3.67	1.98
Materials & Curriculum	51	1.14	3.57	1.81
Teacher Capacity	58	1.00	3.50	1.76
Partners	19	1.00	3.67	1.92
Community	19	1.00	3.25	1.75

Table 2 has the average rubric scores for the district teams who participated in the workshops and who also completed the goal setting activities. As indicated by the data, the participating districts were in varying places in their progress towards implementing CS education, and while they may not represent a fully generalizable picture of the United States, they do comprise a large and diverse sample of the districts in the US.

3.3. Goal Setting Process

During the workshop, district teams--with guidance from the workshop facilitators--approached the areas of reform one topic at a time. For example, all districts were given an hour to work specifically on Leadership. Each area of reform included: (1) targeted self-reflection for each rubric area followed by submission of rubric ratings through an online form, (2) open ended reflection, including identifying rubric content that stood out as well as areas of strength and growth for the district, (3) a resource review, and (4) time-bound goal setting for the respective rubric area, during which the teams were asked to create an unspecified number of 3-month, 6-month, and long-term goals for that area. At the conclusion of the workshop, districts compiled all time-bound goals into one comprehensive 3-month, 6-month, and long-term goal setting plan, re-prioritized items based on calendars and progression, and were asked to submit their final goals through an online form. Table 3 shows a breakdown of the number of goals related to each focus area on the rubric, as well as the range of ratings each district applied within the rubric areas.

Table 3. Percent of Goals Set by Rubric Area

Area	Percent of Districts Setting	Min	Max	Mean
Leadership	96.67%	0.00%	62.50%	36.26%
Materials & Curriculum	83.33%	0.00%	50.00%	24.51%
Teacher Capacity	96.67%	0.00%	100%	31.71%
Partners	31.67%	0.00%	34.78%	6.48%
Community	8.33%	0.00%	18.37%	1.05%

4. PRELIMINARY DATA

As a result of the goal setting process, 1,023 goals were collected from 60 school districts who participated in workshops from July 2017 through June 2018. In this section, we describe methodology for categorizing goals, and compare goal categorization as defined by the districts with categorization as defined by four raters within CSforALL, as well as significant findings around goals within each rubric area.

4.1. Goal Labeling

The 1,023 goals were anonymized and the district's categorization by rubric area was hidden. Additionally, the goals were re-sorted to a random order for consideration of each goal as an individual data point. A team of four raters coded each goal with one of the five rubric areas. Examples of goals and their rubric area categorizations include: Leadership: "Create digital and physical incentives for schools to display their status as a CS school"; Teacher Capacity and Development: "By March 2019, at least 2 elementary CS resource teachers, 1 middle school CS resource teacher, and 1 high school CS teacher will be identified and trained to deliver CS instruction and coaching to teachers in CS in summer 2019 for the 2019-2020 school year".

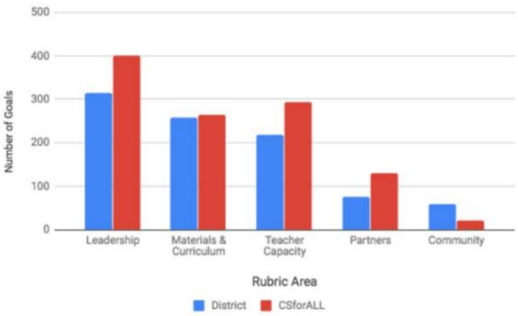


Figure 1. District-Rater categorization by rubric area.

To establish inter-rater reliability, a training set of 50 randomly selected goals was selected and categorized by each rater independently. The rating group then reviewed the ratings in the training set and agreed upon rules and norms for further categorization. Raters then independently categorized an additional 100 goals, with perfect agreement among the four raters in 61 percent of goals. In the remaining 39, there was high inter-rater agreement for three out of four raters in each goal. Raters were then each randomly assigned 224 of the remaining 896 goals to code.

4.2. Goal Categorization

After the goals were coded by each of the raters, we compared the agreement between districts' and raters' categorization, demonstrated in figure 1. We predict that the largest discrepancy in agreement, which was in the Leadership rubric area, was due to the rater agreement that goals related to landscaping the current ecosystem within the district, regardless of the target audience, would be considered a Leadership goal. An example of this occurrence is as follows: "Identify teachers who would benefit from attending PD, bring back and share," which was categorized as a Teacher Capacity and Development goal by the district but as a Leadership goal by the raters.

4.3. Rubric Focal Areas

In an effort to understand how districts set goals around CS education implementation, specifically whether they were focusing on areas of strength or weakness in initial goal setting, we compared the average rubric scores by area for each school district to the number and categorization of goals set during their initial goal setting. The following is a example of the comparison for one school district: Leadership: Avg Rubric Rating - 1.83, Percentage of goals: 36.11%; Teacher Capacity: Avg Rubric Rating - 1.50; Percentage of goals: 19.44%.

In order to determine if there was a significant correlation between the focus of the districts, as determined by the relative number of goals districts set in each of the rubric areas, a one-way ANOVA was used. The percentage of goals set for each area was used as the dependent variable, and the districts rubric averages were the independent variable in separate comparisons.

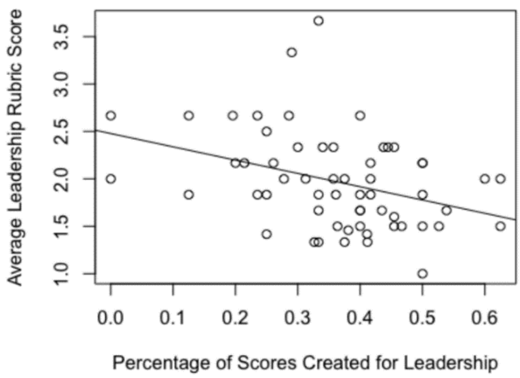


Figure 2. Leadership goals set as a percentage.

There was a significant relationship between the relative number of goals set for Leadership ($F(1,55)=8.386$, $p=0.005$) and Teacher Capacity ($F(1,55)=5.162$, $p=0.027$) and the average district self-assessed rubric score for

Leadership. There was no significant relationship between the relative number of goals for any other rubric area and the rubric scores the districts set.

Figures 2 and 3 shows the plot of the percentage of goals set by the districts for two sections of the workshop compared to the rubric scores districts recorded for leadership. As shown by the regression line, there was an inverse relationship between the average leadership rubric score and the percentage of goals created focused on leadership, and a direct relationship between the average leadership rubric score and the percentage of goals focused on Teacher Capacity.

These preliminary data demonstrate that district teams that have stronger Leadership ratings are more focused on the creation and execution of teacher-centric goals, echoing the aforementioned general research around educational effectiveness and school reform that highlights the importance of district leadership in reform efforts. A further analysis is needed to examine if the combination of a strong Leadership rating and a substantial focus on Teacher Capacity and Development as a part of goal-setting positively impacts the attainment of Teacher Capacity goals, and if so, if this attainment shows demonstrable progress within the Teacher Capacity rubric (i.e. from Novice to Emerging).

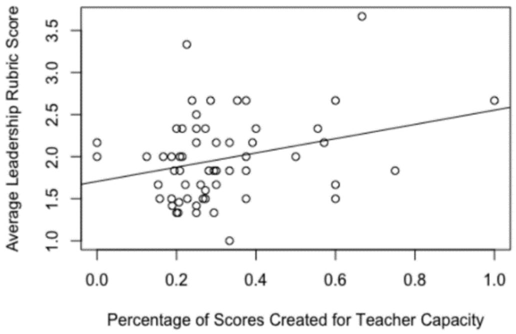


Figure 3. Teacher Capacity goals set as a percentage.

5. DISCUSSION

In this paper we presented an analysis of 1,023 goals set by school districts. Overall, we see districts focusing on the Leadership of CS education efforts in their district, the selection of Materials and Curriculum and the development of Teacher Capacity.

5.1. Including Leadership in CS Education Efforts

Interestingly, and aligned with other research in educational reform, the self-assessed rating for the involvement of leadership in CS education efforts was significantly correlated with an increase in goals for teacher capacity. This could indicate that teams with strong leadership felt capable to move on to Teacher Capacity as a focus. These findings could indicate a need for the CS education community to include school and district leadership in research and implementation efforts.

5.2. Future Work

CSforALL team will continue to follow up with districts at the stated intervals (3-months, 6-months) in order to determine if goals that are set are being met and if districts find greater success in some rubric areas over others, and if goals were not attained, identify specific impediments to attainment. Additionally, the examination of the strategies employed by districts to accomplish goals is necessary to inform the CSforALL team and the greater CS education community about how best to support districts in their implementation efforts.

6. REFERENCES

Coburn, C. E., Touré, J., & Yamashita, M. (2009). Evidence, Interpretation, and Persuasion: Instructional Decision Making at the District Central Office. *Teachers College Record*, *111*(4), 1115-1161.

DeLyser, L. A., & Preston, M. (2015). A Public School Model of CS Education. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*. The Steering Committee of the World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 223.

Dettori, L., Yanek, D., Hu, H., & Brylow, D. (2018, February). The Role of Researcher-Practitioner Partnerships in CS4All: Lessons from the Field. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* ACM, 674-675.

DuFour, R., & Marzano, R. J. (2011). *Leaders of Learning: How District, School, and Classroom Leaders Improve Student Achievement*. Solution Tree Press.

Ericson, B., & Guzdial, M. (2014). Measuring Demographics and Performance in Computer Science Education at a Nationwide Scale Using AP CS data. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*. ACM, 217-222.

Flinders, D. J. (1988). Teacher Isolation and the New Reform. *Journal of Curriculum and Supervision*, *4*(1), 17-29.

Goode, J. (2008). Increasing Diversity in K-12 Computer Science: Strategies from the Field. In *ACM SIGCSE Bulletin*, 40 (1), 362-366.

Leithwood, K., & Steinbach, R. (1995). *Expert Problem Solving: Evidence from School and District Leaders*. SUNY Press.

McLaughlin, M. W. (1987). Learning from Experience: Lessons from Policy Implementation. *Educational Evaluation and Policy Analysis*, *9*(2), 171-178.

Waters, J. T., & Marzano, R. J. (2006). *School District Leadership that Works: The Effect of Superintendent Leadership on Student Achievement.*

Witkin, B. R. (1975). *An Analysis of Needs Assessment Techniques for Educational Planning at State, Intermediate, and District Levels.*

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Research on the Current Situation and Development Trend of Computational Thinking in K-12 Education in China

——Keywords Co-Word Analysis Based on Knowledge Map

Jue WANG[1], Yi ZHANG[2*], Xing LI[3], Qiang REN[4], Lin MEI[5]
[12] School of Educational Technology, Central China Normal University, China
[14] School of Teacher Education, Huzhou University, China
[3] School of Education, Jianghan University, China
[5] Research Institute for Open Education, Hubei Radio and Television University, China
wangjue@zjhu.edu.cn, zhangyi@mail.ccnu.edu.cn, lixing1130@mails.ccnu.edu.cn, renq@zjhu.edu.cn, meilin222@126.com

ABSTRACT

The Computational Thinking (CT) skills have developed and fostered in K-12 education in China. Therefore, it is of great significance to know about the current situation and future development of CT. This paper took the keywords of journals, master's and doctoral' s dissertations on CNKI of CT in K-12 education in China as the research objects, and used the keywords co-word analysis method of Knowledge Map to research the characteristics and relationships of related fields. At present, the research fields of CT focus on: curriculum standards, teaching models and instructional design, educational practice based on robot programming, case study of open source programming, cultivation of CT in elementary school mathematics, CT education in senior high school. Generally speaking, the cultivation of CT in K-12 education in China is at the initial exploratory stage. The content in the first quadrant of the Knowledge Map is the focus of current research, while the contents in the second and third quadrants highlight the potential for future development. Primary school is in the center of coordinates. Makerspace education, STE (A) M and Curriculum Standard keywords are closely related and close to abscissa. The teaching mode and strategies are close to the ordinate etc. The paper revealed the hotspots and development trend of research field of CT, which will provide implications for future development in K-12 education in China: constructing interdisciplinary curriculum standard and frame of CT for different level of K-12 education; developing teaching approach and strategies of integrating the core elements of CT; exploring more empirical study of CT formative and summative evaluation and assessment approach.

KEYWORDS

computational chinking, k-12, knowledge map, keywords co-word analysis, development trend

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

我国 K-12 计算思维的现状审视与发展趋势研究

——基于知识图谱的关键词共词分析

王珏[1]，张屹[2*]，李幸[3]，任强[4]，梅林[5]
[12]华中师范大学 教育信息技术学院，中国
[14]湖州师范学院 教师教育学院，中国
[3]江汉大学 教育学院，中国
[5]湖北广播电视大学 导学中心，中国
wangjue@zjhu.edu.cn，zhangyi@mail.ccnu.edu.cn，lixing1130@mails.ccnu.edu.cn，renq@zjhu.edu.cn，meilin222@126.com

摘要

为了解我国 K-12 计算思维现状，文章以中国知网的期刊和硕博士论文关键词为数据来源，采用知识图谱的关键词共词分析法研究热点。目前计算思维研究领域主要集中在：课程标准、教学模式与教学设计、机器人编程教学实践、开源编程教学案例、小学数学计算思维培养、高中计算思维教育。知识图谱第一象限内容是当前研究的重点，二三象限具有未来的发展潜力。小学是重要的培养阶段，课程标准与创客、STE（A）M 教育结合紧密，且对其他领域的影响大。教学模式与策略在领域内具有较高的联系强度。研究结果为 K-12 计算思维的发展趋势提供相应的启示。

关键字

计算思维；k-12；知识图谱；关键词共词分析；发展趋势

1. 前言

“计算思维”是将计算机科学的抽象，分解，建模和算法等的思想、方法扩展到其他学科，形成问题解决的一般化方法，它的界定对 k-12 教育的重要性体现了它是培养思维和解决问题的方法。《地平线报告（基础教育版）》指出，问题解决是计算思维的核心，任何课程都可以培养学生的计算思维能力（NMC, 2017）。因此，计算思维已经成为人才培养的重要方面，在我国 K-12 教育中得到了极大地重视。因此，全面审视我国 K-12 教育中关于计算思维的研究现状，对未来发展具有重要意义。本文选取了中国知网（CNKI）收录的期刊、硕博士论文的关键词作为分析材料，通过建构知识图谱的研究过程来了解我国中小学计算思维研究领域的热点、特征与发展趋势。

2. 研究方法

本研究采用知识图谱的关键词共词分析法，通过表达核心内容的关键词在文献中共同出现的频次，揭示研究主题以及主题间的相互关系。采用 Bicomb，SPSS 等软件进行数据处理与分析。其中，Bicomb 为书目共现分析系统，用于处理从文献数据库下载下来的文献记录。选择中国国家知识基础设施（CNKI）的期刊和论文作为数据来源，将检索主题设置为“计算思维”，检索时间从 1996 年（麻省理工学院 Papert 教授最早提出计算思维概念的时间）至 2018 年。本次研究的搜索时间为 2018 年 7 月 6 日。Bicomb 软件可以从文档记录中识别出 ANSI 编码的文本文件，每条记录包含来源库、题名、作者、单位、文献来源、关键词 6 类信息。在检查和删除不符合要求的记录后，最终得到 441 条有效数据（含学位论文的 43 条记录）。然后，标准化编码文件中关键词的内容和格式。将不同期刊来源的关键词进行审查，对词义相近的关键词进行同义合并等，如核心素养、学科核心素养合并为（学科）核心素养。最后对文本进行格式标准化，保存为 ANSI 编码文件。

该研究可分为以下几个阶段：1.在 Bicomb 中确定高频关键词，建立共词词频矩阵。2.在 SPSS 中构建聚类图，生成相似系数矩阵。3.通过 Excel 转换成相异系数矩阵。4.对相异系数矩阵进行多维尺度分析。5.绘制知识图谱。

3. 研究结果与分析

3.1. 高频关键词词频统计

高频关键词表征文章的引用频次，体现文章的学术水平与价值。根据普赖斯计算公式 $M=0.749\sqrt{N\max}$ 计算得出研究对象的高频关键词频次阈值。其中，Nmax 表示区间学术论文被引频次，M 为高频域值（王佑镁和伍海燕，2012）。其中，被引用频次最高的是：“计算思维：信息技术课程的一种内在价值”一文（李锋和王吉庆， 2013），该文从发表至 2018 年 7 月 6 日，已被引用 63 次。根据 M 值计算高频关键词的频次阈值约为 6。故选取频次阈值>=6 的 33 个关键词作为高频关键词，共占总频次的 57.48%。其中，前 8 位关键词依次为计算思维（17.24%）、信息技术（计算机）（7.67%）、信息技术学科（课程）（4.08%）、高中（3.97%）、（学科）核心素养（3.53%）、信息技术教学（3.25%）、Scratch（1.74%）、信息素养（1.52%）。由此可以看出，目前与计算思维研究相关程度最高的关键词有信息技术（计算机），多以在高中信息技术学科（课程）中作为核心素养进行培养。

3.2. 关键词聚类分析

将计算思维的高频关键词词篇矩阵文档导入 SPSS 软件中进行系统聚类统计，聚合具有相似距离和相似度的关键词。关键词聚类分析结果：领域 1（小学数学、计算数学、计算能力）。领域 2（课程标准、美国、基础教育）、领域 3（教学设计、初中）、领域 4（Scratch、算法思维、创客教育与 STE（A）M、教学模式）、领域 5（教学实践、机器人、算法与程序设计）、领域 6（App Inventor、教学案例）、领域7（编程教育、人工智能、信息技术教育、中小学、计算机科学教育、计算思维教育、计算思维、信息技术（计算机）、高中、信息技术教学、信息技术学科（课程）、（学科）核心素养、信息素养、培养、小学）、领域 8（培养策略）。上述八个领域可归纳为六个主要方面：

3.2.1. *面向 K-12 的课程标准研究*

目前我国高中阶段有计算思维的课程标准，而在小学及初中阶段还未建立相应的标准。有学者对美国《K-12 计算机科学框架》中计算思维的核心概念、核心实践、框架特征等进行解读（赵蔚、李士平、姜强等，2017），从框架内容看，美国已经形成了涵盖 K-12 阶段较为系统的标准体系，对我国计算思维标准的建立具有参考价值。

3.2.2. *教学模式与教学设计研究*

领域 3 为初中教学设计研究，领域 4 为 Scratch、教学模式等的研究。类 1 是以 Scratch 为载体的教学模式研究，Scratch 具有“低门槛”和“高级天花板”的特性，提升学生的计算思维可视化表达能力。中小学信息技术教师注重将计算思维的抽象、分解、纠错等核心要素融入到 Scratch 教学过程中去，通过自主探究、小组协作和绘制流程图进行 Scratch 教学实践研究。类 2 是创客教育、STE（A）M 的教学模式研究。目前此类教育涉及机器人产品套件，以图形化编程设计为载体，使用基于项目/问题的学习，基于设计的迭代方式，培养学生逻辑思维等的高阶思维能力和创新能力。

3.2.3. *基于机器人编程的教学实践研究*

领域 5 为机器人、教学实践、算法与程序设计。机器人教育可以更好地发展学生计算思维等 21 世纪科技素养。信息技术课程标准中，“算法与程序设计”是高中的选修模块，实践表明，将机器人软件的编程方法、算法与程序设计进行关联，能让学生学到程序设计和机器人的相关知识，教学实践活动体现以计算思维的建构模型、论证等核心要素。

3.2.4. *基于开源编程的教学案例研究*

领域 6 包含 App Inventor、教学案例。App Inventor 提供 Android 编程环境，通过创意和代码拼接帮助学习者开发应用程序。相关学者提出了以 App Inventor 为学习工具，结合可视化编程界面，设计中小学信息技术课程培养计算思维能力的教学模式（郭守超、周睿、邓常梅等，2014；郁晓华、肖敏、王美玲等，2017）。

3.2.5. *基于核心素养培养的高中计算思维教育*

领域 7 包含有编程教育、人工智能、信息技术教育、中小学等多个关键词。细分为 5 个子类。其中，人工智能的重要性日益显现，我国在中小学课程标准中提出增加并充实“人工智能和编程课程内容”的明确要求。类 5 为小学。领域 8 为计算思维培养策略研究。

3.2.6. *小学数学计算思维能力培养研究*

数学包含了概念、抽象、模式、结构、算法等计算思维共有的认知模式。随着可视化编程工具的普及，在中小学课堂上利用编程工程解决数学问题成为创新的教学模式，促进计算思维能力的发展。

3.3. *关键词多维尺度分析*

Bicomb 生成频次阈值为[6,317]的词篇矩阵，在 SPSS 软件中打开该矩阵，将其转化为 33×33 的共词相似矩阵。再通过 1 减去相似矩阵数值生成相异矩阵。其他关键词和计算思维之间的相似度从高到低为：信息技术（计算机）（0.503）、信息技术学科（课程）（0.665）、信息技术教学（0.688）、高中（0.693）、Scratch（0.752）、（学科）核心素养（0.791）。表明计算思维研究中将信息技术（计算机）、信息技术学科（课程）和信息技术教学结合研究的概率较其他关键词大。将其导入 SPSS 进行多维尺度分析，通过空间、距离展示文献之间的联系和相似性，得到 Euclidean 距离模型平面图。多维尺度绘制的二维坐标中横轴表征领域间相互影响的程度（向心度），纵轴表征领域内的联系强度（密度）。其中离坐标中心点越近的点其影响力最大。基于多维尺度分析图和聚类分析，绘制出计算思维研究的的知识图谱，如下图 1 所示。

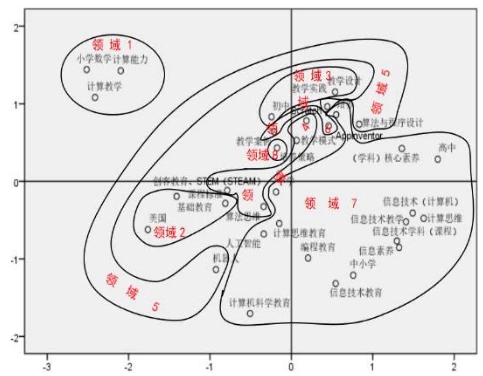


图 1 计算思维研究热点的知识图谱

位于第一象限的领域 3、领域 4、领域 5、领域 6、领域 7 关键词间自身内部连接紧密，具有较高的密度和向心度，是国内中小学计算思维研究的重点与热点，其研究成果目前处于中心地位，且各领域与其他研究领域之间的联系也很紧密。领域 7 跨越多个象限，处于第一象限的高中、（学科）核心素养研究成果较多，研究相对较为成熟。教学模式与教学设计、教学实践、算法与程序设计、编程工具、高中的核心素养等是研究的热点。第二象限领域 1 小学数学计算教学主题词内部联系紧密，但它与该象限内的其他关键词距离较远，说明数学学科的计算能力培养尚未与计算思维的其他

领域建立紧密的联系。在问题解决、建模和分析等领域，数学与计算思维有着众多的联系，如何在数学学科中与计算思维的核心要素进行关联是重要的研究内容。因此，未来的数学学科将从以往侧重于记忆和计算能力培养转向基于批判、创造性思维、计算思维的能力培养。

第三象限领域 2（课程标准、基础教育、美国）内部关键词间联系紧密。它与领域 4（创客教育、STE（A）M、算法思维）、领域 5（教学实践、机器人、算法与程序设计）、领域 7（小学、计算思维教育、人工智能）关键词间的耦合度高。以机器人等新兴技术为载体，可将计算思维融入到创客教育、STE（A）M 等课程中进行培养。处于第四象限的领域 7 涉及信息技术（计算机）、计算思维、信息技术教学、信息技术学科（课程）等关键词相互之间的联系紧密。从目前的发展来看，原有的软件操作技术不能适用计算思维培养的发展模式，需进一步探究计算思维与信息技术课程的关系以适应这种变革。有学者提出了计算思维应用于信息技术课堂的重要意义与实施的可行性（任友群、隋丰蔚和李锋，2016）。通过计算机科学的定义问题、提出设想、设计原型和测试改进等问题解决过程对信息技术的教学内容与流程进行重组。编程教育离领域 7 的“计算思维教育”和纵坐标最近，说明它不仅是计算机科学的基本内容，也是支持计算思维教育的关键教学内容。编程教育与信息技术教育教学距离较远，说明目前编程教育与信息技术学科（课程）的结合度还不够。领域 5 横跨第一和第三象限，算法与程序设计、教学实践与机器人内部联系强度还不够。如何合理地运用计算思维方式组织机器人模块开展教学，成为未来研究的重点。第三象限领域 7 的“人工智能、计算思维教育、计算机科学教育”这些关键词离纵坐标很近，在未来的发展中，这些教学内容的重要性日益凸显。

通过知识图谱可以发现我国目前中小学计算思维研究呈现以下现状特征：1.总体来讲，我国针对 K-12 阶段计算思维的培养处于起步探索阶段，尚未形成稳定的特征。第一象限的关键词是目前研究的重点与热点领域，处于二三象限的关键词与第四象限相比，在未来研究中更具有重要的发展潜力。2.小学离坐标的中心点最近，表征影响力最大，体现了小学是计算思维培养的重要阶段，这一特征跟国际上的趋势一致。一些发达国家（如美国、澳大利亚、英国、加拿大等）率先在 K-12 阶段培养计算思维，而我国也正在从高等教育阶段向中小学阶段过渡，开始重视 K-12 阶段的计算思维培养。目前我国在小学开展了形式多样的以编程为载体的机器人比赛和相应的课程学习计划。3.创客教育、STE（A）M、课程标准关键词间联系紧密，且离横坐标最近，表征这些课程内容对计算思维领域间影响的程度最大。中小学计算思维课程标准的建立将受到创客教育、STE（A）M 的影响，体现跨学科特性。4.教学模式离纵坐标最近，在领域内具有较高的联系强度。因此结合计算思维的特点与核心要素，在学科课程教学中构建教学模式是目前教学研究和实践的重点。5.此外，与评价相关的关键词还未出现在知识图谱的研究领域内，说明目前对计算思维的评价研究还较为缺乏。

4. 我国 K-12 计算思维研究的发展趋势研究

4.1. 构建跨学科的计算思维课程标准与框架

现阶段对计算思维培养的研究，总体停留在理论阶段，较为缺乏科学的、可操作性的标准体系。2018 年 1 月，我国高中信息技术课程标准将“计算思维”正式作为信息技术学科的核心素养，而在 k-12 的其他阶段尚未建立起对应的课程标准。从计算思维的特征来看，计算思维应用不同学科的共享元素来解决问题，促进对其他领域以及学科交叉问题的理解。学习科学、认知心理学认为构建相互关联的概念比单独的概念更利于学习者的意义建构与基于真实情境的问题解决，为未来建构跨学科的课程标准提供了理论依据。

总体来讲，我国的中小学教育目前处于分科式教学，在对跨学科融合计算思维的研究还处于初始阶段，需要对跨学科基础内容、核心概念、计算思维之间的融合方式进行探索。美国、英国、新加坡、加拿大等国重视建构大概念为核心的跨学科概念体系（李春密和赵芸赫，2017)，它是以分析学科共性概念与特征为基础，明确学科间的内在联系，提取并凝练如“抽象与具体、图式与模式”等跨学科的核心大概念来反映对客观世界的基本认知。在我国，祝智庭和雷云鹤（2018）认为 STEM 课程的四个主要学科本身具有天然的联系性，较易实现跨学科的整合，并提出整合机制。在课程标准的构建上，需要考虑 k-12 各学段的衔接性、知识的系统性和跨学科特性，做到统筹协调与循序渐进。

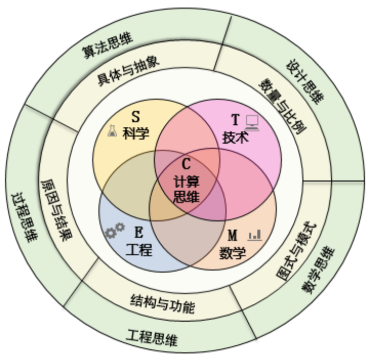


图 2 跨学科 STEM 融合计算思维培养的理论框架

本研究团队基于国内外文献分析，构建跨科学 STEM 融合计算思维培养的理论框架，分为三层，如上图 2 所示：1.内层为学科内容层，以计算思维为核心，融合 STEM 的学科知识，实现学科内容的跨越。2. 中间层为跨学科大概念层，以构建跨学科的核心概念（如抽象与具体、原因与结果等），实现学科核心概念的跨越。3. 外层为学习思维层，抽取计算思维通用算法，对计算思维过程进行分步设计，综合数学思维、算法思维等多种计算思维的方法，实现计算思维的跨越。

4.2. 构建计算思维教学模式

Hickmott，Prieto-Rodriguez 和 Holmes（2017）对 2006 至 2016 年间出版的计算思维文献的审查发现目前欠缺跨学科研究项目。目前国内以信息技术学科作为培养计算思维的主要课程载体，其教学模式一般以计算思维为内核、设计为主线、协作为过程的教学模式。江绍祥（Kong, 2016）提出了一个培养基础教育的课程设

计框架，并设计以兴趣为导向的教学策略。中小学的综合实践等社团活动，利用编程工具、机器人技术、游戏/模拟等形式开展教学。相应的编程活动、游戏、玩教具使低龄段学生更易理解计算思维，提高学习兴趣。学生使用图形对象构建程序，通过拖放式界面，在可视化编程环境中进行编程探究。STE（A）M 是培养计算思维的跨学科教学的重要的教学模式，在教学设计上大多以项目活动设计过程。综合跨学科的计算思维实践，根据学科的不同特点，关注计算思维的不同侧重点，运用计算思维的视角，探寻基于项目的问题解决方案。本研究团队对 K-12 基础教育的课程进行广泛调研后，提出基于单学科、社团和全校统整三种融入计算思维的教学模式。第一种基于数学、科学等基础学科内容，教学中分步融入计算思维的算法过程。第二种是在社团中基于计算机编程为内核制作产品原型，迭代完善产生创造性作品。第三种是由学校层面推动，在不同的年级确定适合的主题活动，利用计算思维算法制作基于物联网的产品原型，并进行展示与评价。

4.3. 开展基于评价的实证研究

由于计算思维是对学习者思维的逻辑性进行评价，评价具有复杂性，因此建立一套多维度、多工具的计算思维测评体系是研究的重点。国外主要通过认知、能力、情感等维度进行计算思维的评测。认知层面的主要方法有作品分析和评估思维过程，能力测试是计算思维的核心测评部分，主要包括计算思维总结性测试、形成性迭代测试以及应用技能测评，情感层面主要有自我效能感、编程态度等方面的测量。Brennan 和 Resnick（2012）提出了基于 Scratch 进行计算思维评价维度，受到了广泛的国际关注。Seiter（2013）提出理解和评估小学一至六年级计算思维的（PECT）框架。我国在这方面也作了一些积极的尝试，任友群、隋丰蔚和李锋（2016）总结计算思维教育在中小学信息技术课程中的能力评估方法，围绕计算思维开展实证研究（郁晓华、肖敏、王美玲等，2017；宁可为和杨晓霞，2018；王颖欣，2018）。但总体上相关的研究还比较少，需要开展更多的实证研究来验证理论、模式和应用效果的有效性，这将是我国计算思维未来研究重要的实践内容。

基金项目：国家自然科学基金项目"促进小学生计算思维培养的跨学科 STEM+C 教学理论与实证研究"（编号：71874066）；教育部人文社会科学研究青年基金项目（编号：14YJC880074）。华中科技大学附属小学合作项目"智慧教室中的教学促进学生深度学习"。

5. 参考文献

王佑镁和伍海燕（2012）。中国高教研究领域高频被引论文的学术特征分析——基于《中国高教研究》2000-2011 年刊载论文的计量分析。**中国高教研究，1**，33-37。

王颖欣（2018）。**基于计算思维三维框架的 Scratch 教学设计与实证研究（Doctoral dissertation）**。

宁可为和杨晓霞（2018）。基于 app inventor 的初中计算思维培养实证研究。**课程。教材。教法，38（2）**，112-117。

任友群、隋丰蔚和李锋（2016）。数字土著何以可能？——也谈计算思维进入中小学信息技术教育的必要性和可能性。**中国电化教育，1**，1-8。

李春密和赵芸赫（2017）。Stem 相关学科课程整合模式国际比较研究。**比较教育研究，05**，13-20。

李锋和王吉庆 (2013) 。计算思维:信息技术课程的一种内在价值。**中国电化教育，8**， 19-23。

郁晓华、肖敏、王美玲和陈妍（2017）。基于可视化编程的计算思维培养模式研究——兼论信息技术课堂中计算思维的培养。**远程教育杂志，6**，12-20。

郭守超、周睿、邓常梅、狄长艳和周庆国（2014）。基于 app inventor 和计算思维的信息技术课堂教学研究。**中国电化教育，3**，91-96。

祝智庭和雷云鹤（2018）。Stem 教育的国策分析与实践模式。**电化教育研究，1**，75-85。

赵蔚、李士平和姜强等（2017）。培养计算思维，发展 STEM 教育——2016 美国《K-12 计算机科学框架》解读及启示。**中国电化教育，5**，47-53。

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: American Educational Research Association.

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2017). A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms. *Digital Experiences in Mathematics Education*, *4*(1), 48-69.

Kong, S.C. (2016). A Framework of Curriculum Design for Computational Thinking Development in K-12 Education. *Journal of Computers in Education*, *3*(4), 377-394.

NMC (2017). *NMC Horizon Report K-12 Edition.* Retrieved December 17, 2018, from https://www.nmc.org/publication-type/horizon-report/.

Papert, S. (1996). An Exploration in the Space of Mathematics Educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Seiter, L., & Foreman, B. (2013). Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. *Proceedings of International ACM Conference on International Computing Education Research*. ACM, 59-66.

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Learning Effectiveness of Using Augmented Reality to Support Computational Thinking Learning Board Game

Wei-chen KUO[1], Ting-chia HSU[2*]
[12] Department of Technology Application and Human Resource Development,
National Taiwan Normal University, Taiwan
miranda831226@gmail.com, ckhsu@ntnu.edu.tw

ABSTRACT
This study utilized the computational thinking educational board game named “Robot City” as the instructional material. The board game corresponds to the concept of sequential, selection, repetition and subprogram of the programming language. This study integrated the augmented reality into helping the seventh grade students cultivate the competence of computational thinking from the board game. The study also explored the effects of student's learning achievement and cognitive load. The participants of this study were 50 students from two classes. They were divided into an experimental group (n = 27) and a control group (n = 23). The students in the experimental group used augmented-reality system to support their learning in the computational thinking board game. The students in the control group used the augmented reality learning system integrated with traditional teacher-centered multimedia instruction to support their learning in the computational thinking board game. From the experiment results, the learning effectiveness of the experimental group was significantly higher than those of the control group. The cognitive load of the students in the experimental group was significantly lower than those of the control group.

KEYWORDS
computational thinking, game-based learning, board game, augmented reality, cognitive load

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

擴增實境運算思維教育桌遊之學習成效與認知負荷之分析

郭韋辰[1]，許庭嘉[2*]
[12] 國立臺灣師範大學科技應用與人力資源發展學系，臺灣
miranda831226@gmail.com，ckhsu@ntnu.edu.tw

摘要

本研究提出以機器人蓋城市教育桌遊為實驗教材，學習結構化程式設計，分別對應到程式語言的循序、選擇、重覆、副程式之概念。本研究實驗組透過擴增實境輔助同學進行運算思維教育桌遊，讓學生從遊戲中自主學習運算思維，幫助國中七年級學生培養與內化運算思維之概念，並探討學生學習成就與認知負荷之影響。本研究控制組以相同實驗時間但是在學生學習過程介入教師為中心的傳統多媒體引導教學，讓學生透過運算思維教育桌遊來培養運算思維素養。研究結果發現，實驗組的學習成就顯著高於控制組，而且實驗組的認知負荷顯著低於控制組。

關鍵字

運算思維；遊戲式學習；教育桌遊；擴增實境 ；認知負荷

1. 前言

運算思維是一種用電腦的邏輯來解決問題的思維，然而資訊不斷進步，運算思維的能力是各國皆相當重視的議題，運算思維的能力也被認為是二十一世紀的基本能力之一（Barr, Harrison & Conery, 2011）。而雖然程式設計，不是培養運算思維的唯一途徑，卻是最快速且符合當下趨勢的選擇（Grover & Pea, 2013），不論學生未來是否會成為工程師，擁有運算思維之能力，對學業及生活，絕對是有利的。因此培養運算思維的能力成為現今熱門的研究議題。而心理學家也發現，透過玩遊戲，學生可以學習如何有效的解決問題，培養運算思維。本研究採用教育桌遊-機器人蓋城市作為學習教材，其中有學者發現單純使用桌上遊戲，因提供資訊有限（魏珮君，2017），不足以提高學生的學習動機。然而科技產品推陳出新，擴增實境已被廣泛運用於教育產業，讓學生將抽象之事物可視化，並透過互動給學生最即時的回饋，以提高學生的學習成效。部分研究顯示擴增實境對學生學習是有幫助的（Cascales-Martínez, Martínez-Segura, Pérez-López & Contero, 2017; Huang, Chen & Chou, 2016），但也有研究指出擴增實境為新科技，對學生來說造成學生認知負荷過大（Akçayır & Akçayır, 2017），介面對學生來說不直觀，學生不會使用等問題（Frank & Kapila, 2017）。

因此，本研究將擴增實境結合傳統教師授課，以不增加學生負擔為原則，使學生有更多時間自主探究，輔助學生學習，應用於國中七年級的資訊教育課程中，以探討在擴增實境環境下搭配傳統教師授課的學習模式，與擴增實境搭配教師多媒體輔助之學習模式，學生在學習成就、學習認知負荷之影響。期望實驗組擴增實境與傳統教師授課結合，能使學生自主探究學習更清楚概念增加學習成效且降低學生認知負荷。

2. 文獻探討

2.1. 運算思維

運算思維簡稱（CT），近期，受到大家廣為重視，有些國家甚至已將運算思維納入十二年國民教育中（Grover & Pea, 2013）。此一詞是由學者 Wing 於 2006 年提出，並將其定義為問題解決，更有系統與效率的瞭解人類的行為，且認為運算思維是每個人應具備的能力（Wing, 2006）。更有學者認為運算思維之概念應該融入其他科目以豐富學習。雖然程式語言並不等同於運算思維，但是學習程式語言如 Java、Python、C++等，和培養運算思維的方式相似，因此，被研究者認為是培養運算思維的主要管道（Smith, Cypher & Tesler, 2000）。更有學者提到若先進行培養運算思維能力的相關課程，對於學生在一般程式語言的學習上有更好的幫助（Davies, 2008）。

2.2. 遊戲式學習

遊戲式學習簡稱（GBL），近年來，成為高度討論的學習方式，遊戲式學習之方式已成為趨勢（Kim, Park, & Baek, 2009），學生在玩遊戲的過程中，增加沉浸感提高學生的興趣與學習成就（Prensky, 2001）。傳統一般教室授課方式會阻礙學生的創造力，只強調有一個正確答案，因此遊戲式學習被許多學者提出認為是一種有效的教學方法（Connolly, Boyle, MacArthur, Hainey & Boyle, 2012）。而 GBL 的方式有很多種，本研究以教育桌遊機器人蓋城市為遊戲式學習的教材。而將桌遊應用於教育中已經有數十年的歷史（Ramani & Siegler, 2008），許多研究也指出，透過桌遊，應用於教學環境中，對於學生的學習成就有顯著的成效（Hinebaugh, 2009）。如有學者將桌遊應用於物理和天文學來幫助學生學習（Cardinot & Fairfield, 2019），在美國有學者甚至指出，約過半數的企業願意將桌遊融入於新人教育訓練中（Faria & Nulsen, 1996）。

2.3. 桌上遊戲

桌上遊戲，近年來已廣泛運用於教育中，也融入眾多學科中，且部分研究顯示透過桌遊學習對學習動機是有顯著提升，在美國，有學者透過桌上遊戲來輔助學生學習數學，研究結果顯示對學生的數學學習成就有顯著提高（Ramani & Siegler, 2008），而在馬來西亞也有學者將桌上遊戲納入教育學科課程中，參與者也對桌遊學習做出正面的回應

（Sasidharan & Tan, 2017）。以上皆證明透過桌上遊戲可以幫助學生學習，提高學習成就。

2.4. 擴增實境

隨著科技產品不斷地發展與普及，現今，人們習慣在科技產品上查詢資料或閱讀資訊。但僅在行動裝置上進行閱讀，無法即時的給學生回饋。也沒辦法讓學生有身歷其境之感。然而擴增實境技術，可以解決此問題，能在適當的時機給使用者適當的資訊，可以讓使用者與虛擬物件即時互動進而加深使用者的印象（Ibáñez, Di Serio, Villarán, & Kloos, 2014）。因此，Cheng & Tsai（2012）指出將擴增實境應用於教育方面是有潛力的。且能有效的解釋學習內容之概念（Ibáñez & Delgado-Kloos, 2018）。而 Di Serio、Ibáñez 與 Kloos（2013）之研究也證明，比起使用傳統教學方式，在教學中使用擴增實境的更能夠吸引學生學習以及提高學生的學習成就。而擴增實境也廣泛的應用於各個學科中，有學者將擴增實境運用於國語寫作教學中，透過擴增實境，輔助學生寫作，有別於過去寫作僅能看到平淡無奇的文字敘述情境，擴增實境能將寫作情境活靈活現以虛擬畫面呈現，有效幫助學，豐富學生的想法，並寫成文章（Wang, 2017），擴增實境也可能增加其他教育發展像是數學能力或學習外語（Hsu, 2017）。

3. 擴增實境回饋機制學習系統

3.1. 系統架構

本系統是使用 Unity 開發，系統由老師先將編輯擴增實境教材，而系統中的資料庫包含，學生的學習歷程資料庫、個人資料、擴增實境教材資料庫、範例卡牌掃描資料庫，其中擴增實境教材資料庫包含影片教材資料庫、測驗題教材資料庫、程式語言教材資料庫，最後學生操作擴增實境系統，而老師可以透過後台資料庫模組，查看學生的學習狀況及學生的基本資料。

3.2. 應用程式安裝

學生須有機器人蓋城市-Robot City 教育桌上遊戲及行動裝置，如智慧型手機或平板搭配使用。此應用程式有上架於 google play 商店，至 play 商店搜尋欄位輸入「機器人蓋城市」點選搜尋，即可找到該應用程式，點選下再安裝，可安裝完成。

3.3. 系統介面與功能介紹

首先，先介紹機器人蓋城市教育桌遊，主要為卡牌遊戲，其玩法為蒐集任務卡牌上的元素，如學生抽到寺廟，則蒐集石頭與木頭兩種元素即完成任務。

而一人手中僅有八張移動卡牌，卡牌種類分別對應程式語言的循序、選擇、重覆、副程式之概念，學生須進行思考，如何以最短路徑，得到所需元素完成任務，培養學生問題解決能力。學生會依照範例牌組，排出欲掃描的卡牌，接著進行掃描。掃描成功後，系統會依照辨識出的卡牌給予學生不同的學習內容，包含教學影片、程式碼及測驗題。一開始老師先引導學生進行教學影片觀看，目的是希望學生透過教學影片培養運算思維之概念，接著請學生，了解相關內容之程式碼，程式碼部分，分成四種不同的程式語言，包含 Scratch、VB、Python、C，學生可依照自己目前正在學習的程式語言進行學習，也可以比較不同程式語法之間的差異。最後老師會引導學生進行測驗題，確保是否真的理解其內容，測驗題部分分成選擇題及配合題，配合題主要是以卡牌拖拉方式答題，學生要真的理解其內容將卡牌拖拉到正確位置才算答題成功。

4. 研究方法

4.1. 研究對象

研究對象為台灣北部國中七年級的學生，年齡約 12 至 13 歲，兩個班級，共 50 名學生參與本次實驗活動，採隨機分組，實驗組為 27 名學生以擴增實境輔助運算思維教育桌遊自主學習模式，控制組為 23 名學生以擴增實境結合教師主導多媒體教學模式。

4.2. 研究架構

本研究的自變項為多媒體教學模式，我們使用自行開發的機器人蓋城市 Robot City 擴增實境應用程式，而多媒體教學則是使用投影片教學方式輔助學生學習，控制組學生利用遊戲式學習結合擴增實境及教師多媒體教學引導操作模式進行學習; 實驗組學生利用遊戲式學習結合擴增實境引導操作模式進行學習。本研究之依變項為運算思維學習成就後測驗卷及學習認知負荷。藉由不同的學習模式，探討其在運算思維學習成效上的影響。本研究欲減少不相關之變項造成的影響，提高內在效度。因此，在教學實驗活動上，所有學生皆由同一位授課超過五年的資訊專科老師進行教學，以避免因不同教師教學風格不同而影響實驗之結果如圖 1 所示。除此之外，所有教學活動的內容皆為運算思維的基本概念，所有學生先前都上過相同的先備課程。

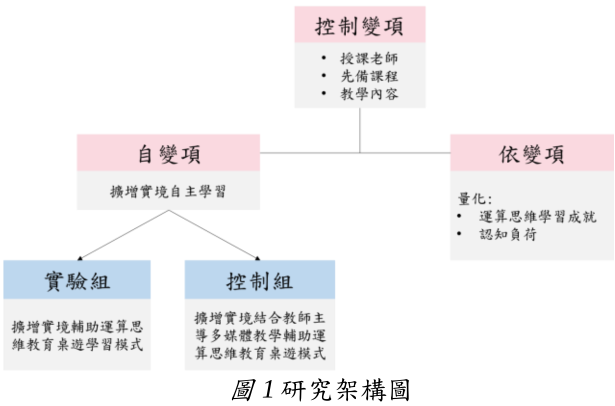


圖 1 研究架構圖

4.3. 實驗流程

實驗流程如圖 2 所示。在學習活動前，所有學生須進行運算思維測驗前測驗卷填寫，再由同一位資深資訊老師說明操作及介紹，實驗組及控制組依照不同學習模式進行 120 分鐘操作本系統學習。學習活動結束後，學

生再進行六十分鐘的運算思維後測驗卷，了解學生之學習成就是否提高同時，學生也會在活動後填寫認知負荷問卷調查。為期 300 分鐘。

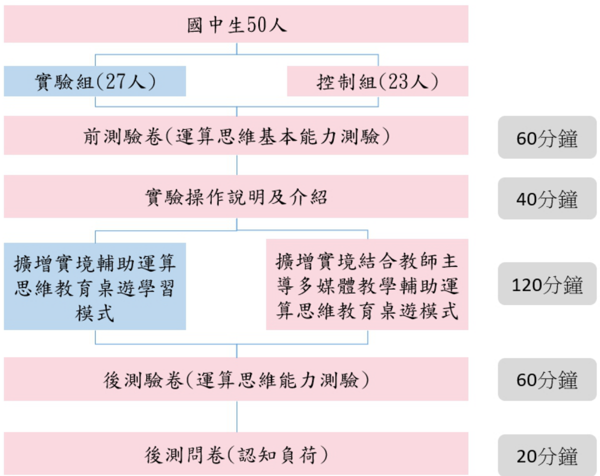


圖 2 實驗流程

為配合國中資訊教育內容，採用研究工具為運算思維學習成就測驗、學習認知負荷問卷量表。本研究學習成就分為前測驗及後測驗，題目為 Bebras 國際運算思維題目及流程圖，8 題流程圖其中包含一題簡答題及 10 題運算思維題目，共 17 題單選題一題簡答題。本研究認知負荷問卷總共有 8 題，採用李克特 5 點量表，其中心智負荷（內在負荷）共 5 題，心智努力（外在負荷）共有 3 題，Cronbach's alpha 為 0.97（Hwang & Wang, 2013）。本量表採用李克特五點量表，1 分為「非常不同意」，5 分為「非常同意」。

5. 研究結果與分析

本研究透過共變數分析（ANCOVA）來比較實驗組與控制組的學生排除前測成績的差異後，其學習成就，是否達到顯著差異。本研究針對實驗組與控制組的學生在學習活動後的認知負荷進行獨立樣本 t 檢定，以比較學生在學習活動後的認知負荷的差異是否有達到顯著差異。

5.1. 運算思維學習成就測驗

為瞭解學生之運算思維學習成就是否產生差異，將運算思維學習成就前、後測問卷採用單因子共變數分析（ANCOVA），探討擴增實境輔助運算思維教育桌遊自主學習的學習成就之影響。在進行運算思維學習成就之單因子共變數分析之前，先進行組內迴歸係數同質性考驗，F 值=0.999；p=0.323>0.05，未達顯著水準，可繼續進行共變數分析。

在排除運算思維學習成就前測分數對於運算思維學習成就後測成績的影響後，學生之學習成就的共變數分析摘要如表 1 所示。由分析結果得知，實驗組的學生平均分數為 95.40 分，調整後平均為 95.54 分，控制組之平均分數為 77.91 分，調整後平均為 77.76 分。並將學習活動運算思維學習成就前測驗成績的影響力排除之後，組別所造成的變異數達顯著水準（F=4.167，p<0.05），顯示運算思維學習測驗成績會因為不同學習模式而有顯著差異。研究結果表示，實驗組的學生，使用一般傳統授課之學生，運算思維學習能力顯著提升。

表 1 學習成就之 ANCOVA 分析摘要表

組別	人數	平均值	標準差	調整後平均數	標準誤	*F*
實驗組	27	95.40	39.50	95.54	5.91	4.167*
控制組	23	77.91	31.59	77.76	6.40	

*p<.05

5.2. 運算思維學習認知負荷

為瞭解擴增實境輔助運算思維教育桌遊自主學習的認知負荷，本研究採獨立樣本 t 檢定，以比較學生在學習活動後的認知負荷的差異是否有達到顯著差異。

計算後實驗組的認知負荷為 2.77，控制組為 3.25，t 值為 2.11，雙尾顯著性 p 值=0.04<0.05，拒絕虛無假設。證明使用擴增實境輔助運算思維教育桌遊學習，對學生的認知負荷有顯著影響。實驗組的學生其認知負荷顯著較控制組低。

表 2 認知負荷之獨立樣本 T 檢定分析摘要表

組別	人數	平均值	標準差	*t*
實驗組	27	3.25	0.80	*2.11**
控制組	23	2.77	0.81	

*p<.05

6. 結論與建議

本研究使用擴增實境輔助運算思維教育桌遊，以學生最熟悉之方式自主學習，以幫助學生提高學習成就及降低學生認知負荷。由研究結果得知，實驗組之學生使用擴增實境輔助運算思維教育桌遊進行自主學習，與透過擴增實境和教師之多媒體教學輔助運算思維教育桌遊的學習成效相比，不僅有效增加學生的學習成就，同時也降低學生的認知負荷。本研究推測擴增實境對學生來說，使用擴增實境輔助運算思維教育桌遊的組別，給學生自主學習和內化自主探究的學習內容時間較足夠，而且沒有其他多媒體干擾，故學習成效較佳。反之，控制組以教師為中心的多媒體學習模式進行教學，即便一樣有提供學生擴增實境輔助運算思維教育桌遊，然而過多分階段的多媒體反而會造成學生注意力分散及心流中斷，並且犧牲學生使用擴增實境輔助運算思維教育桌遊自主學習和主動內化的時間，造成學生的負擔，進而提高學生之認知負荷。此結果和 Hwang、Chen & Chou（2016）的研究結果相同，學者們認為適當的使用擴增實境能輔助學生學習，但若

再加入其他學習策略會導致學生負擔過大，無法承受等問題。因此本研究認為，若使用擴增實境輔助學生學習，應減少學生其他外在負擔，避免心流中斷，讓學生可以全然沉浸在遊戲過程，如同本研究使用擴增實境輔助運算思維教育桌遊。然而，在本研究中仍有部分研究限制及研究建議，已提供後續的研究學者可以在設計擴增實境實驗時得以完善。本研究的實驗時間較短，僅透過 120 分鐘的遊戲式結合擴增實境進行學習，考慮學生在遊戲式學習的沉浸感，因此建議未來學者可以將時間拉長，增加學生對遊戲的沉浸感及熟悉度。在使用教學教材上，本研究採用桌上遊戲，研究者也可以選用不同種教育桌遊或數位遊戲式學習來培養學生運算思維之概念。此外，未來也建議學者可以嘗試將不同的學科或不同的教學策略導入於擴增實境中，幫助學生學習。

7. 致謝

本研究感謝科技部研究計畫編號: MOST 105-2628-S-003-002-MY3 與 MOST 107-2511-H-003-031 補助。

8. 參考文獻

魏珮君（2017）。桌上遊戲融入差異化教學對國小英語學習成就與動機研究。**淡江大學教育科技學系數位學習碩士在職專班學位論文**，1-115。

Akçayır, M., & Akçayır, G. (2017). Advantages and Challenges Associated with Augmented Reality for Education: A Systematic Review of the Literature. *Educational Research Review, 20*, 1-11.

Barr, D., Harrison, J., & Conery, L. (2011). Computational Thinking: A Digital Age Skill for Everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Cardinot, A., & Fairfield, J. A. (2019). Game-based Learning to Engage Students with Physics and Astronomy Using a Board Game. *International Journal of Game-Based Learning (IJGBL), 9*(1), 42-57.

Cascales-Martínez, A., Martínez-Segura, M.J., Pérez-López, D., & Contero, M. (2017). Using an Augmented Reality Enhanced Tabletop System to Promote Learning of Mathematics: A Case Study with Students with Special Educational Needs. *EURASIA J. Math., Sci Tech, 13*(2), 355-380.

Cheng, K. H., & Tsai, C.C. (2013). Affordances of Augmented Reality in Science Learning: Suggestions for Future Research. *Journal of Science Education and Technology, 22*(4), 449-462.

Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T., & Boyle, J. M. (2012). A Systematic Literature Review of Empirical Evidence on Computer Games and Serious Games. *Computers & Education, 59*(2), 661-686.

Davies, S. (2008). The Effects of Emphasizing Computational Thinking in an Introductory Programming Course. In *2008 38th Annual Frontiers in Education Conference*. IEEE, T2C-3

Faria, A. J., & Nulsen, R. O. (1996). Business Simulation Games: Current Usage Levels. A Ten Year Update. Paper Presented at the Developments in Business Simulation and Experiential Learning: *Proceedings of the Annual ABSEL conference.*

Frank, J. A., & Kapila, V. (2017). Mixed-reality Learning Environments: Integrating Mobile Interfaces with Laboratory Test-beds. *Computers & Education, 110*, 88-104.

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

Hinebaugh, J. P. (2009). *A Board Game Education*: R&L Education.

Hsu, T. C. (2017). Learning English with Augmented Reality: Do learning Styles Matter?. *Computers & Education, 106*, 137-149.

Huang, T.C., Chen, C.C., & Chou, Y.W. (2016). Animating Eco-education: To See, Feel, and Discover in an Augmented Reality-based Experiential Learning Environment. *Computers & Education, 96*, 72-82.

Hwang, G. J., Yang, L. H., & Wang, S. Y. (2013). A Concept Map-embedded Educational Computer Game for Improving Students' Learning Performance in Natural Science Courses. *Computers & Education*, *69*, 121-130.

Ibáñez, M. B., Di Serio, Á., Villarán, D., & Kloos, C. D. (2014). Experimenting with Electromagnetism Using Augmented Reality: Impact on Flow Student Experience and Educational Effectiveness. *Computers & Education, 71*, 1-13.

Ibáñez, M.-B., & Delgado-Kloos, C. (2018). Augmented Reality for STEM Learning: A Systematic Review. *Computers & Education*.

Kim, B., Park, H., & Baek, Y. (2009). Not Just Fun, but Serious Strategies: Using Meta-cognitive Strategies in Game-based Learning. *Computers & Education, 52*(4), 800-810.

Prensky, M. (2001). Fun, Play and Games: What Makes Games Engaging. *Digital game-based learning, 5*(1), 5-31.

Ramani, G. B., & Siegler, R. S. (2008). Promoting Broad and Stable Improvements in Low-income Children's Numerical Knowledge through Playing Number Board Games. *Child development, 79*(2), 375-394.

Sasidharan, A., & Tan, K. E. (2017). Pupils'and Teachers' Perceptions of a Language Board Game, Challenge. *The English Teacher*, 3, 20.

Smith, D. C., Cypher, A., & Tesler, L. (2000). Programming by Example: Novice Programming Comes of Age. *Communications of the ACM*, *43*(3), 75-81.

Wang, Y. H. (2017). Exploring the Effectiveness of Integrating Augmented Reality-based Materials to Support Writing Activities. *Computers & Education, 113*, 162-176.

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

Kong, S.C., Andone, D., Biswas, G., Hoppe, H.U., Hsu, T.C., Huang, R.H., Kuo, B.C., Li, K.Y., Looi, C.K., Milrad, M., Sheldon, J., Shih, J.L., Sin, K.F., Song, K.S., & Vahrenhold, J. (Eds.). (2019). *Proceedings of International Conference on Computational Thinking Education 2019.* Hong Kong: The Education University of Hong Kong.

Exploring Convergence and Divergence in Infinite Series

David ZEIGLER
California State University, Sacramento, The United States
zeigler@csus.edu

ABSTRACT

This brief article outlines a series of proposed activities designed to encourage students to solidify the concepts of convergence and divergence of series using computational exploration. The exercises can be embedded into the coursework of a typical Calculus class as a homework assignment or group activity. Attention is focused on the harmonic series. After a heuristic experiment to establish the divergence of the harmonic series, students are led through non-trivial activities to tweak to make a divergent series converge. The list of activities is not intended to be complete but to offer the reader additional computational thinking-based tools to highlight concepts that usually cause students difficulty.

KEYWORDS

harmonic series, conditional convergence

1. INTRODUCTION

The concept of Computational Thinking (CT) was first presented by Alan Perlis (Perlis, 1962) who stated that everyone should learn to program as part of liberal education. Recognizing the emergence of computers and computing in education, he argued that programming is an exploratory process. Exercises such as calculating square roots by Newton-Raphson or Gaussian elimination or interpolation are not appropriate vehicles for revealing the real issues in programming. Rather, provide problems that were of a very simple kind that did not require students to have a deep understanding of mathematics to understand the intent of the program or the problem and the goal they were to reach. Problems were very simple and students understood exactly what they were supposed to do, and yet nowhere in their education had they ever been supplied with the techniques to complete the tasks.

The idea gained traction in 2006 when Jeanette Wing (Wing, 2006) presented CT as a new approach to problem solving and designing systems. As a new form of analytical thinking, CT shares with mathematics the general way to approach problem solving by representing data through abstractions. These abstractions necessarily introduces layers of abstraction and the relationship between the layers must be kept in mind. The basic mechanics of CT is to define the abstractions, work with the layers of abstraction, and understand the relations between them. Crucial to this process is the ability to thinking algorithmically and understanding the consequences of scale. In addition to abstraction, recurring themes are efficiency and heuristics.

Since the publication of Wing's essay, there has been a steady increase in the popularity of the concept of computational thinking. The National Science Foundation's Computing Education for the 21st Century (CE21) and CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) funding competitions encourage projects that develop computational thinking competencies. The College Board has adopted a new Advanced Placement course in Computer Science that built around a set of CT Practices. CT was used as a common thread to link the three levels of K-12 learning standards developed by the Computer Science Teachers Association. Arguably, a good indicator of the popularity of CT is the recognition from technology companies that produce consumer devices. Google for Education has a program, Exploring Computational Thinking, with resources for educators and administrators to integrate CT into their classrooms. Microsoft Research sponsors the Center for Computational Thinking at Carnegie Mellon University.

However, the adoption of CT in higher education has been more scattered (Czerkawski, 2015). This is surprising considering the strong demand for developing computer science undergraduates. The ubiquity of calculus in STEM education makes it an ideal candidate for enhancement using CT methods. The focus of this document is to propose a sequence of open-ended activities to highlight the concept of convergence of an infinite series. Success with series relies upon a student's ability to recognize convergence (and divergence) patterns. Using the computer to perform the raw calculations frees the student to conduct mathematical investigations, first describing the rules they discover and then expressing these as mathematically. The activities are based on CT principles and are designed to be easily included in a traditional Calculus course without significant disruption. The foundational idea is to provide a framework for students to explore Calculus through numerical experiments.

2. BACKGROUND

Arguably, one of the more challenging topics for students to master in a typical Calculus series at the university level is the infinite series. Most students have been exposed to the idea of a convergent series with the decimal representation of an irrational number.

For some, the nature of an infinite process is such that it may not be completed in a finite amount of time and so summing a series is bound to remain unclear (Martínez, 2012). Series problems are well-suited for computational thinking because of the simplicity of the computational aspect of the problems. Hidden in the mechanics of series problems are deeper questions. What exactly is meant when we say that a series converges or diverges? Is there a threshold for a series to be convergent?

The following details a sequence of activities that take advantage of computing to solidify the concepts of convergence and a convergent series. Students are encouraged to explore and try to come up with their own rules for predicting convergence (or divergence).

3. ACTIVITIES

The starting point will the harmonic series

$$\sum_{n=1}^{\infty}\frac{1}{n}=1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots$$

A traditional Calculus class shows that the harmonic series diverges. The growth of the harmonic series is modest but it can be measured. Ask students to write a simple snippet of code that can generate the n^{th} partial sum of a series.

3.1. Activity #1

An old joke says that the harmonic series is known to diverge but it has never been observed to do so. Suppose we began at the Big Bang, approximately 13.8 billion years ago, and added one term of the harmonic series per second.

1. What would be the partial sum today?

   As it turns out, the sum is in the low 40's which may be disheartening for students expecting a grand number.

2. How many steps are required for the partial sum to reach 10? 100? 1000?

   Students should be encouraged to plot the points (n,S(n)), where S(n) is the n^{th} partial sum, to see if the resulting curve is familiar. In particular,

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}\approx\ln n$$

3.2. Activity #2

Similar to appearance to the harmonic series is the p-series

$$\sum_{n=1}^{\infty}\frac{1}{n^p}$$

that converges if $p>1$ and diverges if $p\leq 1$. This suggests that the series converges if the terms in the series are smaller than the corresponding terms of the harmonic series. Conversely, the series will diverge if the terms are larger than the corresponding terms in the harmonic series. To bring this intuitive argument to life, students should compare the harmonic series with the two series

$$\sum_{n=1}^{\infty}\frac{1}{n^{1+1/n}}$$

and

$$\sum_{n=1}^{\infty}\frac{1}{n^{1-1/n}}$$

Students should be encouraged to plot the partial sums of the series along with the harmonic series. Based only on the plots, can they determine which series converges and which diverges?

3.3. Activity #3

From the previous activities, students should have a sense that the harmonic series is “just divergent”. Can a divergent series be made convergent? This question will be investigated using the Kempner Series, a modification of the harmonic series, formed by omitting all terms whose denominator expressed in base 10 contains the digit 9.

$$1+\frac{1}{2}+\cdots+\frac{1}{8}+\frac{1}{10}+\frac{1}{11}+\cdots+\frac{1}{18}+\frac{1}{20}+\cdots$$

The series converges (Ballie, 1979) but finding the exact sum of a Kempner is an intractable problem. Will the series converge if a different digit is removed? Removing digits in the denominator of any digit will result in convergence but, as one can guess, the sum changes. Is there a relationship between the digit removed and the resulting sum?

4. CONCLUSION

The proposed activities described in this paper have focused on helping students build a framework for understanding the dual concepts of convergence and divergence of an infinite series. The focus has been on developing an intuitive sense of properties of a convergent series. Within this framework, students are encouraged to explore non-traditional series. In particular, to try to describe and categorize series behaviors based upon their construction.

5. REFERENCES

Baillie, R. (1979). Sums of Reciprocals of Integers Missing a Given Digit. *The American Mathematical Monthly, 86*(5), 372-374.

Czerkawski, B., & Lyman, E. (2015). Exploring Issues about Computational Thinking in Higher Education. *TechTrends, 59*(2), 57-65.

Martínez-Planell, R., Gonzalez, A., DiCristina, G., & Acevedo, V. (2012). Students‘ Conception of Infinite Series. *Educational Studies in Mathematics, 81*(2), 235-249.

Perlis, A. (1962). The Computer in the University. In M. Greenberger (Ed.). *Computers and the World of the Future*, 180-219. Cambridge, MA: MIT Press.

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

CTE 2019

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

CTE 2019

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

URL:
www.eduhk.hk/cte2019

Email:
cte2019@eduhk.hk

Created and Funded by

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

Co-created by

香港教育大學
The Education University of Hong Kong

Massachusetts Institute of Technology

CityU
香港城市大學
City University of Hong Kong