The 8th APSCE International Conference on Computational Thinking and STEM Education

第八届APSCE计算思维与STEM教育国际会议

CTE-STEM 2024

Beijing Normal University, China

中国·北京师范大学

2024.5.28-30

Conference Proceedings

CTE-STEM 2024

The 8th APSCE International Conference on Computational Thinking and STEM Education

第八届APSCE计算思维与STEM教育国际会议

Conference Proceedings of International Conference on
Computational Thinking and STEM Education 2024
(CTE-STEM 2024)

2024.5.28-30

Beijing Normal University, China

中国·北京师范大学

Organized by
Asia-Pacific Society Computers in Education(APSCE)

Publication of Asia Pacific Society for Computers in Education
ISSN 2664-5661

About the Beijing Normal University (BNU), Beijing, China

Beijing Normal University (BNU) grew out of the Education Department of Imperial University of Peking established in 1902, which initiated teacher training in China's higher education. After the development for over a century, BNU has become a comprehensive and research-intensive university with its main characteristics of basic disciplines in sciences and humanities, teacher education and educational science.

BNU consists of Beijing Campus and Zhuhai Campus. The University has 3 faculties, 27 schools, 2 departments, 11 research institutes and 4 academies. In addition, there are more than 5.4 million books and 8.2 million e-books in its libraries. BNU is home to more than 35,000 full-time students, and has more than 8000 faculty members, including 2562 full-time teachers, 94% of whom have earned a doctoral degree.

At present, the university has established cooperative ties with about 300 universities and international organizations from more than 40 countries and regions. Each year, above 900 international professors and scholars are invited to lecture and research at the University. And BNU has around 2000 long-term international students, the scale of which ranks among top in China's universities.

The motto of Beijing Normal University is "Learn, so as to instruct others. Act, to serve as example to all."

For more information, please visit: https://english.bnu.edu.cn/

Asia-Pacific Society for Computers in Education

The **Asia-Pacific Society for Computers in Education (APSCE)** was formed on 1 January 2004. It is an independent academic society whose broad objective is to promote the conduct and communication of scientific research related to all aspects of the use of computers in education, especially within the Asia-Pacific.

The specific objectives of APSCE are:

- To promote the conduct and dissemination of research employing the use of computing technologies in education within the Asia-Pacific region and internationally.
- To encourage and support the academic activities of researchers in member countries and to nurture a vibrant research community of younger as well as more experienced researchers.
- To enhance international awareness of research conducted by researchers in member countries.
- To obtain greater representation of active researchers from the Asia-Pacific region in committees of related leading academic and professional organizations and the editorial boards of reputable journals.
- To organize and hold the International Conference on Computers in Education (ICCE) conference series in member countries.
- To engage in other appropriate academic and professional activities including but not limited to the setting up of Special Interest Groups (SIGs) and the publication of a Society newsletter and a Society journal.

For more information, please visit: https://apsce.net/

Perface

The APSCE International Conference on Computational Thinking (CT) and STEM Education (CTE-STEM) is a global academic conference that focuses on the field of computational thinking and STEM education. It serves as a platform for researchers, educators, policy makers, and industry professionals to exchange and share the latest research findings, experiences, and perspectives. The conference covers various aspects of computational thinking and STEM education, including teaching methods, curriculum design, educational technology, assessment and evaluation, and more. Participants have the opportunity to engage in in-depth academic exchanges and collaborations with experts and scholars from different countries and regions through presentations, research reports, workshops, seminars, and other formats.

With globalization, technology, and informatization, CT and STEM education have new perspectives and trends in the context of multilateral international relations and complex social environments. The 8th APSCE International Conference on Computational Thinking and STEM Education 2024 (CTE-STEM 2024) is organized by the Asia-Pacific Society for Computers in Education (APSCE). CTE-STEM 2024 is hosted by the Beijing Normal University, China (BNU). CTE-STEM 2024 will focus on these themes, sharing experiences, discussing differences, reaching consensus, and promoting the development of CT education. The conference will include keynote speeches, panel discussions, a teachers' forum, and paper presentations.

On behalf of APSCE, BNU and the Conference Organizing Committee, we would like to thank all the invited panelists, the keynote speakers, as well as paper presenters for their contribution to the success of CTE-STEM 2024.

We sincerely hope all of you will enjoy and be inspired from participating in and attending CTE-STEM 2024.

Conference Theme and Conference format

Theme:
Computational Thinking and Computing-related STEM Education

Sub-themes:

- Computational Thinking and Unplugged Activities in K-12
- Computational Thinking and Coding Education in K-12
- Computational Thinking and Subject Learning and Teaching in K-12
- Computational Thinking and Teacher Development
- Computational Thinking and IoT
- Computational Thinking Development in Higher Education
- Computational Thinking and STEM/STEAM Education
- Computational Thinking and Non-formal Learning
- Computational Thinking and Psychological Studies
- Computational Thinking and Special Education Needs
- Computational Thinking in Educational Policy
- General Submission to Computational Thinking Education
- Computational Thinking and Evaluation
- Computational Thinking and Data Science
- Computational Thinking and Artificial Intelligence Education
- Computational Thinking and its Key Elements
- Computational Thinking as Method
- STEM and Interdisciplinary Integration
- Open-Source Software and Hardware for CT and STEM Education

Conference format

CTE-STEM 2024 plans to be run by offline. Accepted English papers will be published in Scopus-indexed conference proceedings. Accepted Chinese papers will be indexed by CNKI.

Conference Organization

General Conference Chairs

Siu Cheung Kong, The Education University of Hong Kong, Hong Kong

Lei Fan, Capital Normal University, China

Program Committee Chairs

Jinbao Zhang, Beijing Normal University, China

Valentina Dagienė, Vilnius University, Lithuania

Sun Daner, The Education University of Hong Kong, Hong Kong

Local Organizing Committee Co-Chair

Tingwen Chang, Beijing Normal University

Guang Chen, Beijing Normal University

Yu Lu, Beijing Normal University

Feng Li, East China Normal University

Nan Wang, Beijing University of Posts and Telecommunications

Teacher Forum Committee Co-Chair

Qian Fu, Beijing Normal University

Xiaoya Yu, Beijing Institute of Education

Fanny Kong, The Education University of Hong Kong

Program Committee Members

Diana ANDONE	Politehnica University of Timisoara
Heinz Ulrich HOPPE	University of Duisburg-Essen
Hiroaki OGATA	Kyoto University
Jean HAUCK	Federal University of Santa Catarina
Jon MASON	Charles Darwin University
Ju-Ling Shih	National Centre University
Ki-sang SONG	National University of Education
Kuo Bor-chen	National Taichung University of Education
Looi Chee-kit	The Education University of Hong Kong

Marcus Specht	Technical University of Delft
Marcelo MILRAD	Linnaeus University
Mi-song KIM	University of Western Ontario
Ronghuai HUANG	Beijing Normal University
Shitanshiu MISHRA	Vanderbilt University
Tak-Wai CHAN	National Central University
Ting-chia Hsu	National Taiwan Normal University
Yanyan Li	Beijing Normal University
Zehra Altınay	Near East University
Charalampos Karagiannidis	University of Thessaly
Mohamed Shahat	Sultan Qaboos University
Maria Giulia Ballatore	Polytechnic University of Turin
FAHRİYE ALTINAY	Near East University, Northern part of Cyprus
Wei Ni	Chongqing University of Technology
Zuheir N. Khlaif	An Najah National University, Palestine
Qi Bian	Inner Mongolia Normal University
Haipeng Wan	Capital Normal University
Zhihe Li	Shanxi Normal University
Youmei Feng	Tianjin Normal University
Peng Chen	Capital Normal University
Baoping Li	Beijing Normal University
Samia Kouki	Higher colleges of Technology, UAE
Külli Kori	Tallinn University
Mouna DENDEN	Polytechnic University Hauts-de-France
Robert John D. De La Cruz	Department of Education - Province of Rizal, Philippines
Jing Zhu	Guangxi Normal University
Xiulin Ma	Beijing Normal University

Jamil Itmazi	Palestine Ahliya University, Palestine
Mouza Alshemaili	Higher Collage of Technology, UAE
Xiongjun Zhou	Sichuan Normal University
Guang Chen	Beijing Normal University
Xiaoya Yu	Beijing Institute of Education

Paper Submissions to CTE-STEM 2024

The conference received a total of 174 submissions (72 academic paper and 102 teacher cases) by 322 authors from 11 countries/regions (see Table 1).

Country/Region	No. of Authors	Country/Region	No. of Authors
Belgium	7	Macao	2
China	265	Malaysia	2
Hongkong	11	Netherlands	6
India	11	Singapore	2
Israel	2	United States	4
Japan	10	Total	322

The International Program Committee (IPC) is formed by 41 members and 8 co-Chairs worldwide. Each paper with author identification anonymous was reviewed by at least two IPC Members or co-chairs. Meta-reviews then made recommendation on the acceptance of papers based on IPC Members' reviews. With the comprehensive review process, 78 accepted papers (20 academic full paper, 22 academic short paper and 36 teacher cases) are presented at the conference.

Table of Contents

An Exploratory Study of the Relationship between Fixed/Growth Mindset and Computational Thinking among University Students

Masanori FUKUI[1*], Li XIANG[2], Yudai SANO[3], Wanda Nugroho YANUARTO[4], Subuh ANGGORO[5], Peter CHEW[6], Eng Tek ONG[7], Khar Thoe NG[8]
[1*]Iwate Prefectural University, Japan.
[2]Chongqing Institute of Foreign Studies, China.
[3]Naruto University of Education, Japan.
[4,5]Universitas Muhammadiyah Purwokerto, Indonesia.
[6]PCET Multimedia Education, Malaysia.
[7,8]UCSI University, Malaysia.
[*]fmasanori@aol.com, lixiangchristina@gmail.com, 22851039@naruto-u.ac.jp, wandanugrohoyanuarto@ump.ac.id, subuhanggoro@ump.ac.id, peterchew999@hotmail.my, OngET@ucsiuniversity.edu.my, postgradreview@gmail.com

ABSTRACT

This study explores the relationship between computational thinking and fixed/growth mindset, with the aim of informing educational strategies that enhance computational thinking. A survey of 578 university students was conducted to examine the relationship between mindset and factors of computational thinking. The results revealed significant positive correlations between mindset and computational thinking factors: creativity, algorithmic thinking, cooperativity, critical thinking, and problem-solving. Notably, the relationship between critical thinking and mindset emerged as the strongest. The relationship between mindset and cooperative thinking exhibited a weaker correlation than other computational thinking factors. Furthermore, significant differences in scores were observed between the growth mindset group and the fixed mindset group for each computational thinking factor. These findings suggest that students with higher creativity, critical thinking, and problem-solving scores are more likely to belong to the growth mindset group. In addition, cooperation may affect the probability of belonging to the higher mindset group, whereas algorithmic thinking shows no significant effect on mindset. Considering these findings, we propose a curriculum grounded in the practical aspects of computational thinking and mindset.

KEYWORDS
Computational thinking, fixed mindset, growth mindset, cognitive process, students' belief

1. INTRODUCTION

1.1. Research Background

In today's context, problems to be solved have become increasingly complex and sophisticated, requiring solutions beyond conventional knowledge and experience. Students are now expected to learn independently and tackle unknown problems without giving up. When solving problems using AI and other technologies, students must solve them efficiently within predetermined time frames. Computational thinking is important as it equips students with the skills to apply AI and programming solutions efficiently.

Since Wing (2006) introduced the key constructs of computational thinking outlined by Papert (1993), extensive research has been conducted in this area, resulting in numerous practical examples. Computational thinking plays a significant role in developing human resources to address social problems.

Several definitions of computational thinking (*e.g.*, International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA), 2011; Yadav et al., 2014) and several studies identifying sub-concepts of computational thinking (*e.g.*, Selby & Woollard, 2013; Wing, 2011; Angeli et al., 2016) exist. In other words, focusing on sub-concepts and activities in developing computational thinking is important.

Winthrop (2016) categorizes computational thinking into four main groups: data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices. Kalelioglu (2016) conducted a qualitative content analysis, revealing that the main topics addressed in the papers focus on activities promoting computational thinking in the curriculum, whether computerized or unplugged. Game-based learning and constructivism form the basis of these papers and their main theories. The analysis further identifies the most frequently used words in the definition and scope of computational thinking, forming the framework for computational thinking.

To enhance the development of computational thinking, it is imperative not only to accumulate examples but also to develop curricula tailored to learners' actual conditions, taking into account their psychological state and the relationship with various skills they possess.

This study focuses on "mindset" as a psychological state, drawing on Dweck's (2008; 2014) distinction between fixed and growth mindsets. Individuals with a fixed mindset believe their abilities are static and tend to avoid challenges. On the other hand, those with a growth mindset believe in improving their abilities through effort and learning. The study posits that a fixed or growth mindset influences learning attitudes and outcomes.

Mindset plays a significant role in learning; Limeri (2020) examined the impact of student's academic motivation and engagement on mindset development. The study found varying degrees of mindset orientation throughout an introductory computer science course, with certain groups leaning more toward a fixed mindset. This suggests that students' motivational characteristics may influence mindset development. Jiang et al. (2023) conducted a study that focused on the positive effects of a growth mindset on students' intentions toward self-regulated learning during the COVID-19 pandemic. The results show that students' growth mindset is associated with the support they receive from teachers and other involved individuals, even in an online setting. This study underscores the importance of a growth mindset in an educational context.

Since mindset significantly impacts learning effectiveness and attitudes toward learning, it is likely to play a crucial role in classes and practices aimed at fostering computational thinking.

1.2. *Identification of Problems*

It has been emphasized that the relationship between computational thinking and mindset is important for enhancing future computational thinking development. For instance, Stella et al. (2021) studied the relationship between computational thinking and mindset. The study investigated data science and computational thinking by examining the mindsets of high school students enrolled in a STEM-focused curriculum and STEM researchers working on modeling complex systems. The results indicated that STEM professionals exhibited consistency with important aspects of computational thinking, such as logical reasoning and positive attitudes toward data and simulation. In contrast, high school students demonstrated knowledge of logical reasoning but needed a more developed understanding of the relationship between models, simulation, and computation. This difference underscores the importance of computational thinking, interpreted by the authors as indicative of a gap in the development of the computational mindset. They argue that this partially undeveloped computational mindset requires additional psychological impetus to solve problems and understand the world.

Asmara (2020) researched computational thinking as a problem-solving skill, surveying international students from Taiwan's engineering and social science schools. The study found differences between engineering and social science students in their problem-solving approaches, especially in using structured algorithms. This fundamental insight sheds light on how differences in academic backgrounds affect students' computational thinking and problem-solving skills. Lodi (2017) emphasized the importance of interventions that stimulate a CS growth mindset for students and teachers, recognizing this as a fundamental and valuable area of computer science education research.

Although the above studies have reported on computational thinking and mindset, they have yet to focus on the difference between a growth mindset and a fixed mindset. They have been examined in a limited context. To enhance computational thinking in the future, it is essential to focus on the relationship between computational thinking, growth mindset, and fixed mindset and to conduct research based on the relationship between them. For this purpose, it is necessary not only to accumulate concrete examples but also to grasp the actual conditions of learners' computational thinking and mindset and to examine the relationship between them. However, to the best of the authors' knowledge, no previous studies have addressed this specific aspect.

In this study, we formulate the following research questions to understand the relationship between computational thinking and mindset and to obtain basic knowledge for the future enhancement of computational thinking development.

RQ: What is the relationship between computational thinking, fixed mindset, and growth mindset?

2. METHOD

2.1. *Survey Participants and Survey Method*

Rakuten Intage, a research company, conducted the survey. The participants comprised 578 university students (295 males and 283 females), with a mean age of 21.30 years (SD 2.03) and a survey duration of about 20 minutes. The participants remained anonymous, with no inclusion of names, school names, or other personally identifiable information. Upon completing the survey, participants received the designated points from this company.

2.2. *Survey Items*

For demographic information, we included items to ascertain age and gender. The Japanese version of the mindset scale, developed by Dweck et al. (2008) (Muto, 2020), was used to measure both fixed and growth mindsets (refer to Table 1). Although the scale's validity has yet to be examined, we used it for this study. Items 1 to 3 measure fixed mindset (invert items), and items 4 to 6 measure growth mindset. All items were answered on a 6-point Likert scale ranging from "6: Strongly agree" to "1: Strongly disagree." The sum or average of all items can measure the participants' mindset. The total or average value of all items can be used to measure the mindset of the participants. In this context, it is assumed that participants with values above the mean have a growth mindset, while those with values below the mean have a fixed mindset (Hong et al., 1999).

Table 1. Items of Fixed/Growth Mindset.

1	You have a certain amount of intelligence, and you really can't do much to change it. *
2	Your intelligence is something about you that you can't change very much. *
3	You can learn new things, but you can't really change your basic intelligence. *
4	No matter who you are, you can change your intelligence a lot.
5	You can always greatly change how intelligent you are.
6	No matter how much intelligence you have, you can always change it quite a bit.

*Invert Items

We used the Japanese version of the computational thinking scale developed by Chikazawa et al. (2022), adapted from the original scale by Korkmaz et al. (2017), to assess computational thinking. The reliability and validity of this scale have been previously examined. The original computational thinking scale consisted of 29 items, but when applied to humanities students, it was reduced to 21 items. However, for this survey, it is impossible to specify whether the participants are science majors or humanities majors. Therefore, we opted for the Japanese version of the 29-item computational thinking scale, comprising five factors: "creativity (CR)," "algorithmic thinking (AT)," "cooperativity (CO)," "critical thinking (CT)," and "problem-solving (PS)." All items were answered on a 5-point Likert scale ranging from "5: Strongly agree" to "1: Strongly disagree."

2.3. Analysis Procedure

A confirmatory factor analysis served as a preliminary analysis to examine the validity of the items used in this study. Following this, descriptive statistics were obtained. Since no normality was observed between the mindset scale and the computational thinking scale, the correlation coefficient between the mindset scale and each factor of computational thinking was calculated. The upper and lower groups were defined based on the mean value of the mindset scale, where the upper group refers to the "growth mindset group," and the lower group refers to the "fixed mindset group." The differences in each factor of computational thinking between the upper and lower groups were evaluated using the Wilcoxon rank-sum test. Subsequently, a two-way logistic regression analysis was conducted, taking the upper and lower mindset groups as dependent variables and each factor of computational thinking as an independent variable. This analysis aimed to understand how the factor of computational thinking affects the probability of belonging to a mindset group. The analysis was performed using R version 4.3.2 with a statistical significance level set at 5%.

2.4. Ethical Considerations

In this study, we did not include items that could identify individuals, such as names and school names, and respondents experienced minimal psychological burden while answering the survey items. Consent for participation was obtained at the time of response. This study received approval from the Tokushima University Ethics Review Committee (No. 2023-3). The authors declare no conflicts of interest.

3. RESULTS

3.1. Preliminary Analysis

As a preliminary analysis, we examined the validity of the items and scales used in this survey. Confirmatory factor analysis was used to examine the validity of the GFI, CFI, SRMR, RMSEA, and Cronbach's alpha coefficient. In addition, the Shapiro-Wilk test was used to confirm the normality of each factor. The results are shown in Table 2.

Table 2. Results of Preliminary Analysis.

	α	CFI	GFI	RMSEA	SRMR	W
CR	0.77	0.88	0.95	0.09	0.05	0.98**
AT	0.87	0.95	0.96	0.10	0.05	0.98**
CO	0.84	0.98	0.98	0.13	0.02	0.98**
CT	0.81	0.98	0.99	0.07	0.02	0.98**
PS	0.66	0.95	0.98	0.07	0.03	0.99**
Mindset	0.68	0.41	0.73	0.34	0.21	0.98**

**$p < .01$ ($N = 578$)

Table 2 shows that the results of reliability coefficients and Cronbach's alpha coefficient for each computational thinking factor supported the scale's reliability and validity. However, all the values tended to be low for mindset. This suggests that the mindset scale may differ from standard interpretations in the target population of this study and that, in general, the scale needs to be reviewed and contextualized. It also implies that when using mindset scales, appropriate adjustments and supplementary explanations are necessary, considering the target population's characteristics and cultural background. However, the validity and reliability of the Japanese version of the mindset scale itself have yet to be confirmed. Although these are issues that need to be addressed in future research, the purpose of this study was to understand the relationship between the mindset scale and the computational thinking scale and to obtain basic knowledge for enhancing education to foster future computational thinking.

This study aimed to understand the relationship between the mindset and computational thinking scales. It is meaningful to proceed with the analysis using the mindset scale to achieve this goal. Therefore, we decided to conduct the research using the mindset scale in this study. Since it was not confirmed that each factor and mindset scale of computational thinking had normality, we used nonparametric analysis in the subsequent studies.

3.2. Descriptive Statistics

Descriptive statistics are shown in Table 3.

Table 3. Results of Descriptive Statistics.

	Mean	SD
CR	3.56	0.76
AT	3.41	0.62
CO	3.02	0.87
CT	3.35	0.91
PS	3.19	0.80
Mindset	2.96	0.67

($N = 578$)

3.3. Relationships between the Mindset Scale and Each Factor of Computational Thinking

Spearman's rank correlation coefficients were computed to examine the relationship between the mindset scale and each factor of computational thinking. The results are shown in Table 4.

Table 4. Results of Correlation Analysis.

	CR	AT	CO	CT	PS
CR	1.00	--	--	--	--
AT	0.42**	1.00	--	--	--
CO	0.45**	0.28**	1.00	--	--
CT	0.60**	0.54**	0.39**	1.00	--
PS	-0.06	-0.05	-0.14**	-0.03	1.00
Mindset	0.31**	0.23**	0.18**	0.32**	0.29**

**$p < .01$ ($N = 578$)

Table 4 shows a moderate to high correlation among creativity, algorithmic thinking, cooperativity, and critical thinking, reflecting correlations among the factors of computational thinking. Critical thinking showed a strong correlation with algorithmic thinking and cooperativity. On the other hand, problem-solving was weakly correlated with other factors of computational thinking, suggesting that it may have its unique elements.

Next, as for the relationship between mindset and the factors of computational thinking, significant positive correlations were found between mindset and creativity, algorithmic thinking, cooperativity, critical thinking, and problem-solving. The relationship between mindset and cooperativity was weaker than that between mindset and other factors of computational thinking, but it was still significant.

3.4. Differences in Computational Thinking Due to Differences in Mindset

To examine the differences in each factor of computational thinking based on mindset differences, we analyzed the differences between the upper group (growth mindset group) and the lower group (fixed mindset group) using the Wilcoxon rank-sum test. The results are shown in Table 5.

Table 5. Results of. Differences in Computational Thinking Based on Mindset.

	Upper Group		Lower Group		W
	Mean	SD	Mean	SD	
CR	3.59	0.58	3.25	0.60	54418 **
AT	3.18	0.89	2.87	0.83	50215 **
CO	3.53	0.87	3.19	0.91	50170 **
CT	3.43	0.73	2.97	0.80	54554 **
PS	3.10	0.70	2.84	0.63	50195 **

**$p < .01$ ($N = 578$)

Table 5 suggests that students with a high or growth mindset score higher on all the computational thinking factors: creativity, algorithmic thinking, cooperativity, critical thinking, and problem-solving. In other words, a high level of mindset functions as a factor that positively influences each factor of computational thinking.

3.5. Impact of Computational Thinking on Fixed/Growth Mindset

A two-way logistic regression analysis was conducted to examine the computational thinking factors' influence on the growth and fixed mindset groups. The upper and lower mindset groups were used as dependent variables, and each computational thinking factor was used as an independent variable. The results are shown in Table 6.

Table 6. Results of Two-way Logistic Regression Analysis.

Factor	Estimate	Std. Error	z value	Pr (>|z|)
Intercept	-6.95	0.88	-7.92	0.00 **
CR	0.57	0.21	2.76	0.01 **
AT	0.02	0.13	0.15	0.88
CO	0.23	0.12	1.90	0.06
CT	0.52	0.17	3.01	0.00 **
PS	0.81	0.16	5.24	0.00 **

**$p < .01$ ($N = 578$)

In Table 6, the coefficients of creativity, critical thinking, and problem-solving are positive and significant, indicating that higher scores in these factors increase the probability of belonging to the growth mindset group. The coefficient of cooperativity is also positive and tends to be significant, a factor that may increase the probability of belonging to a group with a higher mindset. On the other hand, the coefficient of algorithmic thinking is not statistically significant and has no significant effect on growth and fixed mindset.

4. DISCUSSION

These results suggest that students with higher mindsets tend to exhibit higher scores and abilities in each aspect of computational thinking. Critical thinking shows a particularly strong association with mindset. Although the results are complex and difficult to decipher, all factors affect mindset when focusing on a single factor of computational thinking. However, the growth mindset group's influence on the mindset differed depending on the factor. The growth mindset group demonstrated higher creativity, critical thinking, and problem-solving scores.

In contrast, the effects of algorithmic thinking and cooperativity on the mindset group were not significant. In other words, educational programs and teaching methods focusing on the relationship between mindset and computational thinking may need to differ between the high and low mindset groups. In addition to focusing on the relationship between a single factor of computational thinking and mindset, it is necessary to emphasize the relationship between computational thinking and mindset and develop curricula based on this reality.

The fact that algorithmic thinking's influence on mindset was not significant when considering the whole of computational thinking suggests that this ability may be influenced by factors other than mindset. The weak correlation between mindset and cooperativity also indicates that cooperative activities have characteristics different from other elements of computational thinking, which should be considered in the design of educational curricula.

5. IMPLICATIONS FOR CLASSROOM ACTIVITIES

Adopting flexible and effective teaching methods based on students' mindsets and computational thinking abilities is important in designing educational programs. Specifically, the significant positive correlations between the computational thinking factors and mindset underscore the need for an individualized approach in educational programs.

The strong association of critical thinking skills with mindset has an important implication for educational settings. The results indicate that fostering students' ability to evaluate their thinking processes and consider issues from different perspectives may promote a growth mindset. Therefore, it may be beneficial to provide students with opportunities for self-evaluation and self-reflection in the classroom, along with incorporating activities that stimulate critical thinking.

On the other hand, the relatively weak correlation between cooperativity and mindset indicates that educational interventions focused on cooperative activities may not directly impact mindset development, or a reverse relationship may exist. Educators need to recognize this and pay attention to both the development of problem-solving skills and mindset. However, since the correlation between cooperativity, creativity, and critical thinking was observed, it is possible that the setting of combined activities, such as cooperative problem-solving activities rather than cooperativity alone, may be effective in improving one's mindset as well. Nonetheless, further investigation is need due to the potential negative relationship between cooperativity and problem-solving.

Furthermore, significant differences in computational thinking scores between the growth and fixed mindsets highlight the importance of adjusting the educational program based on the student's mindset level. In other words, based on the reality of mindset and computational thinking, the following approaches are assumed to be effective in meeting the needs of each group.

Special support and approaches that help students develop their mindset and computational thinking skills, such as creativity and critical thinking, may be effective for students with a low mindset. Supporting students in developing their self-confidence, motivating them to take on new challenges, and promoting a growth mindset should be prioritized. Educators should teach students the concept of a growth mindset, emphasize the importance of accepting failure as part of the learning process, and encourage them not to shy away from challenges. Focusing on the relationship between activities that promote a growth mindset and computational thinking, providing simple computational thinking-related tasks, and offering appropriate support are essential.

On the other hand, lessons and curricula for students with higher mindsets may be more suitable for those who aim to improve their critical thinking and creative problem-solving skills. For example, activities stimulating students' creativity and problem-solving skills may be practical by providing complex and challenging computational thinking-related tasks. Introducing new concepts and advanced techniques may arouse students' interest and curiosity. Furthermore, it is expected that students will be able to choose their projects and research topics and develop deeper understanding and application skills through self-directed learning and inquiry activities.

It may be helpful to promote critical thinking and reflection, provide opportunities for students to evaluate and reflect on their work and thought processes, and further develop critical thinking skills through discussion and presentation.

Common elements of the above curriculum that accommodate both the highs and lows of mindset include providing enhanced feedback and support for each learner based on actual computational thinking and mindset, offering appropriate feedback and instruction for all students, and providing individualized progression. This approach effectively provides appropriate feedback and support to all students and supports students according to their learning progress.

As described above, the educational field requires the design of diverse and comprehensive educational programs that focus on mindset and computational thinking. This is expected to realize education tailored to each student's needs and maximize all students' potential.

It is important to note that some analyses of the relationship between mindset and the factors of computational thinking treat them as causal relationships. Still, they are compound correlations, and the relationship between mindset and computational thinking needs to be clarified. The relationship between mindset and computational thinking needs to be clarified. Therefore, it would be effective to consider these points in curriculum development and create a flexible form that focuses on both mindset and computational thinking. Remember that this discussion is intended to present a case study, not a causal relationship, and that correlation does not imply causation. Additionally, other potential factors may exist.

6. SUMMARY AND FUTURE WORK

6.1. Summary of This Study

This study aimed to examine the relationship between computational thinking and fixed/growth mindset and obtain basic knowledge for enhancing education that enhances mindset and computational thinking.

In this study, we clarified the relationship between mindset and computational thinking and showed the complex relationship between high/low mindset and each factor of computational thinking. The results offer valuable insights for the educational field.

In contrast to prior studies concentrating on the relationship between mindset and computational thinking, this study was conducted to understand the actual conditions of students' growth/fixed mindset, and computational thinking, aiming to delineate a specific direction for the curriculum. The direction of the curriculum was concretely indicated through a survey study, providing insights into genuine circumstances of students' fixed/growth mindset, and computational thinking. This research is unprecedented; the results are novel, innovative, and original. The survey encompassed a substantial number of participants, ensuring its reliability is assured and contributing to the advancement of education for nurturing computational thinking in the future.

6.2. Limitations of This Study

Although the results of this study are important findings for enhancing the development of computational thinking, several limitations need consideration, as they may impact the interpretation of the results of this study and the direction of future research.

Firstly, although our study found significant correlations between mindset and each component of computational thinking, it is important to note that correlations do not imply causality. In other words, these relationships may be influenced by other potential variables not considered in the study, necessitating exploration of these potential factors. Secondly, within the realm computational thinking, the impact of algorithmic thinking on the mindset group was found to be insignificant. Thirdly, this study focused on specific computational thinking elements and mindsets, and it is challenging to generalize to all aspects of these complex structures. Future research should explore the relationship between other dimensions of computational thinking and different types of mindsets.

Subsequent research endeavors should involve the development of educational practice programs grounded in the findings of this study, with a subsequent examination of their effectiveness in practice. Thus, further comprehensive research is warranted to advance our understanding in this field.

7. ACKNOWLEDGEMENTS

This work was partially supported by JSPS KAKENHI (Grant Number 21K13644, 22KK0200) and the Inamori Foundation.

8. REFERENCES

Asmara, A. (2020). Study on Computational Thinking as Problem-solving Skill: Comparison Based on Students Mindset in Engineering and Social Science, *arXiv*, 2007.04060.

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.

Chikazawa, Y., Moriyama, J., Takahashi, K., Morihiro, K., Kakegawa, J., Ogawa, H., Nakahara, H., & Usami, M. (2022). Development of the Japanese Version of the Computational Thinking Scales for First-Year University Students in Humanities, *Japan Journal of Educational Technology*, 46(1), 103-114. [in Japanese]

Dweck, C. S., Walton, G. M. & Cohen, G.L. (2014). Academic Tenacity -Mindsets and Skills that Promote Long-Term Learning, Retrieved January 9, 2024, from https://ed.stanford.edu/sites/default/files/manual/dweck-walton-cohen-2014.pdf.

Dweck, C. S. & Master, A. (2008). *Self-Theories Motivate Self-Regulated Learning*, In Schunk, D. H. & Zimmerman, B. J.(Eds), Motivation and Self-Regulated Learning, Taylor & Francis.

Hong Y., Dweck C.S., Chiu C., Lin D. M. S. & Wan W. (1999). Implicit Theories, Attributions, and Coping: A Meaning System Approach, *Journal of Personality and Social Psychology*, 77(3), 588-599.

International Society for Technology in Education (ISTE) & the Computer Science Teachers Association (CSTA) (2011). *Operational definition of computational thinking for K–12 education*, Retrieved January 9, 2024, from https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf.

ISTE (2016), *ISTE Standards for Students*, Retrieved January 9, 2024, from http://www.iste.org/docs/Standards-Resources/iste-standards_students-2016_one-sheet_final.pdf?sfvrsn=0.23432948779836327.

Jiang, Y., Liu, H., Yao, Y., Li, Q., & Li, Y. (2023). The Positive Effects of Growth Mindset on Students' Intention toward Self-Regulated Learning during the COVID-19 Pandemic: A PLS-SEM Approach. *Sustainability*, 5(3):2180.

Kalelioglu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem-solving skills. *Baltic Journal of Modern Computing*, 4, 583-596.

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

Limeri, L.B., Carter, N.T., Choe, J., Harper, H. G., Martin, H. R., Benton, A. & Dolan, E. L. (2020). Growing a growth mindset: characterizing how and why undergraduate students' mindsets change. *IJ STEM Ed*, 7, 35.

Stella, M., Kapuza, A., Cramer, C., & Uzzo, S.(2021). Mapping computational thinking mindsets between educational levels with cognitive network science, *Journal of Complex Networks*, 9(6).

Lodi, M. (2017). Growth Mindset in Computational Thinking Teaching and Teacher Training. *In Proceedings of the 2017 ACM Conference on International Computing Education Research (ICER '17). Association for Computing Machinery*, New York, NY, USA, 281–282.

Muto, H. (2020). Mindsets: What Having a Growth Mindset Affects Student Learning?, *Bulletin of the Graduate School of Education of Waseda University*, 28(1), 71-82. [in Japanese]

Papert, S. (1993). *Children's Machine: Rethinking School in the Age of the Computer*, New York, NY: BasicBooks.

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *J Sci Educ Technol*, 25, 127–147.

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

Wing, J.M. (2011), *Research Notebook: Computational Thinking -what and why?*, The Link Magazine, 20-23. Retrieved January 9, 2024, from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education, *ACM Transactions on Computing Education*, 14(1), 5.

The Effect of Music Producing on Computational Thinking Among Primary School Students in Childcare

Nardie FANCHAMPS[1*], Emily van GOOL[2] , Jerôme van DONGEN[3] , Metske STRAUS[3] , Kim de MEYST[4]
[1]Open University, the Netherlands
[2]Fontys University of Applied Sciences, the Netherlands
[3]MIk & PIW Groep, the Netherlands
[4]Image-Line, Belgium
nardie.fanchamps@ou.nl, evangool57@gmail.com, jerome.vandongen@mik-piwgroep.nl, metske.straus@mik-piwgroep.nl, kim@image-line.com

ABSTRACT
Stimulating creativity can make an important contribution to computational thinking (CT) of elementary school students who attend childcare after school. Since learning goes beyond time spent in school, it makes sense to offer students in childcare-centres meaningful CT-activities that exceed more common applications of technology for CT-development, like programmable robots or serious games. Such approaches provide, besides a focus on visual modality, promising opportunities emphasising auditory modality. Reasoning from such different contextual perspective, it's interesting to explore whether self-creation of electronic music using music-producing software has an effect on CT-development, and what added value can be attributed to creativity. Therefore, an experimental mixed-methods-study was conducted among elementary school students aged 10 to 13 within a childcare setting using FL-Studio© music producing software. Quantitative data were obtained by pretest-posttest-assessment administering the validated Computational Thinking test (CTt). Qualitative data were obtained conducting interviews and observations during and after each session asking questions to what magnitude students grasped CT sub-characteristics, to what extent creativity and creative thinking played a role, and what perceptions students themselves had in this regard. Our results indicate that applying music-making software has a significant effect on CT-development where the focus is on the invocation and utilisation of auditory modality. Remarkable effects could be identified on CT-(sub)characteristics 'loops,' 'conditionals,' 'nesting,' and required CT-tasks. Our study also found that technology-enhanced music producing stimulates creativity, which appears an important parameter regarding CT-development. It is recommended to conduct further research on the intersection between CT and creativity using combinations of different modalities.

KEYWORDS
Technology-enhanced learning, computational thinking, creativity, music producing, output modalities

1. INTRODUCTION
A variety of plugged-in and unplugged approaches offer effective opportunities regarding the development of computational thinking (CT). Frequently this involves more traditional applications of technology such as programmable artefacts like robots, or the use of physical board games. The intervention and subsequent determination of the effect of programming thereby occurs primarily by means of visual perception through observation, interpreting icons, reading text, writing syntax or by physical, kinaesthetic experience. Furthermore, fostering creativity appears to be an important parameter in the development of CT. Previous studies show that creative thinking and creative action can lead to extraordinary discoveries in solution processes for challenging problems, and that the learning environment used can be conditional for this. It is therefore valuable to investigate whether less common approaches such as appealing to auditory modality can also have a distinctive impact on the (further) development of CT, and what important influence can be attributed to creativity in this regard.

Previous studies indicated a link between music making and CT, to which combining self-producing music and programming into one activity enables an identifiable development on CT (Chong, 2018; Petrie, 2019). There is also evidence for interrelationships between CT and creativity (Israel-Fishelson & Hershkovitz, 2022) and point to positive effects of creativity through the application of educational robotics on CT development (Chevalier et al., 2020; Noh & Lee, 2019). In addition, research also distinguishes between two types of creativity, namely: creative thinking and computational creativity, where creative thinking should be seen as the innovative process of solving challenging problems, and where computational creativity is characterised by applications of computer technology to mimic, study, stimulate and enhance human creativity (Israel-Fishelson et al., 2021).

Our experimental mixed-methods design study focuses on whether self-creation of electronic music using technology enhanced music-producing software has an effect on CT-development, and what added value can be attributed to creativity. Participants were 8 primary school students who attend childcare after their regular schooling and in which CT proficiency was assessed using the validated Computational Thinking Test (CTt) (Román-González et al., 2017) targeted to students from 10 to 16 years old.

2. METHODOLOGY
This study was conducted among 8 primary school students aged 10 - 13 years in after school childcare in the Netherlands, focusing on the last cycle of primary education (grades 5 and 6). In five sessions of one hour each, participating students worked together in dyads to produce an electronic dance music track (Edm) using the professional music producer software FL-Studio© (Figure 1 and Figure 2). These five sessions were held at the

childcare location where students attend after their regular school day. In each of the five sessions an instruction part - processing part - own design part – in-between & end evaluation part - and appreciation part were applied.

Figure 1. Students Composing Music.

Figure 2. Impression FL-Studio© Music Producing Software.

All students first received a basic instruction on how the music producing software works, how music is arranged, in which structure an Edm-track is designed (intro - break - build-up, drop - outro), how samples and instruments from the music library can be dragged into the worksheet, how to add effects to it, etc. After all the basics were explored and mastered, students were given the task of creating their own Edm-track where they could independently make their own choices in terms of rhythm, instruments, percussion, sound samples, vocals, effects and volume settings to be selected. Based on their own creative disposition, students could then make their own choices and decide how they wanted to shape the Edm-track to be produced. Students could increasingly take their own initiative and direction, each time using their own creativity within the musical parts to be designed, with the final goal of a completely elaborated Edm-track.

Before and after the study, students where administered the validated CT questionnaire (CTt) that assessed their proficiency on CT. This questionnaire, based on visualisations, pictograms and representations using arrows and Scratch-based programming images, consists of 28 items in which knowledge regarding the computational concepts addressed *"loops, conditionals and functions"*, the presence of "nesting", and the required CT-tasks *"debugging, completion and sequencing"* can be determined (Figure 3 and Figure 4).

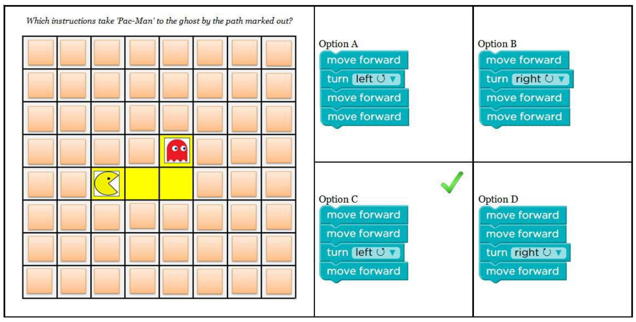


Figure 3. Example Question CTt-Questionnaire.

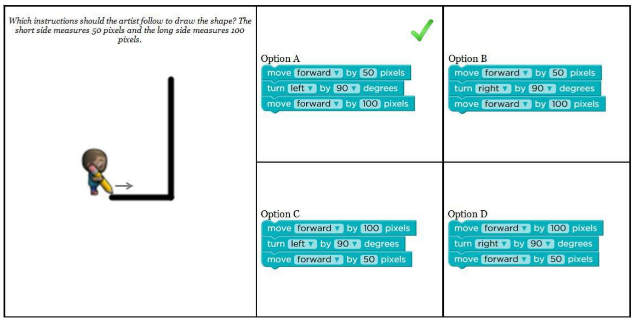


Figure 4. Example Question CTt-Questionnaire.

Regarding the relationship between CT and creativity, we apply the same principles as formulated by Brennan & Resnick (2012) in their 3D-framework. In it, a relationship between CT and creativity is assumed. In our study, in which technology-enhanced music tracks are composed, we focus on developing CT-skills and include the findings of a creative attitude. We ground on studies by Doleck et al. (2017) and Rotem et al. (2020) which demonstrate that creativity is strongly related to CT. We focus on the perception of creativity through flexibility, analytical problem solving, entrepreneurship, perseverance, imagination and cooperation (Fukui et al., 2022). Therefore, in addition to quantitative data collected, qualitative data were also established regarding the importance of creativity and a development on CT. For this purpose, a survey was prepared consisting of 14 questions according to a 3-point Likert scale (‘*not true - sometimes – true*’) including 3 open-ended questions, that was benchmarked prior to the study with a focus group of educational subject matter experts. By administering this survey after each session, each individual student was questioned to what extent he/she had a grasp of CT sub-characteristics, to what extent creativity and creative thinking played a role, and what perception the students themselves had in this regard. This involved using questions such as: *"Do you remember the arrangement you used in your work?"*; *"Were you able to understand and organize the information you were given?"*; *"Were you able to find and also resolve/debug errors?"*; *"In what ways were you creative? Where does that show?"*. Based on the survey data obtained, an analysis was conducted that reflected a more detailed development of the students in addition to an analysis of the CTt-questionnaire.

Furthermore, and in addition to the questionnaire and survey administered, free observations were carried out during each session in which approach behaviour, the level of cooperation between students and other worthwhile perceptions were recorded.

3. RESULTS

First, it can be stated that based on the chosen approach, all students were well able to design and produce an Edm dance track independently from their own creative perspective. Through the structured setup, each student managed to apply all the necessary parts of the music producer software. Selecting chords, designing a melody, choosing own samples and instruments, assigning them to the mixer, arranging them in the playlist and then building the track in layers worked well. Adding markers to build and structure the track also made it feasible and manageable. All students managed to design and arrange a full Edm-track in five sessions. As a result, the musical results are all different, extraordinary and appealing.

Second, regarding a determinable development on CT, an analysis of the data obtained from the validated CTt-questionnaire by studying the means (Table 1) shows that students score better on all measured characteristics in the post-test in a comparison with the pre-test.

Table 1. Analysis CTt-Questionnaire.

Variables	Pre-test		Post-test	
	M	*SD*	*M*	*SD*
Total (28)	.54	.149	.70	.147
Loops: repeat times	.49	.159	.66	.116
Loops: repeat until	.48	.182	.60	.198
Loops: combined	.48	.151	.63	.149
Conditionals: if/simple	.35	.243	.54	.173
Conditionals: if-else	.47	.411	.66	.352
Conditionals: while	.34	.129	.56	.116
Conditionals: combined	.39	.183	.59	.183
Functions	.53	.452	.78	.248
Nesting	.41	.190	.62	.177
CT-skill: completion	.53	.165	.67	.214
CT-skill: debugging	.53	.212	.73	.237
CT-skill: sequencing	.56	.181	.71	.132

Note: Variable = measurable value; total = number of questions Correct CTt questionnaire; computational concept addressed = loops, conditionals, functions, nesting; completion = completed by CT; debugging = reformulating of problems; sequencing = sequence; M = mean-value; SD = standard deviation.

To determine at which of the different CT (sub)characteristics a significant effect was measurably caused by the intervention and what measurable effect could be attributed to it, a paired-sample *t*-test was performed (Table 2). This conducted analysis shows that significant differences were found for the full CTt-questionnaire (‘Total’), as for the CT (sub)characteristics: a) ‘loops - repeat times, simple, combined’; b) ‘conditionals - if/simple, while, combined’; c) ‘nesting’; and d) for the CT concepts addressed - ‘debugging & sequencing’. From the data presented, by calculating Hedges *g*, it can also be inferred that the intervention applied achieves a small (g = 0.2) to low-medium effect (g = 0.5) on the various sub-characteristics of CT.

Table 2. Development of Computational Thinking.

Variable	Paired Sample *t*-test (n = 8)		
	t	*p*	*g*
Total (28)	-5.32	.001*	.094
Loops: repeat times	-4.58	.003*	.120
Loops: repeat until	-3.00	.020*	.133
Loops: combined	-5.30	.001*	.090
Conditionals: if/simple	-2.55	.038*	.234
Conditionals: if-else	-2.05	.080	.291
Conditionals: while	-3.86	.006*	.180
Conditionals: combined	-4.89	.002*	.129
Functions	-2.16	.068	.369
Nesting	-4.43	.003*	.147
CT-skill: completion	-2.12	.072	.209
CT-skill: debugging	-2.65	.033*	.241
CT-skill: sequencing	-3.66	.008*	.132

Note: Variable = measurable value; total = number of questions correct CT questionnaire; computational concept addressed = loops, conditionals, functions, nesting; completion = completed by CT; debugging = reformulating of problems; sequencing = sequence; M = mean-value; t = t-value paired samples; p = p-value paired samples; g = effect size based on Hedges g for different sample sizes; * = significant effect measured (≤ .05).

Third, a further analysis of the administered survey asking questions regarding CT sub-characteristics (*'problem (re)-formulation, data collection, data analysis, data visualisation, problem decomposition, abstraction, algorithms and procedures, automation, simulation and modelling, and parallelisation'*) reveals in general that, as the number of sessions progressed, the scores for recognition ('true') increased and even doubled in total when comparing the findings from the first with the last session. We notice that the characteristics 'problem decomposition' (breaking down a task into smaller orderly tasks) and the component 'algorithms and procedures' (using a series of ordered steps to solve a problem) were rated "sometimes" the most by students. Interestingly, the CT sub-characteristic ‘analysing data’ (logically arranging, analysing and understanding data) received high scores. Also, the CT sub-characteristic ‘visualise data’ (being able to use and process information) predominantly showed the highest score among students in all meetings. Furthermore, the scores for ‘not true’ decreased by 33 %. We could perceive that during the experiment, students started to self-recognise the skills related to CT more. The level of decrease in the ‘not true’ component reflects this shift.

Fourth, in order to determine a development on creativity and creative thinking, students answered three open-ended questions for each session. These questions were about (1) *The way students felt they had been creative*, (2) *The perceived extent of collaboration*, and (3) *Whether students could apply what they had learned to other subjects at school*. The first thing to note is that the responses to the questions became more specific and clear as the number of sessions progressed. For example, in the beginning the responses were, *"We tried a lot, you got to do it yourself and I got creative”*. At the last sessions, the responses were such as: *"I got to choose a lot myself and try a lot, we got to design our own beat, and we have a very different melody”*.

Fifth, similar comments are recorded in several observation reports. One observation report mentions that students are having an attentive conversation about what choices to make, and also that they save the composed Edm-track when they both like it. It is also observed that students comment on their track. In another observation, it was recorded that students can quickly pick up what they learned on their own, with the observation mentioning that students remember well what they did last time. Being able to apply what they learned to other subjects at school proved a difficult connection for students. There were responses such as: *"I can apply it to music, I can't apply it at school, but I can apply it at home”*. Yet there were two responses that made a connection. They formulated this as: *"I can apply it in discovering your own mistakes and sometimes I can apply it in reviewing at school”*. All observation reports noted a high level of cooperation among students. Even when cooperation resulted in a difference of opinion, work continued constructively. Observations from the observation reports included students discussing together, actively questioning each other, choosing together which music sample to use. They remind each other to adopt silence when explanations were given or questions were answered. One observation in particular stated: *In time, the Edm-tracks take a bit more shape, now the students get a bit more aware of the other pairs and the track they have made, want to hear it from each other.* Of particular note is another comment from students that was reflected in an observation report: *"Maybe next time we can really figure it out ourselves", "Yes, but we still have to learn it completely first"*.

Apart from all data obtained from the survey and the observations, the Edm-tracks created were also examined more closely to obtain an indication of the level of creativity and uniqueness. What is striking to report is that all Edm-tracks differ in choices selected, contain a totally different sample-usage, arrangement and combinations made, as well as showing extensive use of parallel arranging of music samples. Despite the fact that all tracks follow the same build-up and design-structure, all tracks have an originality and unique sound introduced by students themselves. The ability to select and combine available samples and instruments themselves certainly contributes to this, as does the facilitator's provision of space for the students’ own choices and creative process.

4. DISCUSSION AND CONCLUSIONS

Overall, this research shows that students are still eager to learn after a busy school day and demonstrates the opportunities provided by less obvious technology-enhanced environments for CT-learning. More specifically, a significant development in CT could be unambiguously identified through the use of music-producing software, where an application as such implies a clear affinity with programming and required underlying skills. The focus on the combination of both visual and auditory modality indicates interesting development opportunities in this regard. The chosen approach regarding working with FL-Studio© allows for active stimulation of CT, creativity development and creative thinking in addition to producing and making music. This appears to be conditional for approaching challenging problems open-mindedly and solving them creatively. The set-up and organisational form used shows that students can learn a lot from and with each other. This is reflected in the high rating the students themselves gave about their level of cooperation and also becomes insightful from the observation reports. Our findings pave the way for further research into the effect and impact music producing applications can generate and potentially contribute to students’ developmental potential.

4.1. Limitations and Further Directions

Despite the limited number of respondents in this study, a significant development on almost all CT sub-characteristics, computational concepts addressed and required computational tasks has been demonstrated. In follow-up research, larger numbers of respondents will be used for which it is expected that more impactful results can be demonstrated. It is also worthwhile to investigate whether other types of music producing software programmes generate the same yields. It is also interesting to explore whether other types of modalities (feel, taste, smell, etc.) or combinations made could add value to CT-development in a similar study.

5. ACKNOWLEDGEMENT

We would like to thank the students and supervisors of MIK & PIW Groep, specifically BSO location Montessori - MIK childcare Maastricht, the Netherlands for their cooperation. Also a special word of thanks to the Belgium-based music software company Image-Line Software in the person of Kim de Meyst for providing the digital audio workstation program FL-Studio©.

6. ETHICAL STANDARDS

The research ethics committee (cETO) of the Open University of the Netherlands has reviewed the proposed research and concluded that this research is in accordance with the laws, regulations and ethical codes for research involving human subjects (reference: Ceto_RP102/U202302855).

7. REFERENCES

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proc. of the 2012 Annual Meeting of the American Educational Research Association*. Vancouver, BC, Canada: AERA.

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: a model for creative computational problem solving. *International Journal of STEM Education*, *7* (1), 1-18.

Chong, E. K. (2019). Teaching and learning music through the lens of computational thinking. *International Conference on Art and Arts Education (ICAAE 2018)*. Yogyakarta, Indonesia: Atlantis Press.

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, (4), 355-369.

Fukui, M., Sasaki, Y., & Hirashima, T. (2022). An Exploratory Study of the Relationship between Computational Thinking and Creative Attitudes among University Students. *CTE-STEM 2022 Conference*. Delft, Netherlands: TU Delft OPEN Publishing.

Hershkovitz, A., Sitman, R., Israel-Fishelson, R., Eguíluz, A., Garaizar, P., & Guenaga, M. (2019). Creativity in the acquisition of computational thinking. *Interactive Learning Environments*, *27* (5-6), 628-644.

Israel-Fishelson, R., & Hershkovitz, A. (2022). Studying interrelations of computational thinking and creativity: A scoping review (2011–2020). *Computers & Education*, *176*, (104353), 1-22.

Israel-Fishelson, R., Hershkovitz, A., Eguíluz, A., Garaizar, P., & Guenaga, M. (2021). The associations between computational thinking and creativity: The role of personal characteristics. *Journal of Educational Computing Research*, *58* (8), 1415-1447.

Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development*, *68* (1), 463-484.

Petrie, C. (2022). Interdisciplinary computational thinking with music and programming: a case study on algorithmic music composition with Sonic Pi. *Computer Science Education, 32* (2), 260-282.

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.

Hierarchical Computer Science Curriculum in High School:A Practice in Beijing National Day School

Weiqi ZHANG[1*], Zijie ZHENG[2*], Siyao HAN[3], Xingming WANG[4]
[1,2,3,4]Beijing National Day School, China
zhangweiqi@bnds.cn, zhengzijie@bnds.cn, hansiyao@bnds.cn, wangxingming@bnds.cn

ABSTRACT

The People's Republic of China's Ministry of Education introduced the first K-12 Standard for high school computer science (CS) courses in 2017. CS course is also known as Information Technology course in high school period in China. However, the majority of Chinese high schools lack a systematic curriculum, operationalized courses, and experienced teachers, resulting in challenges in implementing the courses as per the Standard. To address this, our paper presents a hierarchical framework for CS courses at Beijing National Day School (BNDS), including compulsory, elective compulsory, and optional courses. We share a synopsis, examples, and student performance to offer insights for other high schools, particularly in developing countries, aiming to develop their own CS courses.

KEYWORDS

Computer science curriculum, Hierarchical curriculum framework, Information technology, Beijing National Day School

1. INTRODUCTION

Information technology and its influence in 5G, Artificial Intelligence, Internet of Things and Cloud Computing has been booming in decades. More and more countries have realized that the information technology plays an increasingly important role in the international competition. Besides reforming the college curriculum to train more professionals, the ministries of education in these countries have also published before-college national curriculum standards in computer science (CS) to guarantee a general education of computer programming and computational thinking around their societies (Every, 2015). In 2011, the Computer Science Teachers Association (CSTA) in U.S. published its first K-12 CS curriculum standard to provide the guidance and contents for the programming courses in primary schools and high schools (K-12, 2011). In 2016, CSTA revised the curriculum standard, in which the concept of computational thinking is emphasized and many advanced contents such as Data Analysis and Artificial Intelligence are added (K-12, 2016). In many other developed countries, such as U.K. and Australia, the ministries of education published the standards in last decade and required that each high school and primary school must offer compulsory CS courses for their students (National, 2013) and (National 2015) In Asia, Japan issued its new edition of curriculum standard in 2018, in which each high school is required to establish the optional programming courses for each student before 2021 (Learning, 2018).

In recent years, the State Council and the Ministry of Education of the People's Republic of China have also realized the significance of the before-university CS courses (Information Technology). In 2017, the Ministry of Education published the first edition of national curriculum standard (Standard) of CS courses in high school (National, 2017). Under a series of guidance documents published between 2018 and 2020 by the State Council such as *Education Modernization 2035* (Education, 2019), the Ministry of Education revised the Standard and published a new edition in 2020 (National, 2020). Compared with the CS curriculum standards in many other countries, the Standard in China emphasizes more on the "competence" than "contents".

In the Standard, four core competence which the high school students develop through CS courses are proposed: information consciousness, computational thinking, electronic learning and innovation, responsibility in information society Information consciousness is the sensitivity to information and judgment of information value. Computational thinking refers to the thought process involved in expressing solutions as computational steps or algorithms. Electronic learning and innovation is to the ability to solve problems creatively, complete learning tasks and form innovative works by evaluating and selecting digital resources and tools. Responsibility in information society is the responsibility of individuals in cultural cultivation, moral standards and self-discipline in the information society.

Based on four core competence, the Standard also illustrates a hierarchical framework of courses which can be operationalized and guide each high school to organize and practice its own CS courses. The framework includes three layers of courses: compulsory courses, elective compulsory courses, and optional courses. Each student in the high school must join the compulsory courses to learn the fundamental concepts of computing, data science, information system, and information society. In the elective compulsory courses, students can select two or three specific directions among 6 modules for further study: data structure, web science, data analysis, fundamental of artificial intelligence, 3D design, and open-source hardware project. For the students who are very interested in CS and tend to choose CS as the major in the university, they can select the optional courses such as the computing algorithms and mobile application design.

Although the Standard has provided a forward-thinking guidance in core competence and CS courses, the reality is far away from the ideal. Most high schools in China have difficulties in carrying out the entire CS course framework as the Standard expects. In Zhejiang province, even though the CS is an elective subject of National College Entrance

Examination in China, i.e., Gaokao, which has an equal status with biology and chemistry, most high schools can only provide compulsory courses for their students and train their students how to handle the exams rather than to teach the knowledge and improve the programming abilities (Gaokao, 2014). In other provinces, the situations are much worse, a number of high schools even compress or cancel their CS courses. The reasons are very complicated. For example, compared with the giant internet companies, the salary for staff in high school is relatively low. Thus, most high schools can hardly recruit capable teachers to offer CS courses. In addition, there are few systematic and suitable teaching experiences and materials in CS as the reference for high school teachers and students (D. Sun, 2019).

Fortunately, in some large cities in China, such as Beijing, Shanghai, and Shenzhen, a few high schools in the forefront of education reform have attached much significance to the CS sources. Some of them have practiced a complete framework of courses. In this paper, we briefly introduce the hierarchical and systematic CS courses and the teaching experience in BNDS, which is one of the top schools in China. About 12 percent of the 600 students in BNDS will enter the top 2 universities in China after high school each year. There are more than 10 CS teachers, most of whom has master or higher degrees. They are the very foundation to carry out this framework of courses.

Specifically, we first introduce the entire framework of courses. Then, we will provide how we develop and practice the compulsory courses, the elective compulsory courses, the optional courses, and the Innovative talents program, respectively. In each kind of courses, we provide a synopsis, some typical examples, and the performance of students. We hope that the framework can provide a certain degree of reference for other high schools to develop their CS courses.

2. Hierarchical CS Courses

In BNDS, the framework of CS courses has been refined in 2017 since the Standard was published by the Ministry of Education. The original technology courses are classified into two categories: the general technology courses and the CS courses. In the general technology courses, students learn various skills of engineering, e.g., mechanical technique, circuit design, architecture, and even costume design. In comparison, the CS courses concentrate on the knowledge and skills in computer science, e.g., programming, algorithms, data analysis, and computer system.

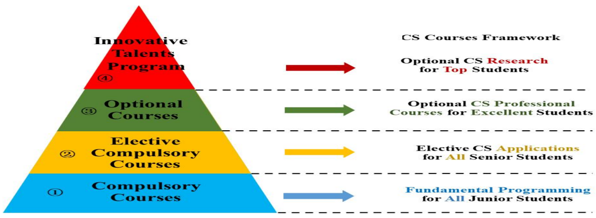


Figure 1. Hierarchical CS Course Framework

The framework of CS courses in BNDS is shown as a pyramid in Figure 1. From the bottom to the second top layer, the framework of CS courses includes compulsory courses, elective compulsory courses, optional courses as the Standard requires. The compulsory courses on the bottom layer are similar to the course of introduction to computing in college, the goal of which is to make students understand the basic principles of computer and program through the learning of a specific programming language like Python. Python is a programming language that is easy to get started. Python is a relatively high-level language, which is not as complex as C++ and Java in syntax. It is very close to the expression of natural language. Therefore, Python is suitable for beginners especially for high school students. Programmers can usually complete some simple work on Python after a short period of learning.

The compulsory courses are open for all junior high school students. On the second layer, when all students have the fundamental abilities using programming language and handling simple algorithms, the students can select two elective courses to do some applications in their senior high school career based on their own interests, such as robotics and big data. On the third layer, the optional courses are carried out for the students specific interested in CS, in which the student can learn deeper and more professional knowledge in CS, e.g., algorithms in Olympiad in Informatics (OI), machine learning methods, and mathematical modeling.

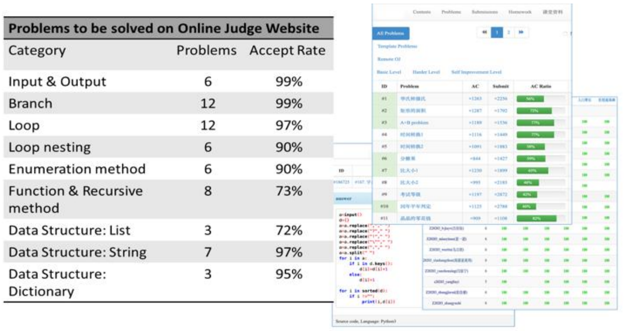
In addition, a number of talent students can accomplish the regular high school courses or even the undergraduate courses in college without difficulties. The regular courses can hardly satisfy these students' desires. Thus, besides the courses in the Standard, on the top of the Pyramid, we provide the Innovative Talents Program for the talent students with the assistance of universities and companies. The students in the Innovative Talents Program can do their own research under the guidance of many CS professionals in universities’ labs.

In the rest of this paper, we will provide more details of the courses on each layer. On each layer, the synopsis, some typical examples of teaching process, and the performance of students are given, respectively. Furthermore, we will introduce how we improve the framework during practice and provide a typical course project in the end of this paper.

3. Compulsory Courses

The compulsory courses aim to make students understand the basic principles of computer and program through learning a specific programming language like Python. The compulsory courses are open for all 250 students in the K-12 program between Grade 8 and Grade 9(Grade 9 is the first year of the senior high school in this program). The courses are based on school-made text books according to the Standard, which last for 4 semesters. In each semester, 2 × 45 minutes are taken per week. All the tasks are finished during the class so that there is no homework for students.

The compulsory courses can be divided into two stages. Each stage lasts for 2 semesters. In the first stage, the students learn the basic syntax and logic of Python, a number of basic algorithms and data structures in the first semester, including input/output, branch and loop, list, string, dictionary, enumeration method etc. in python. Some comprehensive algorithms such as recursion backtracking method are also offered to the students who are interested in CS. Besides the regular interpretation by the teachers in class, the teaching and learning process is supported by a customed Online Judge (OJ) platform. On the OJ platform, hundreds of questions such as *A + B Problem* (input a and b then print a + b) are designed by the CS teachers for the students to practice the basic logic and syntax of Python. Algorithms, and data structures can also be practiced using OJ. Students can upload their program to the OJ website, which will judge validity of the program and give students feedback in seconds. All the problems are designed carefully by the teachers according to the students' fundamentals in mathematics and science, and cognitive level.



Problems to be solved on Online Judge Website		
Category	Problems	Accept Rate
Input & Output	6	99%
Branch	12	99%
Loop	12	97%
Loop nesting	6	90%
Enumeration method	6	90%
Function & Recursive method	8	73%
Data Structure: List	3	72%
Data Structure: String	7	97%
Data Structure: Dictionary	3	95%

Figure 2. The OJ platform and the performance of students

Some typical questions on the OJ platform and the performance of students are given in Figure 2. According to the teaching practice experience, we find that OJ scoring and feedback mechanism can effectively stimulate students' interests in programming. At the same time, the OJ can help teachers to analyze the performance of students and improve the teaching and learning process. We find that more than 60 percent of students can apply the algorithms and data structures to solve new problems on OJ platform at the end of the semester. About 30 percent of students can make approximate imitations to solve the similar problems taught in class. Only a few students need to solve the same questions repeatedly to learn some basic concepts and the skills of programming. Finishing the first stage, students can learn how to program in Python and how the computer works and their responsibility in the information society.

Figure 3. The project of word statistics in the Romance of the Three Kingdoms

The stage 2 of the compulsory courses lasts for 2 semesters. In stage 2, the CS teachers design a series of projects to help students improve the capabilities of CS engineering. The problem in each project is usually generated from a problem in the daily life. Specifically, the CS teachers simplify the problems, highlight the concepts of CS and integrate the algorithm and data structure training in each project. An example is given in Figure 3. The students learn *The Romance of the Three Kingdoms* (RTK) in their Chinese courses in Grade 8. Thus, the CS teachers establish a project on the RTK, i.e., to find the most frequent names in RTK. To do the projects, the students need to learn the basic concepts of word segmentation and word frequency statistics in computational linguistics, and eventually use the file operations and dictionary structure to accomplish the program. More than 60 percent students can accomplish the task under the guidance of CS teacher, in which 30 percent excellent students can further revise the algorithm and apply this method to analyze other books. The projects in stage two can help students review the knowledge in stage one. Building their own programs to solve the real-life problems can also help them reach the deeper level understanding of knowledge.

Through compulsory courses, the students cultivate the fundamental abilities of programming and computational thinking, which help them to have a better understanding of the digital world and live a better life in the future. The students can apply the algorithms and data structures to solve simple problems with computers. Further, through a number of projects generated from the daily life, the students have already developed the fundamental CS engineering capabilities.

4. Elective Compulsory Courses

In order to meet the personalized needs of different students, we also design and carry out a variety of school-based elective compulsory courses, which correspond to one or more modules of the elective compulsory courses in the Standard. All students in BNDS must select at least two elective compulsory courses in Grade 10 or Grade 11. Each course lasts for 1 semester and costs 2 × 45 minutes per week. There is no homework for students.

Table 1. The scheme of elective compulsory courses

Course Name	Introduction
AI technology	Develop a number of AI programs
App Inventor	Develop APP for mobile devices
Database	Build a website by Python using database. Focus on backend
Web Design	Build a website by Python and web design. Focus on frontend
Big Data and Intelligent System	Train voice data with advanced machine learning model and test on Robot
Programming and Gaming	Develop a game and game AI by Python
AI & Robot	Build and program robot to accomplish given task
NLP	Use Python to process language data
Image processing	Process image by PS and Python

As shown in Table 1, the scheme of elective compulsory courses includes multiple CS directions: artificial intelligence (AI) technology, APP inventor, web design, database, big data and intelligent system, programming and gaming, AI$\&$robot, and image processing. In each course, the students accomplish a series of comprehensive projects in a specific realm of CS. For example, in the Programming and Gaming course, the students design the Gomuku AI and compete with each other. In the APP inventor course, the students design APPs to automatically recognize the followers and cars. In the AI & robot course, students train the robot to accomplish all kinds of tasks. In the AI technology course, the students learn AI algorithms and use Python modules to develop multiple applications. An example is shown in Figure 4, in the AI technology course, three students design an intelligent campus guide on WeChat for the visitors to BNDS, which is exhibited in the first International Conference on Artificial Intelligence and Education in 2019 (A report, 2019).

Figure 4. Student's project: an intelligent campus guide

Through elective compulsory courses, each student can develop abilities to write relatively complicated CS programs and deal with the comprehensive realistic problems in different CS directions. We believe that these abilities can help students accomplish various projects of their majors in college.

5. Optional Courses

The optional courses are open for the excellent students who are very interested in CS and want to learn deeper knowledge and skills in the CS-related fields. Thus, students in all grades can select the optional courses based on their interests and abilities. Meanwhile, the duration of each course and the class hours per week depend on the specific needs of each course. Some students may spend more than ten hours digging into a specific project, which is motivated by their interests. The scheme of the optional courses is shown in Table 2, which includes the mathematical modeling, geographic information system, MIT Sea Glide, automatic driving, first tech challenge, internet-of-things (IoT) system, advanced technologies in AI, and the OI course.

Table 2. The scheme of optional courses

Course Name	Introduction
Mathematical Modeling	Model real-life problem and solve it by program
Geographic Information System	Use program to monitor and model geographic information
MIT Sea Glide	Build and program an under-water robot
Automatic driving	Build and program a self-driving car
First Tech Challenge	Build and program a remote-control robot to complete all kinds of challenge
IoT System	Build all kinds of smart home tools using board and chips
Advanced Technologies of AI	Learn advance AI tech and use it to solve real life problem
Olympiad in Informatics	Learn algorithms and data structure for Olympiad in Informatics

The difficulty of each course is high and the contents are almost approximate to the corresponding courses in the universities. For example, in Mathematical Modeling course, many famous partial differential equations (PDE) models are introduced such as Arm Race Model. The students will model the realistic problems using the typical PDE models and solve them with computers. In the Advanced Technologies of AI course, the students can learn the core ideas of many machine learning algorithms such as supporting vector machine and neural network. The students in OI courses can learn complicated algorithms and data structures, and take part in the OI Competitions (Kolstad, 2007). In Figure 5, we provide the increasing number of first prize winners of BNDS in the National Olympiad in Informatics in Province (NOIP) and the Certified Software Professional-Senior (CSP-S) competitions, which are the typical provincial level programming contests in China. More than 20 students per year won the first prize since 2018. It is worth pointing out that one student won the gold medal in the National OI contest of China (NOI) in 2019 and two students won the same prize in 2021, one of which is the 8th place among all the competitors in China. Our students win at least one gold medal in each NOI since then.

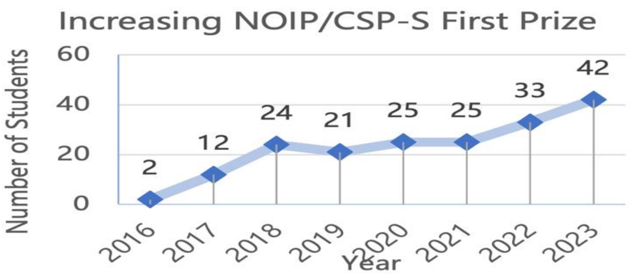


Figure 5. The increasing number of NOI/CSP-S first prize winners in BNDS

Even though each optional course is difficult to learn and most contents in the optional courses are not directly related to the National College Entrance Examination, the students in BNDS hold very high enthusiasm for participation. The standard class size is 12 in each course per semester. However, the applicants are usually far more than 12 for some optional courses. Thus, we believe that through proper education and guidance, more students themselves will pay more attention to learning knowledge and improving real abilities rather than test training.

6. Innovative Talents Program

A number of talented students in BNDS can easily accomplish the regular high school courses and even the foundation undergraduate courses. Meanwhile, they also have great interest and talent in one or more specific research field. Those students are expecting to enter the professional laboratories carrying out the scientific research under the guidance of professors in college.

In order to satisfy these students' expectation, BNDS associated with the China association for Science and Technology have established the Innovative Talents Programs to support these talented students. The Programs have been supported by Tsinghua University, Peking University, University of Chinese Academy of Sciences, Beihang University and more than 25 top universities. In each year, BNDS can recommend more than 40 talented students enter these universities to engage in the scientific research in the math, physics, chemistry, biology, computer science or engineering science. During the one-year training process, these students master the basic methods of literature search, scientific investigation and data processing.

Figure 6. A project developed by the student in the Innovative Talents Program

For example, a talented student in Grade 10 proposed a novel skin color model for the mobile image under the guidance of the computer science professor in Beihang University. As showed in Figure 6, compared to the skin-color model for solving the complex illumination and background, the model has the characteristics of pertinence and high computational efficiency based on the normalization method in the process of the combination with haircut and face. Based on this achievement, this student achieved the First Award of China Adolescents Science and Technology Innovation Contest, and Mayor's Award for Beijing Youth in Science and Technology.

In recent years, an increasing number of talented students haven been trained with the scientific research and practical skills through the Innovative Talents Program. These students show promise to become scientists in the fundamental and advanced subjects in the field of math, physics, chemistry, biology, computer science or engineering science.

7. Courses Iteration

It has been more than 6 years since the Standard was published and 5 years since we began to establish our new CS course framework. The teachers in BNDS keep revising the framework and the courses in it. With the courses reform, we paid more attentions improving the students' consciousness of solving practical problems with computers, i.e., information consciousness, to purely teaching the programming skills. The teachers brought more synthetic and cultural projects into the compulsory courses in recent years in comparison with the course beginning in 2018. For example, to teach the concept of Key-Value relation in Python, the teacher designed a "peer comment projects". In the project, the students need to build a comment program using Dictionary in Python that record the comments of their friends. This project can also guide students to learn how to do the proper comments on the Internet.

In addition, the teachers are trying to bring more and more real-life projects into courses and make the class more student-centric these years. The students can build their own recite app in their literature class. In math class they use python to modelling coronavirus propagation during the Covid-19 pandemic for instance.

As for the elective compulsory courses and optional courses. A number of courses which is less interested by students was replaced by new one. A robot building course transformed into an AI recognition course for example.

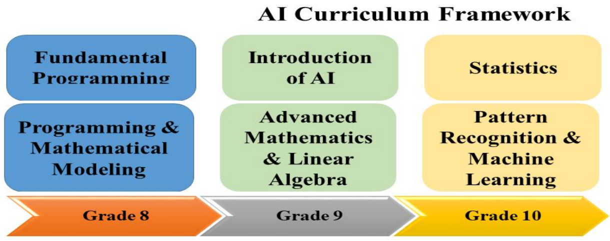


Figure 7. The 3-year AI Framework

Finally, since more and more students have already accepted Python programming training in their primary school careers. A number of students in Grade 8 and Grade 9 feel that the CS course content is too simple for them. A 3-year AI curriculum framework, as shown in Figure 7, which might be fully explained in other papers (Zijie, 2023). A part of the material has been accomplished as open source for reference.

8. Case Study

In this part, we will introduce a typical project we practiced. We call it *the Average Face of BNDS*. It is in the last semester of the compulsory courses when the students are at Grade 9. Students are required to solve a real-life problem, to calculate the average face of BNDS which can be used to help school's information platform check whether an ID photo is proper or not. The project contains the basic idea of encoding, image processing and classification algorithm, which will not only deepen students' understanding of how computer works but also lead to some other AI project in future.

Students can use web crawler or other techniques to get a bunch of images first, which will review the knowledge about Web, file, and the Internet. They use python to create the final work base on the images they got. The second part of this project will help the student get a deeper understanding of how images are stored in the computer system. The project supported by all the modules provided in Python such as CV2 and requests. One of students' code in the image processing part of the project is shown in Figure 8. The annotation is added afterwards for readers of this paper. Since the students have learned web crawler and CV2 in former projects, the coding can be done by themselves. They can also get structured guidance throughout the project on our course web site. Teachers work as an advisor.

```python
import cv2     #using CV2 module to load and calultation images
imgs=[]   #a list of image to be calulated, init as empty
img0=cv2.imread("0.jpg")      #load a random image at first
row,col,num=img0.shape   #record the size of the first image
for i in range(100):     #iterate all the images downloaded
    img=cv2.imread(str(i)+".jpg") #all the images' name are pretreated as numbers after download.
    img=cv2.resize(img,(col,row)) #resize each image to the same size of the first one
    imgs.append(img)   #add each image to the image list after resizing
img1=img0.copy()    #copy the first image to store the final result
for i in range(row):     #iterate each rows
    for j in range(col):     #iterate each cols
        for k in range(3):  #iterate B,G,R in each pixel
            tmp=0     #add all numbers of the same pixel from every images to tmp
            for t in imgs:      #iterate every images in image list
                tmp+=t[i][j][k]     #calulate sum
            tmp=tmp//len(imgs)  #calulate average number of certain color in certain pixel
            img1[i][j][k]=tmp   #change certain color in certain pixel of img1
cv2.imshow("pic",img1)    #display img1 in screen
cv2.imwrite("Average cat face(ORANGE).jpg",img1)  #write img1 to file
```

Figure 8. CV code of the project

Most of students can get a picture of the average face of BNDS as showed in Figure 9. A number of students want to create a personalized picture which is different from others. Some students create the average face of their favorite cats, others may focus on basketball teams. Some students use the same procedure to create an art image in Ninja NARUTO as Figure **10**. They download hundreds of different images using web crawler. Then they merge all the pictures into one.

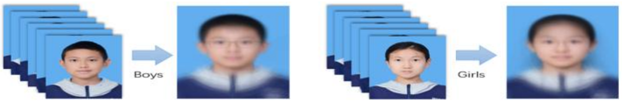


Figure 9. Average Face Demonstration

Figure 10. Image Fusion Demonstration

Students can get art pictures of their own in this project, review their computer science knowledge, achieve a higher programming ability and get some basic idea of classification method. 20 percent of the students can migrate the skills they learned in this project and use them to solve similar problems. The rest of the students can finish the project with teachers' help.

9. Conclusions

In this paper, we introduced the hierarchical framework of CS courses and the teaching experience in BNDS. The framework includes the compulsory courses, the elective compulsory courses, and the optional courses according to the Standard, and the Innovative Talents Program based on the student cognitive situations in BNDS The students can develop the fundamental competence in computational thinking and CS engineering through compulsory and elective compulsory courses. Through practicing, we verified that with individualized education and guidance by schools and teachers, more students themselves can pay more attention to learning knowledge and improving real abilities in CS rather than test training. We also verified that the high school students have abilities to accomplish the entire curriculum in the Standard. We hope that our framework and teaching experience can provide a certain reference for other high schools to develop their own CS courses, especially for the high schools in the developing countries.

10. REFERENCES

A report of UN's AI conference in China http://m.haiwainet.cn/middle/3543599/2019/0524/content_31562878_1.html People's Daily, May, 2019.

AI course material of BNDS https://www.bilibili.com/video/BV12R4y1F7UQ?share_source=copy_web&vd_source=ca9ee67f2189c612bd2ca8755b7037a3

BNDS Online Judge Platform https://onlinejudge.bnds.cn

Bradski G. The Open CV Library[J]. Dr Dobbs journal of software tools, 2000.

D. Sun and Y. Li, "The Development Status, Research Hotspots and Enlightenment of Youth Programming Education in China and Foreign Countries: Also Discuss the Implementation Strategy of Programming Education for China in the Age of Intelligence," Journal of Distance Education, Vol. 3, pp. 47-50, Mar. 2019.

"Education Modernization 2035," State Council of The People's Republic of China, 2019.

"Every Student Succeeds Act," Department of Education of Unite States, 2015

"Gaokao Reform in Zhejiang Province," The Peoples' Government of Zhejiang Province, 2014.

Kolstad, Rob, and Don Piele. "USA computing Olympiad(USACO)." Olympiads in Informatics 1 (2007): 105-111.

"K-12 Computer Science Standards," Computer Science Teachers Association, 2011.

"K-12 Computer Science Standards," Computer Science Teachers Association, 2016.

"Learning Guidance Essentials," Ministry of Education, Culture, Sports, Science and Technology-Japan, 2018.

National Olympiad in Informatic of China https://www.noi.cn/gynoi/jj

"National Curriculum England: Computing Programmes of Study," Department of Education, 2013.

"National Curriculum of Digital Technology," Australian Curriculum Review Consultation, 2016.

"National Curriculum Standard," Ministry of Education of The People's Republic of China, 2017.

"National Curriculum Standard," Ministry of Education of The People's Republic of China, 2020.

Requests module of Python http://docs.python-requests.org

The International Olympiad in Informatics https://ioinformatics.org/

Webb, Norman L. "Depth-of-knowledge levels for four content areas." Language Arts 28. March 2002.

Zijie Zheng, Siyao Han, Weiqi Zhang, Xingming Wang, and Chao Gong,"Why and How to Teach Machine Learning in High School –A Practice in Bejing National Day School in China", ACM TURC '23: Proceedings of the ACM Turing Award Celebration Conference - China 2023 July 2023 Pages 89–91.

Block-based versus Text-based Programming: A Comparison of Learners' Programming Behaviors, and Computational Thinking Skills toward Programming

Zhanyan QI[1], Yutong FU[2], Yicheng QIAN[3], Liwen SHI[4], Dan SUN[5] and Cheekit LOOI[6]
[1,2,3]Jing Hengyi School of Education,Hangzhou Normal University,China
2023211003003@stu.hznu.edu.cn, 2023211003039@hznu.edu.cn, 2023211003014@hznu.edu.cn

ABSTRACT

In an era where computational literacy is paramount, many global schools are stressing the importance of K-12 programming education. This field is predominantly composed of two separate modalities – block-based programming modality and text-based programming modality. Previous research may not have provided a complete understanding of the differences between these two modalities as it did not take into account both the learning process and learning outcomes. This study aimed to compare secondary students' programming behaviors and computational thinking skills between two modalities through a quasi-experimental design in a Chinese secondary school. The findings showed that (1) learners in TPM encountered more syntactical errors and spent more time between two clicks of debugging, while learners in BPM had more code-changing behaviors by adjusting programming blocks, made more attempts of debugging, and had more irrelevant behaviors; (2) learners in BPM achieved a higher level of computational thinking skills; (3) Code Changer, Minimal Debugger, Maximal Debugger, Distracted Coder and Average Coder were identified through students' programming behavior in two programming modalities, and differences in their CT skills and attitudinal data were revealed. Lastly, pedagogical implications based on the findings are also discussed.

KEYWORDS

Computational thinking, Text-based and block-based programming modality, Programming behaviors

1. INTRODUCTION

Computational thinking (CT) is a vital 21st-century competency, integral to K-12 student education (Wing, 2014). Nations like China, the US, and the UK have integrated computer programming into their K-12 curricula, primarily employing two core methodologies: block-based programming modality (BPM) and text-based programming modality (TPM) (Jocius et al., 2021). BPM utilizes visual aids, making it more approachable to beginners, whereas TPM involves traditional coding in text-based programming languages, equipping learners for professional programming tasks (Weintrop & Wilensky, 2019).

Although past studies have evaluated differences between TPM and BPM regarding student learning and the necessary skillset for each modality, there is a lack of research examining how the learning process influences performance within the context of TPM and BPM (Weintrop & Wilensky, 2017). This gap underscores the need for an in-depth assessment of the divergence between TPM and BPM, considering both the learning process and the outcome. Given the prominence of CT and programming in K-12 curricula, it's essential to comprehend each modality's respective characteristic and how they can efficiently bolster student learning (Grover, 2021).

Our research aimed to bridge this gap by contrasting the effectiveness of TPM and BPM in a Chinese secondary school through a quasi-experimental design. We scrutinized students' programming behaviors and CT skills across both modalities to evaluate which encourages programming practice. The insights garnered from our study can guide educators in creating more effective instructional methodologies to foster students' programming skills and CT development.

2. LITERATURE REVIEW

2.1. Block-based and Text-based Programming Modalities

Turing Award recipient Dijkstra once underscored the profound impact our tools have in shaping our cognitive capacities. This principle resonates with the significant role that programming styles play in altering students' thought processes, a crucial consideration for educators in computer programming. Block-based and Text-based programming, BPM and TPM respectively, are extensively utilized i K-12 education. BPM, designed with a unique "programming-primitive-as-puzzle-piece metaphor," visually represents codes to simplify usage and prevent syntax mistakes, making computer science more approachable for novice learners (Bau et al., 2017). Multiple BPM platforms like Scratch, Blockly, and Alice are built to reduce the learning curve by addressing syntax complexities (Grover, 2021). Given these advantages, BPM is a preferred choice in introductory computer science curriculums nationwide.

Compared to BPM, learners perceive TPM as more authentic and empowering in their programming journey. It offers advanced learning opportunities and participation in professional work that can't be fulfilled by BPM alone. Nevertheless, as students progress, they often grapple with syntax errors and frustration, leading to higher dropouts in TPM courses. Hence, various teaching strategies are deployed to alleviate these difficulties, to support text-based programming, and to boost students' interest and motivation towards TPM (Sun et al., 2021).

2.2. Computational Thinking and Programming

CT encompasses analytical thinking common to areas such as mathematics and engineering and can be fostered through diverse means such as computer science games, language learning techniques, and STEM activities (ISTE, 2015; Wing, 2014). Programming, among other strategies, is known to effectively improve learners' CT skills. The concept of CT has been divided into domain-specific and domain-general definitions. The former relates to knowledge and skills for solving specialized problems in computer science or programming; tools like Dr. Scratch or the Bebras test are used for evaluation. The latter views CT as competencies for solving everyday life problems across all learning domains, constituting creative, algorithmic, critical thinking, problem-solving, and collaboration skills (Tsai et al., 2021). Higher-level metacognitive skills regulate CT, and it's seen as algorithmic problem-solving while promoting team communication and collaboration. Its main elements include algorithmic thinking, problem-solving, creativity, critical thinking, and cooperativity. A CT scale was developed to measure students' proficiency in these elements (Román-González et al., 2017). Previous studies regarding programming's effectiveness typically assessed learners' knowledge mastery, with fewer focusing on higher-order thinking. Preliminary programming teaching often starts with console-based languages for learners to comprehend basic concepts without the added complexity of a visual interface.

Consequently, the understanding of differences between BPM and TPM is a research interest, particularly in benefiting secondary students' programming learning and fostering interest in computer science. This research will gather multi-modal data to analyze students' programming behaviors and CT skills across these modalities.

3. METHODOLOGY

3.1. Research Purpose and Questions

This study aimed to understand how TPM and BPM impact learners' CT skills. Carried out in a secondary school, it employed a quasi-experimental design and collected data encompassing learning logs and CT skills test. Analytical methods such as statistical and cluster analysis were used to compare behaviors and proficiencies in TPM and BPM. The results will inform future pedagogical strategies and research in computer programming. The specific research questions are: RQ1: What were the differences in learners' behaviors in learning via TPM versus BPM? RO2: What were the differences in learners' CT skills in learning via TPM versus BPM?

3.2. Educational Context

The research was conducted in a compulsory course titled "Information Technology," which was carried out in a Chinese secondary school during the autumn of 2020. A quasi-experimental design was utilized to investigate the differences in learners' programming behaviors and CT skills toward programming in TPM and BPM. There were 32 learners in the TPM class and 32 learners in the BPM class. Students were around 13, and most of them did not have programming experience in formal education. Classes were taught by the same instructor, who maintained a similar teaching style under two modalities, offered the same instructional materials to learners, and used the same teaching guidance for each class, except the materials were presented via TPM or BPM.

The course encompassed three phases and six instructional sessions, each taking 45 minutes. Phase I introduced programming basics and either TPM or BPM. Phase II involved practice in programming with sequential, selective, looping, and function structures in the designated programming modality. Phase III was a series of projects that tested knowledge from Phase II. Concepts, algorithms, and coding were taught contextually in the TPM or BPM class. The platform used was Code4all (see Fig. 1), a Pencil Code-developed environment that supports both TPM and BPM. However, unlike Pencil Code, Code4all restricts users to either a block-based or text-based interface, without switching between them. Thus, BPM class learners used a drag-and-drop mechanism, while the TPM class typed commands one character at a time. The platforms were identical apart from their modality, including language, visual execution environment, and environmental scaffolds. The underlying language was CoffeeScript, known for its syntactically light nature and active user base (Weintrop & Wilensky, 2019).

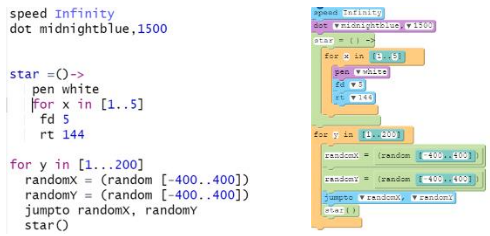


Figure 1. The Code4all programming platform and two programming interfaces used in this study.

3.3. Date Collection

To capture learners' programming learning performance, we collected multi-modal data, including platform logs, students' CT skills, and attitudinal data. Secondly, we collected data about learners' CT skills before and after the intervention, and learners in TPM and BPM took the same test. The test instrument for CT skills was adapted from the CT scale (CTS) developed by Korkmaz et al. (2017). This 5-point Likert scale contains five factors (creativity, algorithm thinking, cooperativity, critical thinking and problem solving) and 22 test items. Notably, the CTS was validated and applied among Chinese K-12 learners by Bai & Gu (2020).

3.4. Date Analysis

We used multiple analytical approaches, such as statistical and clustering analysis, to examine the impact of learners' programming learning quality between TPM and BPM. Firstly, according to extracted variables and previous studies (Pereira et al., 2020), this study identified five programming behaviors: Average number of code-changes (AnC), Number of Irrelevant behaviors (NoIB), Number of debugs (NoD), Average time between two debugs (AtD), and Number of errors (NoE). It should be noted that, despite the learner's coding in different programming modalities, the codes recorded by the Code4all platform were in a text-based format. Descriptive statistics were used to provide an overall view of distribution of programming behaviors between TPM and BPM. To uncover hidden patterns in the complex dataset, we used clustering algorithms. Noting the heterogeneity in learners' behaviors, we utilized their logs to cluster them and analyze their programming behaviors within each cluster. We employed the popular k-means algorithm which involves selecting the optimal number of clusters, variable selection, data standardization, and initial cluster center selection. Repeat iterations of assigning points to the closest center and recalculating the centers ensued until convergence was reached. The cluster characteristics were evaluated and interpreted per the research question and hypotheses, using the mean silhouette coefficient to determine the ideal number of clusters. RapidMiner and Python were used for this clustering analysis (Rousseeuw, 1987).

Secondly, we conducted a T-test to examine the difference in CT skills between the TPM and BPM groups. Furthermore, we utilized Analysis of Variance (ANOVA) to explore the difference in learners' CT skills among different clusters.

4. RESULTS

4.1. Learners' Programming Behaviors in Two Modalities

The results in Table 1 showed that some obvious differences in learners' programming behaviors between the two modalities. In particular, learners in TPM encountered more errors (NoE: M = 8.81, SD = 20.17) than those in BPM (NoE: M = 4.97, SD = 5.31). Learners' average time spent between two clicks of debug button in TPM (AtD: M = 84.74, SD = 83.35) was almost twice as long as that in BPM (AtD: M = 46.97, SD = 32.19). In addition, learners' average amount of code changing in BPM (AnC: M= 25.80, SD = 15.93) was slightly higher than that in TPM (AnC: M= 24.81, SD = 39.33). The number of clicks on debug button made by learners in BPM (NoD: M = 37.73, SD = 20.25) was higher than that in TPM (NoD: M = 32.03, SD = 43.06). Learners in BPM (NoIB: M = 35.20, SD = 24.06) had more irrelevant behaviors than those in TPM (NoIB: M = 25.53, SD = 17.26).

Table 1. Mean and standard deviation of the programming behaviors for TPM and BPM

M	N	AnC	NoIB	NoD	AtD	NoE
TPM	32	24.81 (39.33)	25.53 (17.26)	32.03 (43.06)	84.74 (83.35)	8.81 (20.17)
BPM	32	25.80 (15.93)	35.20 (24.06)	37.73 (20.25)	46.97 (32.19)	4.97 (5.31)

To further explore the differences in programming behaviors between the two modalities, learners' programming behaviors were modeled by using the features presented in Table 2. Previous studies found that it was possible to draw patterns using fine-grained data from one programming course (Estey & Coady, 2016). We inspected the k-means clusters, and the convergence of k-means was achieved in the 10th iteration with k = 5 as the best value with the highest value of mean silhouette coefficient (0.57). 23.44% of the learners were assigned to Cluster 1, 20.31% to Cluster 2, 3.12% to Cluster 3, 6.25% to Cluster 4 and 46.88% to Cluster 5. Fig. 4 depicts the programming profile of learners for each cluster in two modalities.

Comparing the features among five clusters, Cluster 1 was identified as the Code Changer, they had the second highest frequency of code changing (AnC: M = 44.53, SD = 10.69) and the number of clicks on debugs (NoD: M = 60.73, SD = 11.28), the second lowest number of irrelevant behaviors (NoIB: M = 23.73, SD = 12.09), and the moderate frequency of the number of errors (NoE: M = 10.67, SD = 7.89) amongst the five clusters. Based on the clustering analysis, we found that five students from TPM and ten students from BPM were assigned to this cluster. Cluster 2 was identified as the Minimal Debugger where students had the longest time interval between two clicks of debug (AtD: M = 174.43, SD = 72.95), and the lowest frequency of code changing (AnC: M = 4.00, SD = 2.45), number of clicks on debugs (NoD: M = 7.69, SD = 3.73), and the number of errors (NoE: M = 0.38, SD = 0.62). According to the cluster analysis results, it was determined that this particular cluster comprised ten students from TPM and three students from BPM. Two learners from TPM were clustered into Cluster 3 which was identified as the Maximal Debugger. Here, they had the highest frequency in code changing (AnC: M =162.00, SD = 35.00) and the numbers of debugs (NoD: M =179.50, SD = 40.50), and they encountered the highest number of errors (NoE: M =82.00, SD = 14.00), and had the lowest frequency in irrelevant behaviors (NoIB: M =14.78, SD = 10.78) among five clusters. Cluster 4, which was identified as the Distracted Coder, had the highest frequency of irrelevant behaviors (NoIB: M = 84.25, SD = 11.09), the second lowest frequency of the number of debugs (NoD: M = 10.00, SD = 11.47), and the moderate behavior of the number of errors (NoE: M = 11.75, SD = 3.47). This cluster consisted of two students from TPM and two students from BPM. Cluster 5, which was identified as the Average Coder, had the average frequency of code changing, number of debugs, number of errors, irrelevant behaviors, and the time interval between two clicks of debugs among the five clusters.

4.2. Learners' CT Skills in Two Modalities

As to RQ 2, regarding learners' CT skills, there was no significant difference (creativity: p = 0.286; algorithm thinking: p = 0.185; cooperativity: p = 0.217; critical thinking: p = 0.067; problem solving: p = 0.095; overall CT skill: p = 0.941) before the intervention. The results revealed that, after the intervention, statistically significant differences were identified for algorithm thinking (t = 3.23, p = 0.002), cooperativity (t = -2.11, p = 0.038), problem solving (t = -2.72, p = 0.008) and overall CT skills (t = -2.58, p = 0.012). In terms of algorithm thinking, cooperativity, problem solving and overall CT skills, learners in the BPM group outperformed those in the TPM group (Table 2).

Table 2. Statistical summary of learners' CT skills in the two modalities

CT	Group	*M*	*SD*	*t*	*p*
Creativity	TPM	3.83	0.83	-1.65	0.103
	BPM	4.16	0.60		
Algorithm	TPM	3.57	0.87	-3.23**	0.002

thinking	BPM	4.24	0.66		
Cooperativity	TPM	3.85	1.07	-2.11^{*}	0.038
	BPM	4.35	0.68		
Critical thinking	TPM	3.78	0.94	-1.05	0.296
	BPM	4.00	0.66		
Problem solving	TPM	3.94	0.67	-2.72^{**}	0.008
	BPM	4.39	0.63		
Overall	TPM	3.79	0.75	-2.58^{*}	0.012
	BPM	4.23	0.53		

Note. $^{*}p < .05$; $^{**}p < .01$.

5. DISCUSSION

5.1. Addressing Research Questions

With the more pervasive development of computer programming education, there is increasing research on how learning occurs under the two typical programming modalities (BPM and TPM). This study collected multi-modal data to analyze students' programming behaviors and CT skills in two programming modalities. Learners' CT skills under two modalities were compared through pre-/post-test.

For research question 1, this research revealed that learners in TPM tended to spend more time between two clicks of debug button and encountered more syntactical errors. Students in BPM spent more time on code changing (operating blocks and adjusting parameters), made more attempts at debugging, and had more irrelevant behaviors. The research revealed five clusters based on students' programming behaviors, including Code Changer (C1), Minimal Debugger (C2), Maximal Debugger (C3), Distracted Coder (C4), and Average Coder (C5).

Post-test comparison of students' CT skills in BPM and TPM showed BPM learners excelling in algorithmic thinking, cooperation, and overall CT abilities. The block-based format of BPM provides an intuitive, concrete coding experience, aiding the comprehension of basic programming concepts and the development of algorithmic thinking. BPM's iterative development system allows learners to see immediate results and provides collaboration tools enhancing CT essentials, like communication, cooperation, and social skills (Grover, 2021).

Additionally, in TPM, frequent coding and debugging is associated with higher problem-solving performance, likely due to the expressive nature of languages like Python and their flexibility in designing tailored solutions (Kölling et al., 2015). Conversely, block-based programming languages, such as Scratch or Blockly, rely on pre-built code blocks and may limit the complexity and abstraction that can be achieved (Weintrop & Wilensky, 2019). Active coding and debugging is crucial for developing programming concepts and problem-solving skills, students can develop a deeper understanding of programming concepts and become more proficient at identifying and resolving problems.

6. CONCLUSION AND FUTURE DIRECTIONS

Recognizing specific differences in programming modalities is key to advancing computer programming education and supporting computational literacy. This quasi-experimental study at a secondary school compared BPM and TPM, looking at learners' behavior and CT skills, yielding variation between the methods. Nonetheless, this research has three limitations. First, the study's context and potential unrepresentativeness of the sample necessitate careful interpretation and replication with larger, diverse samples. Second, relying on survey and behavioral data, future studies may benefit from performance-based tests or observations for deeper insight into CT skills. They could also utilize mixed methods, including qualitative interviews to uncover learners' experiences and BMP and TPM's synergistic potential. Third, aligning with multimodal learning analytics trends, our data, from log files and surveys, could be supplemented in future research with various data sources to delve into learners' behavior, cognition, metacognition, and social activities during programming learning.

In essence, instilling CT in young learners to prepare them for future schooling and careers is critical. This study, exploring differences in block-based versus text-based programming learning, provides valuable insights for effective K-12 programming education and relevant CT skill development.

7. REFERENCES

Bai, X. M., & Gu, X. Q. (2019). Research on the construction and application of the computational thinking Instrument in K12. China Educational Technology, (10), 83-90.

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the ACM, 60*, 72–80.

Grover, S. (2021). Teaching and assessing for transfer from block-to-text programming in middle school computer science. In: Hohensee, C., Lobato, J. (Eds.), *Transfer of learning. Research in mathematics education*. Springer, Cham.

ISTE. (2015). *CT leadership toolkit.*

Jocius, R., O'Byrne, W. I., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2021). Infusing computational thinking into STEM teaching: From professional development to classroom practice. *Educational Technology & Society, 24*(4), 166-179.

Kölling, M., Brown, N., & Altadmri, A. (2015). *Frame-based editing: Easing the transition from blocks to text-based programming.* In Judith, G. E., Sue, S., & Jan, V. (Eds.), In *WiPSCE '15: Proceedings of the workshop in primary and secondary computing education* (pp. 29-38). ACM.

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.

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies, 23*(4), 1483-1500.

Pereira, F. D., Oliveira, E. H. T., Oliveira, D. B. F., Cristea, A. I., Carvalho, L. S. G., Fonseca, S. C., Toda, A., & Isotani, S. (2020). Using learning analytics in the Amazonas: Understanding students' behaviour in introductory programming. *British Journal of Educational Technology, 51*(4), 955–972.

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.

Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics, 20*, 53–65.

Sun, D., Ouyang, F., Li, Y., & Chen, H. (2021). Three contrasting pairs' collaborative programming processes in China's secondary education. *Journal of Educational Computing Research, 1*(8), 54.

Tsai, M.J., Liang, J.C., & Hsu, C.-Y. (2021). The computational thinking scale for computer literacy education. *Journal of* Educational *Computing Research, 59*(4), 579–602. 208). ACM.

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education, 18*(1), 1-25.

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education, 142*(103646), 1-17.

Wing, J. (2014). Computational thinking benefits society. *Social issues in computing website*.

Research on the Current Situation of Computational Thinking of Junior High School Students Based on Curriculum Standards[1]

Yu Jia[1] , Jihong Xu[2*] , Yidi Liu[3*] , Shuang Han[4], Luyao Zhao[5]

[1234]Inner Mongolia Normal University, China

2205275056@qq.com 115202177@qq.com 1725672758@qq.com 1362117995@qq.com 2536435482@qq.com

ABSTRACT

With the rapid development of information technology, computational thinking has become one of the core abilities necessary for talents in the 21st century. With the promulgation of the Information Technology Curriculum Standards for ordinary High Schools (2017 edition) and the Information Technology Curriculum Standards for Compulsory Education (2022 edition), computational thinking has attracted more and more attention in school education. Based on the analysis of curriculum standards, this paper studies the current situation of computational thinking of junior middle school students in China through the evaluation tool of IT curriculum standards developed by a research team. It is found that there are significant differences in the level of computational thinking in different grades of junior high school students, with higher level of computational thinking modeling and lower level of automation and systematization. On the basis of this, this paper puts forward some training suggestions in order to provide reference for the development of computational thinking education for junior middle school students in our country.

KEYWORDS

Curriculum standards; Computational thinking; Middle school students; Development status

1. INTRODUCTION

As a new way of thinking, the importance of computational thinking is increasingly recognized. Computational thinking was first proposed in 1996 by Professor Seymour Paper of MIT, What is more recognized is that Professor Jeannette M.Wing of Carnegie Mellon University in the United States published a study on Computational thinking in Communications of the ACM magazine in 2006 Definition of Thinking: Computational thinking is a series of thinking activities covering the breadth of computer science, such as problem solving, system design, and human behavior understanding using the basic concepts of computer science . (WING JM, 2006)

In our country, with the development of science and technology and the reform of education, computational thinking has become the important content of middle school education. *Information Technology Curriculum Standards for Senior High Schools (2017 edition) clearly points out that information awareness, computational thinking, digital learning and innovation, and information social responsibility are the four core qualities of information technology curriculum, highlighting the importance of computational thinking. (Cao Xiaoming & Anna, 2018)* Computational thinking has become an essential skill for individuals to succeed in a complex technological culture. (Fu Qian, Xie Bochao & Zheng Yafeng, 2019) With the promulgation of the"Compulsory Education Information Technology Curriculum Standards (2022 edition)", computational thinking is sinking from the high school stage to the primary and secondary school stage. It clearly distinguishes the different requirements for the development of computational thinking of students in high school and compulsory education, and does not require them to learn and use programming language in middle and high school, while "mastering the basic knowledge of algorithm and programming" is included in the course content (goal).

On the contrary, in the continuous process of the gradual development of computational thinking, the problem-solving exercises completed by students in junior high school are indispensable accumulation to support the development of computational thinking to a higher level, and directly support the learning of program design and algorithms in senior high school. Therefore, the investigation of the development status of junior high school students' computational thinking will help us to have a clearer understanding of current students' computational thinking ability, and find problems from it, so as to make adjustments to the further training of students' computational thinking and achieve the best results. Therefore, this paper investigates the current situation of the development of junior high school students' computational thinking, aiming to find out the existing problems in the development of junior high school students' computational thinking, and provide reference for the research on the cultivation of junior high school students' computational thinking ability.

2. RESEARCH BASIS

2.1. The Concept of Computational Thinking

Since ancient times,computational thinking has been a necessary quality of thinking for the development of social production. (Xie Yueguang, Yang Xin & Fu Haidong, 2017) But the concept of computational thinking is understood differently by different people.As shown in Table 1, it is part of the definition of computational thinking organized by the author.

Table 1. A Description of the Definition of Computational Thinking.

[1] 【Fund Projects】Research on Computational Thinking Assessment Tools Oriented Toward Core Competencies (S20231081Z)

Source	Definition
Jeannette M.Wing, Carnegie Mellon University,USA (2006)	Computational thinking is a series of thinking activities covering the breadth of computer science, such as problem solving,system design, and human behavior understanding using the basic concepts of computer science. (WING JM, 2006)
International Association for Educational Technology(ISTE)and Computer Science Teachers Association(CSTE) (2011)	Computational thinking is a problem solving process, involving the elaboration of problems, the organization, analysis and presentation of data, the formulation, identification, analysis and implementation of solutions, and the migration of problem solving processes. (Cao Xiaoming & Anna, 2018)
Ren Youqun ect.(2016)	*Computational thinking is a unique problem-solving process that can help people better understand and analyze complex problems, so as to form problem solutions with formal, modular, automated, systematic and other computational characteristics. (Ren Youqun, Sui Fengwei & Li Feng, 2016)*
Information Technology Curriculum Standards for Senior High Schools (2017)	*Computational thinking refers to a series of thinking activities generated by an individual in the process of forming a solution to a problem by applying the thought methods in the field of computer science. (Ministry of Education of the People's Republic of China , 2018)*
Curriculum Standards for Information Technology in Compulsory Education (2022)	*Computational thinking refers to thinking activities such as abstraction, decomposition, modeling and algorithm design involved in the process of problem solving by individuals using thought methods in the field of computer science. (Ministry of Education of the People's Republic of China, 2022)*

Based on the above views, this study adopts the description of computational thinking in Ren Youqun and curriculum standards, and defines computational thinking as a series of thinking activities that abstract, decompose, model and design algorithms to solve problems in the process of problem solving, so as to help people better understand and analyze complex problems and form solutions.

2.2. The Literature Development Trend of Computational Thinking Research.

In this paper, the literature of CNKI is used as the source of literature search, with "computational thinking" as the keyword, the search was limited to core journals and the past ten years, and a total of 266 literature records were retrieved so far. In terms of the overall number, there is still a lack of research on computational thinking. Figure 1 shows the quantitative development trend of computational thinking research literature since 2014.

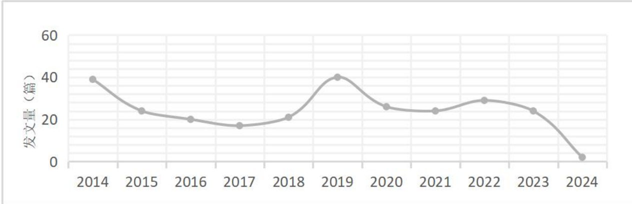


Figure 1. Computational Thinking Literature Publication Trend Chart.

As can be seen from Figure 1, the annual trend of literature publication shows an overall upward trend, indicating that computational thinking research has received more and more attention and attention at present, and this upward trend will continue in the future.

3. RESEARCH METHODS AND TOOLS

3.1. Research Methods

Questionnaire survey method, also known as questionnaire method, is a survey method in which the investigator uses a unified designed questionnaire to learn about the situation or solicit opinions from the selected survey objects. Questionnaire survey is a kind of research method to collect information by asking questions in writing. In this study, a computational thinking assessment tool for middle school students based on curriculum standards developed by the research team was adopted. The questionnaire contains a total of 15 questions, which are divided into five dimensions of modularization, formalization, modeling, automation and systematization according to the curriculum standard. There are 3 questions under each level of index, among which the questions are designed as easy and difficult according to the difficulty level of 25%, 50% and 75% of junior high school students' ability. Four scores were assigned from high to low to reflect students' level of computational thinking after statistical analysis.

3.2. Trial Test

The research team tested the evaluation tool in one grade 7 and one grade 8 (a total of 75 students) of a middle school. The evaluation tool of the test consisted of 15 questions, which were set to be submitted within 35 minutes, and 64 questionnaires were finally collected. The test results show that most of the students completed and submitted the test questions within the prescribed time, indicating that the set of questions is more scientific.

3.2.1. Reliability Analysis

The reliability of assessment tools is analyzed to verify the reliability and stability of test results. Klonbach α value is used to verify the reliability. Its specific value range and corresponding evaluation results are shown in Table 2 below.

Table 2. Klonbach α Value Range.

Value Range	Evaluation Result
above 0.9	The reliability is excellent
between 0.8 and 0.9	Good reliability, no need to delete any questions

between 0.7 and 0.8	Acceptable
below 0.7	Unacceptable

As shown in Table 3, the α value of the computational thinking evaluation tool in this study is 0.815, indicating that the reliability of the test paper is good.

Table 3. Reliability Analysis.

Cronbach's Alpha	Item
0.815	15

3.2.2 Validity Analysis

Structural validity is a reflection of whether the experiment can really measure the constructed theory. In this study,it is used to explain whether the setting of each item is reasonable, and the test questions are tested through confirmatory factor analysis. As shown in Table 4, the KMO value is 0.627, exceeding 0.5, and the significance is less than 0.05, indicating that the data is suitable for factor analysis.

Table 4. Validity Analysis.

KMO		0.627
Bartlett Test of Sphericity	Approximate Chi-square	209.716
	df	105
	P	.000

4. ANALYSIS OF THE DEVELOPMENT STATUS OF JUNIOR HIGH SCHOOL STUDENTS' COMPUTATIONAL THINKING

4.1. Basic information of survey objects

This study focused on sending questionnaires to grade 7 and grade 8 students in three regions. Among them, there were 152 in grade 7, accounting for 49.5%, and 155 in grade 8, accounting for 50.5%; There were 147 boys, accounting for 47.9%, and 160 girls, accounting for 52. 1%; There were 194 people in area A, accounting for 63.2%, 100 people in area B,accounting for 32.6%, 5 people in area C, accounting for 1.6%, and 8 people in other areas, accounting for 2.6%.

4.2. Correlation analysis

The correlation analysis of the five index dimensions of the evaluation tool of computational thinking is carried out, and the results are shown in Table 5.

Table 5. Correlation Analysis between the Dimensions of Computational Thinking of Junior High School Students.

Dimension	1	2	3	4	5
Modularization					
Formalization	.238**				
Modeling	.343**	.316**			
Automation	.113*	.187**	.151**		
Systematization	.181**	.296**	.309**	.248**	

Note: **p<0.01; 1:modularization; 2:formalization; 3:modeling; 4: automation; 5:systematization.

As can be seen from Table 5, there is a positive correlation between the five dimensions.In computational thinking, modularization allows the overall problem to be broken down into smaller, more manageable parts, while formalization allows these parts to be more normalized and precise, leading to better understanding and solving of the problem. Modeling makes problems simpler and clearer, thus providing the basis for automation, which in turn makes the modeled solution more efficient and reliable, thus improving the efficiency and quality of problem solving. Systematization is interrelated with other dimensions, modularization, formalization, modeling, and automation are all in order to better achieve the goal of systematization, that is, to understand and solve problem from a holistic perspective. Therefore, in the teaching process, teachers should pay attention to the connection and integration between these dimensions, so as to better cultivate students' computational thinking ability and creativity.

4.3. Difference analysis

4.3.1. Gender difference analysis

The author conducted an independent sample T test on the gender of the research object. As shown in the Table 6, In the Levin variance equality test, P=0. 155 >0.05, indicating that the score variance of students of different genders is equal, and the significance P=0.299>0.05, indicating that the score difference of students of different genders is not statistically significant.

Table 6. Sex-independent Sample T Test.

		Levin's test for variance equality		Mean equivalence t test				
		F	Sig.	t	df	Sig. (2-t)	Mean Difference	Std. Error Difference
AVG	Equivariance assumed	2.028	0.155	1.04	305	0.299	0.04779	0.04595
	Equivariance not assumed			1.035	292.713	0.302	0.04779	0.04618

4.3.2. Grade difference analysis

Independent sample T-test was conducted for grades. As shown in the Table 7, P=0.003 in the Levin variance equality test indicates that the score variance of students of different genders is not equal, and the significance P indicates that the score difference of students of different grades has significant statistical significance.

Table 7. Grade Independent Sample T Test.

	Levin's test for	Mean equivalence t test

		variance equality						
		F	Sig.	t	df	Sig. (2-t)	Mean Difference	Std. Error Difference
AVE	Equivariance assumed	8.693	0.003	4.344	305	0	0.19387	0.04463
	Equivariance not assumed			4.354	291.22	0	0.19387	0.04453

4.3.3. Regional difference analysis

The variance analysis of regions was performed, as shown in the Table 8 and the significance was 0.417>0.05, indicating that there was no significant difference in the calculation thinking scores of students in different regions.

Table 8. Regional Variance Analysis.

	SS	df	MS	F	Sig.
BG	1.154	7	0.165	1.019	0.417
WG	48.353	299	0.162		
Total	49.507	306			

4.3.4. Dimensional difference analysis

The differences of the five index dimensions of the evaluation tool of computational thinking are analyzed, and the results are shown in Table 9.

Table 9. Differential Analysis between the Dimensions of Computational Thinking among Junior High School Students.

Dimension	Score	F	LSD
Modularization	3.21±0.62	24.728**	5，4＜2＜1＜3
Formalization	3.19±0.60		
Modeling	3.37±0.63		
Automation	2.97±0.63		
Systematization	2.92±0.72		

Note:**p<0.01;1:modularization;2:formalization;3:modeling;4:automation;5:systematization.

As can be seen from Table 9,the F value between the groups is 24.728, and the significance is indicating that the computational thinking level of junior high school students has significant differences in different dimensions. The LSD(minimum significant difference)method was used to compare the difference between the mean values of different dimensions of computational thinking of junior high school students. It was found that the difference between automation and systematization level of junior high school students was not significant, but the level of automation and systematization level of junior high school students was significantly lower than that of the other three dimensions, and the level of modeling was significantly higher than that of other levels.

4.4. Conclusion

4.4.1. Junior high school students have significant differences in the level of computational thinking in different grades

Students in Grade one have a relatively weak foundation of computational thinking. In problem solving, we begin to try to break down complex problems into smaller, more manageable parts, which is the first manifestation of modularization in computational thinking. However,it is still difficult to describe the problem by using normalized language and symbols, and the formalization level needs to be improved. The second grade students have more knowledge accumulation and skills improvement in computational thinking, and begin to try to use abstract models to describe and solve problems, which shows that their modeling ability has been enhanced. And with the in-depth understanding of programming, the second grade students began to realize the importance of automation, and tried to use computers to achieve some automated tasks.

4.4.2. Junior high school students have higher level of computational thinking modeling

From the difference analysis,it can be seen that junior high school students have a higher level of computational thinking modeling. Junior high school students are in their adolescence. At this stage,students have strong curiosity and thirst for knowledge. They like to explore new things and have a strong interest in computer programming and other technologies. Secondly, the cognitive ability of junior high school students is gradually mature, and they can understand and master relatively complex logical relations and abstract concepts. In the learning process of computational thinking, they can better abstract problems in real life into mathematical models for solving them.In addition, more and more schools begin to pay attention to the training of computational thinking, and set up programming, robotics and other courses, which provide students with practical operation opportunities, so that they can continuously improve the level of modeling in practice.

4.4.3. Junior high school students have low level of automation and systematization of computational thinking

From the difference analysis, it can be seen that junior high school students have a low level of automation and systematization of computational thinking. First of all, junior high school students are at a critical stage of thinking development, their logical thinking and abstract thinking abilities are still under construction, and they may not be able to fully understand and master complex concepts and systems, which leads to certain difficulties in the automation and systematization of computational thinking. Secondly, in the current junior high school education system, there are still some deficiencies in the training of computational thinking. Although many schools have begun to pay attention to the teaching of computational thinking, on the whole, the relevant curriculum and teaching resources are still limited, which cannot meet the needs of students' all-round development.In addition, because junior high school students are in adolescence, some students may have problems such as inattention and weak self-management ability, which also affects their learning of automation and systematization of computational thinking.

5. TRAINING SUGGESTIONS ON COMPUTING THINKING FOR JUNIOR HIGH SCHOOL STUDENTS

5.1. Teaching by stages

In the face of the significant differences in the level of computational thinking of junior high school students in different grades, the method of phased teaching can be adopted. First, differentiated teaching plans can be formulated for students in different grades. For lower school students, emphasis is placed on the development of basic concepts and skills, while higher school students can focus more on the development of complex problem solving and systematic thinking. Secondly, the advanced courses are designed in stages to ensure that students gradually transition to the next stage of learning on the basis of mastering the previous stage. This can not only ensure the learning effect of students, but also avoid affecting the enthusiasm of students because the difficulty span is too large.

5.2. Enhance the automation and systematic training of computational thinking

The level of automation and systematic thinking in the computational thinking of junior high school students is relatively low, so we should strengthen their training. First, it is important to ensure that students have a solid basic knowledge of mathematics, algorithms and data structures, which is the foundation for developing the ability to automate and systematize computational thinking. Secondly, teach students some systematic thinking methods, such as system analysis,design and evaluation, to help them master the way of systematic thinking. Thirdly, through the completion of some complex projects involving system design and automation, students can exercise computational thinking in practical operation and improve their automation and systematization ability. In addition, students can be encouraged to explore automated and systematic applications of computational thinking to improve their abilities through independent learning and practice.

6. SUMMARY

With the promulgation of the Core Literacy of Chinese Students' Development, computational thinking has become one of the four core literacy of information technology discipline. (Zhang Xiaoqing, Li Peng, Wen Chang & Li Haixiao, 2019) Training computational thinking is conducive to improving students' information technology knowledge and skills, and cultivating students' interdisciplinary comprehensive problem-solving ability. (Chen Peng, Huang Ronghuai, Liang Yue & Zhang Jinbao, 2018) The promulgation of the "Compulsory Education Information Technology Curriculum Standards (2022 edition)" will sink computational thinking from the high school stage to the primary and secondary school stage.As a transitional stage between primary school and high school, junior high school plays a vital role in bridging the development of computational thinking. Therefore,research on the development status of junior high school students' computational thinking is also extremely important.

In line with the development requirements of the information age, this paper investigates and studies the current situation of junior middle school students' computational thinking ability through the evaluation tool based on information technology curriculum standards developed by the research team, and concludes that there are significant differences in the level of computational thinking of junior middle school students in different grades. The present situation of high level of modeling and low level of automation and systematization of computational thinking in junior middle school students is presented. In this paper,some suggestions on the training of computational thinking in junior middle school students are given in order to provide reference for the development of computational thinking education in junior middle school students in China. However, due to the small number of respondents, the results of this survey are lacking in universality and need to be further improved in the future.

7. REFERENCES

Chen Peng, Huang Ronghuai, Liang Yue & Zhang Jinbao. (2018). How to Cultivate Computational Thinking: Based on Research Literature from 2006 to 2016 and Latest International Conference Papers. Research on Modern Distance Education (01), 98-112.

Cao Xiaoming & Anna. (2018). A Study on the School-based Curriculum of Information Technology in Senior High School for Cultivating Computational Thinking. Modern Educational Technology (07), 106-112.

Fu Qian, Xie Bochao & Zheng Yafeng. (2019). An Empirical Study on the Promotion of Computational Thinking Development of Junior High School Students by Programming Teaching Based on Graphical Tools. Electronic Education Research (04), 122-128.

Ministry of Education of the People's Republic of China (2018). Information Technology Curriculum Standards for General High Schools (2017 Edition). Beijing, People's Education Press.

Ministry of Education of the People's Republic of China (2022). Information Technology Curriculum Standards for Compulsory Education (2022 Edition). Beijing, Beijing Normal University Press.

Ren Youqun, Sui Fengwei & Li Feng. (2016). Why is it possible for digital natives?—— also discusses the necessity and possibility of computational thinking in information technology education in primary and secondary schools. China E-Education (01), 2-8.

WING J M. (2006). Computational Thinking. Communication of the ACM (S0001-0782) (3), 33-35.

Xie Yueguang, Yang Xin & Fu Haidong. (2017). Description and grading of core literacy of high school students in information technology. China E-Education (05), 8-14.

Zhang Xiaoqing, Li Peng, Wen Chang & Li Haixiao. (2019). A Study on the Cultivation of Computational Thinking in Junior High School Students Based on Scratch School-based Curriculum. China Education Informatization (10), 26-29.

Secondary Education Teachers' Self-Efficacy and TPACK Patterns Relating to Computational Thinking: A Cluster Analysis

ABSTRACT

This research aimed to validate a computational thinking-TPACK survey for secondary school teachers in the compulsory education and identify profiles of teachers' self-efficacy in teaching computational thinking (CT). A questionnaire based on the TPACK framework was administered to 234 teachers, and confirmatory factor analysis and cluster analysis were performed. The results showed that while teachers had sufficient pedagogical knowledge (PK), there was room for improvement in the CT content-related dimensions. STEM teachers rated themselves higher in these dimensions compared to non-STEM teachers. However, there were no significant differences based on gender or teaching experience. Three distinct profiles of teachers' CT-TPACK self-efficacy were identified. This study highlights the importance of professional development and support for teachers in successfully integrating CT into education. Government initiatives that recognize the significance of CT should prioritize these areas to enhance teachers' confidence and competence in teaching CT.

Keywords

computational thinking, self-efficacy, teacher profiles, TPACK, secondary education

1. INTRODUCTION

The fusion of computer technology and programming has significantly impacted society, becoming imperative for academic and professional success in the 21st century (Shute et al., 2017). This influence has given rise to the recognition of computational thinking (CT) as a crucial 21st-century skill, integrated into national curricula globally (Angeli et al., 2016). In alignment with this trend, the Flemish government has made "computational thinking and acting" a mandatory learning goal in secondary schools since September 2019 (Gesquière, 2022). However, challenges arise due to the lack of teacher expertise in computer science and the interdisciplinary nature of CT (Angeli & Giannakos, 2020; Güven & Gulbahar, 2020).

This study aims to understand teachers' self-perceived ability to teach computational thinking and proposes a methodology using two-step clustering analysis for a nuanced examination of their self-efficacy beliefs. By exploring teachers' attitudes and perceptions, the research seeks to contribute insights for designing effective training and preparation programs to integrate CT into education.

1.1. Computational Thinking

The integration of programming and CT in education has historical roots, with CT defined by Wing (2006) as a problem-solving approach drawing from computer science concepts. While definitions vary, CT is recognized as a distinct psychological construct, encompassing problem-solving abilities applicable beyond computer programming (Román-González et al., 2019). The study operationalizes CT based on frameworks from CSTA, ISTE, and literature synthesizing CT abilities (ISTE and CSTA, 2011; Oliveira et al., 2019; Shute et al., 2017)

1.2. Teaching Computational Thinking

Teaching CT involves designing practices that develop CT abilities in learners, impacting STEM fields and critical thinking skills (Angeli et al., 2016). Various frameworks guide CT integration, including those from NSF, ISTE, CSTA, and the Partnership for 21st Century Skills (2011). While research suggests positive impacts on students, challenges persist, such as teachers' limited understanding of CT and a lack of professional development opportunities (Feng & Yang, 2022; Kong et al., 2023).

1.3. Teachers' Sense of Efficacy

Teacher self-efficacy, defined by Bandura (1997) significantly influences student learning. High self-efficacy correlates with innovative teaching practices, technology use, and positive classroom environments. This study explores teachers' self-efficacy through standardized scales like TSES and TSECT, emphasizing the link between CT and technological pedagogical content knowledge (TPACK).

1.4. TPACK

The TPACK framework, developed by Mishra and Koehler (2006), provides a lens for understanding the interplay of technology, pedagogy, and content knowledge in teaching. This study utilizes TPACK to explore teachers' self-efficacy in integrating CT, recognizing the need for comprehensive assessment in educational settings.

1.5. Purpose and Research Questions

Given the ongoing integration of CT into education, this study aims to validate a CT-TPACK survey, identifying CT teaching self-efficacy profiles. The research questions focus on the measurement validity of TPACK self-report items, differences in CT-TPACK variables based on teachers' characteristics, and the identification of teacher profiles regarding CT-TPACK.

2. METHOD

2.1. Participants

A total of 209 grade 8-9 secondary school teachers from the Flemish education system participated, recruited through convenience sampling. The sample included 44 male (21%) and 165 female (79%) teachers, reflecting the gender distribution among teachers in Flemish secondary education in grade 8-9.

2.2. Measures

The questionnaire, adapted from (Schmid et al., 2020), measured teachers' self-efficacy in teaching computational thinking. Items were translated into Dutch for the Flemish context, ensuring content validity through expert reviews. Participants rated items on a five-point Likert scale.

2.3. *Analyses*

Confirmatory factor analysis (CFA) assessed the factor structure's validity. Scale reliability was assessed using Cronbach's α. Pearson's correlation analysis examined inter-correlations among factors and relationships with demographic variables. Two-step cluster analysis identified CT self-efficacy profiles. Silhouette measures determined the optimal number of clusters.

2.3.1. *Validation of the Instrument*

CFA assessed the CT-TPACK scale's factor structure. Scale reliability was evaluated using Cronbach's α. Analyses were conducted using R and packages such as lavaan and psych.

2.3.2. *Correlation Analysis*

Pearson's correlation analysis explored relationships between CT-TPACK factors and demographic variables. Effect sizes were interpreted following J. Cohen's conventions.

2.3.3. *Identifying Profiles*

Two-step cluster analysis was used to identify CT self-efficacy profiles. The silhouette measure determined the optimal number of clusters. IBM SPSS Statistics and R were employed for the analyses.

3. RESULTS

3.1. *Validation of the Instrument*

Addressing the first research question, a reliability analysis and confirmatory factor analysis (CFA) were conducted to assess the fit of the data to the theoretical structure. The seven-scale TPACK model showed an acceptable fit (CFI = .95, RMSEA = .05). Internal consistency, measured by Cronbach's alpha, was high across scales, (except for Pedagogical Knowledge (PK), which just exceeded the acceptable threshold.

3.2. *Factor Loadings and Mean Scores*

All factor loadings exceeded 0.4, indicating moderate to strong relationships. Mean scores for the scales revealed that teachers rated their Pedagogical Knowledge highest, followed by Technological Knowledge (TK), Technological Pedagogical Knowledge (TPK), Computational Knowledge (CK), Pedagogical Computational Knowledge (PCK), Technological Computational Knowledge (TCK), and Technological Pedagogical Computational Knowledge (TPCK). The variation in Pedagogical Knowledge ratings was lower than other knowledge types.

3.3. *Correlation Analysis*

Pearson's correlation analysis demonstrated significant intercorrelations among CT-TPACK subscales. Notably, strong correlations were observed between PCK and TPCK, as well as TPK and TPCK. Teacher experience and gender showed marginal or insignificant correlations with subscales. However, a modest correlation was found between teaching STEM subjects and self-efficacy in CK ($r = .35$) and TCK ($r = -.35$).

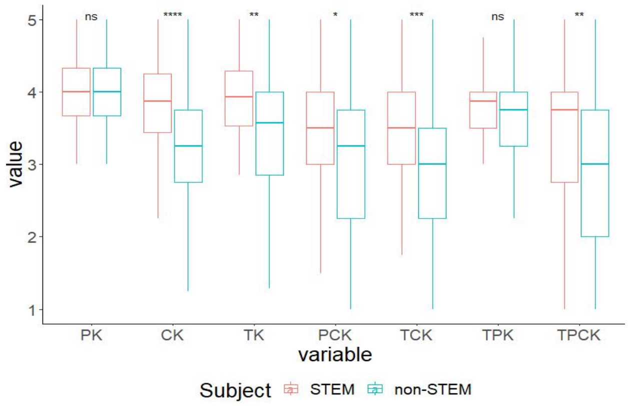


Figure 1. CT-TPACK scale variables by subject

3.4. *3dentifying Profiles*

A three-cluster solution was optimal, explaining variance in knowledge types. Cluster characteristics and a graphic representation were generated. STEM teachers were more prevalent in Cluster 3, while non-STEM teachers were distributed across clusters. Predictor importance indicated TCK, PCK, TPCK, and CK as crucial for cluster membership.

Table 1. Comparison of motivation profiles

	Cluster profile					
	1		2		3	
Scale	*M*	*SD*	*M*	*SD*	*M*	*SD*
Pedagogical Knowledge (PK)	3.89	0.56	4.05	0.47	4.06	0.57
Computational Thinking Knowledge (CK)	2.66	0.56	2.95	0.56	4.04	0.52
Technological Knowledge (TK)	2.76	0.76	3.43	0.54	4.04	0.52
Pedagogical Computational Thinking Knowledge (PCK)	2.01	0.61	3.09	0.50	3.82	0.44
Technological Pedagogical Knowledge (TPK)	3.00	0.76	3.61	0.40	4.05	0.43
Technological Computational Thinking Knowledge (TCK)	1.81	0.50	2.82	0.53	3.73	0.55
Technological Pedagogical Computational Thinking Knowledge (TPCK)	1.89	0.57	2.73	0.59	3.84	0.49
Number of teachers (n=209)	53 (25%)		66 (32%)		90 (43%)	
STEM teachers (n=48)	6 (12%)		9 (19%)		33 (69%)	
Non-STEM teachers (n=161)	47 (29%)		57 (36%)		57 (35%)	

Note. M= score, *SD* = standard deviation, Self-efficacy scale (1-5)

3.5. *Cluster Characteristics:*

Cluster 1 (25%): Low knowledge levels, particularly in PCK and TCK, indicating the lowest TPCK. Cluster 2 (32%): Mediocre knowledge levels across all scales, with slightly higher TPK and neutral levels in other areas. Cluster 3 (43%): High knowledge levels across all scales, with significantly higher TPCK than Clusters 1 and 2.

These results provide a nuanced understanding of CT teaching self-efficacy profiles among secondary school teachers, highlighting variations in knowledge types and their distribution across clusters.

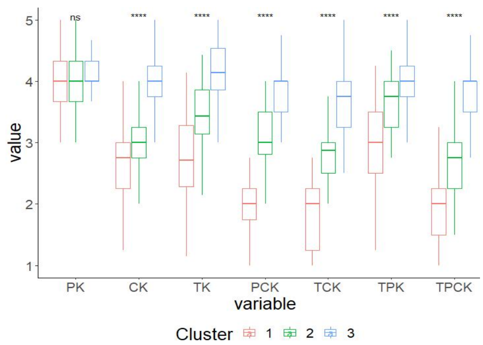


Figure 2. CT-TPACK scale variables by cluster

4. DISCUSSION & CONCLUSION

This study aimed to validate a CT-TPACK survey for Flemish secondary school teachers and explore their self-efficacy in teaching computational thinking (CT) using the seven TPACK knowledge dimensions. The research questions led to valuable insights that contribute to the broader understanding of CT integration in education.

4.1. Research Question 1: TPACK Self-Report Items Validation

The questionnaire demonstrated satisfactory fit indices, validating its reliability. Intercorrelations between subscales aligned with previous studies, reinforcing the robustness and generalizability of the results. Notably, the moderate correlation between Technological Knowledge (TK) and Content Knowledge (CK) suggests an interesting connection, likely rooted in the overlap between Technological Innovation and Computational Thinking.

Varying knowledge levels among teachers, especially in TK and CK, highlight the need for targeted professional development. The overall neutral stance of teachers across domains indicates a call for additional support to facilitate successful CT integration in the Flemish education system.

4.2. Research Question 2: Differences in TPACK Variables

Teaching experience exhibited minimal influence on TPACK self-assessment, emphasizing consistency regardless of experience. However, subject background, particularly STEM vs. non-STEM, revealed noteworthy differences, with STEM teachers reporting higher self-efficacy in CK and TCK. This underscores the role of content specialization in shaping teachers' perceptions of their CT-related knowledge.

These findings offer crucial implications for professional development, urging targeted support for non-STEM teachers in CK and TCK to bridge knowledge gaps effectively.

4.3. Research Question 3: Teacher Profiles

Three distinct clusters of teacher profiles emerged, revealing variations in self-efficacy across TPACK dimensions. Cluster 1, representing 25% of teachers, exhibited the least favorable profile, indicating lower confidence in CT integration. Cluster 2 displayed a moderate profile, while Cluster 3, comprising 43% of teachers, showcased the most favorable profile with high self-efficacy across TPACK dimensions.

STEM teachers predominated in the favorable Cluster 3, aligning with their generally higher self-efficacy. This further emphasizes the importance of subject specialization and content expertise in CT integration.

4.4. Conclusion

In conclusion, this study addressed the challenges of CT integration in compulsory education by validating a CT-TPACK survey and examining teachers' self-efficacy. The findings underscore the importance of targeted professional development initiatives to enhance teachers' knowledge and skills, particularly in content-related dimensions. As governments worldwide recognize the significance of CT, efforts should focus on empowering teachers through ongoing training, fostering effective CT integration, and nurturing critical thinking and problem-solving skills among students.

Building on this study’s contributions, there's a compelling need to translate its findings into actionable strategies for teaching practice. Educators and policymakers can use this research as a blueprint for developing targeted professional development that addresses specific gaps in teachers' CT knowledge and self-efficacy. Future investigations could employ a longitudinal approach with a more varied participant pool to gain richer insights into the evolution of teacher profiles and the tangible effects of professional development on CT pedagogy. By doing so, they can create a more conducive environment for CT integration across educational settings.

5.REFERENCES

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. Computers in Human Behavior, 105, 106185.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. Journal of Educational Technology & Society, 19(3), 47-57.

Bandura, A. (1997). Self-efficacy: The exercise of control. W H Freeman/Times Books/ Henry Holt & Co.

Feng, S., & Yang, D. (2022). Teachers’ Perceived Value, Challenges, and Advice for Implementing Computational Thinking in Elementary Classrooms. Journal of Technology and Teacher Education, 30(3), 293-320.

Gesquière, N. (2022). Computational thinking in Flanders’ compulsory education. CTE-STEM 22: the 6th international conference organized by APSCE,

Güven, I., & Gulbahar, Y. (2020). Integrating Computational Thinking into Social Studies. The Social Studies, 111(5),

ISTE and CSTA. (2011). Operational Definition of Computational Thinking for K-12 Education. In.

Kong, S.-C., Lai, M., & Li, Y. (2023). Scaling up a teacher development programme for sustainable computational thinking education: TPACK surveys, concept tests and primary school visits. Computers & Education, 194,

Mishra, P., & Koehler, M. (2006). Technological Pedagogical Content Knowledge (TPACK) Framework-Educational Technology. Teachers College Record(108), 1017-1054.

Oliveira, A. L., Andrade, W., Guerrero, D., & Melo, M. (2019). How Many Abilities Can We Measure in Computational Thinking?: A Study on Bebras Challenge.

Román-González, M., Moreno-León, J., & Robles, G. (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. In S.-C. Kong & H. Abelson (Eds.), Computational Thinking Education (pp. 79-98). Springer Singapore.

Schmid, M., Brianza, E., & Petko, D. (2020). Developing a short assessment instrument for Technological Pedagogical Content Knowledge (TPACK. xs) and comparing the factor structure of an integrative and a transformative model. Computers & Education, 157, 103967.

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

Wing, J. M. (2006). Computational thinking. Communications of the ACM(49.3), 33-35.

Effectiveness of the design of a gamified Voice-Assisted Chatbot system in aiding modelling, and in personalizing guidance for learning basic OOP

Hannah Xiao-Ai HO[1], Chien-Sing LEE[2]
School of Engineering and Technology, Sunway University
20042537@imail.sunway.edu.my, chiensingl@sunway.edu.my

ABSTRACT

Students often encounter challenges in grasping basic concepts and principles of programming. These are partly due to difficulties in understanding basic concepts, partly difficulties when modelling class diagrams, and partly, the lack of individualized guidance. OOP is chosen for this project, as it enables study of computational thinking's (CT) decomposition, algorithmic logic, pattern recognition and abstraction in diverse similar yet adapted/differentiated scenarios. A chatbot application is proposed in contrast to intelligent tutoring systems (ITS), as ITS requires rigorous modelling and testing. The chatbot can capitalize on CT, to serve as a modular/agile teaching assistant, to assist novice students to reinforce and debug their own understanding, for more meaningful engagement and knowledge retention. The chatbot also provides students with comprehensive OOP and class diagram learning materials, e.g. recap questions, exercises, quizzes, gamified challenges and basic personalized guidance and support, to enhance engagement and motivation in learning OOP and class diagram concepts. A reward system is also implemented to encourage students to take on challenges, earn points and redeem rewards. Alpha-beta user testing are carried out to assess user experience and learner satisfaction. Findings are promising. Sample size however, is small. Furthermore, our weak AI personalized chatbot still needs refinement and enhancement.

KEYWORDS

Efficacy, Design, CT, OOP, Chatbot, Concepts, Modelling, Class Diagrams, Personalized Guidance, Debug, Gamification, Chunking/granularity

1.INTRODUCTION

Mastering OOP requires substantial time and effort to master. This struggle is evident through the prevalence of four common mistakes made by students, when modelling class diagrams. These four common mistakes, include syntactic errors, class-related errors, attribute-related errors and association-related errors. They can eventually affect the quality of the program, as the diagrams may contain inaccuracies, redundancies, or omissions.

Moreover, students sometimes lack individualized guidance as university lecturers lack time to provide instant and personalized feedback to each student, especially in huge class. Lecturers often need to prioritize delivery and discussions, within the given timeframe.

Furthermore, students who have just started learning, are not familiar with programming syntax. Though OOP exercises involve working on a compiler that promptly displays syntax errors whenever mistakes are made, comprehending these syntax errors can be challenging. Errors are mostly described in technical terms, leading to increased frustration and a high likelihood of students giving up easily. Hence, an alternative solution should be developed to address these challenges and enhance students' learning experience.

1.1. Objectives

Wing's (2006) computational thinking (CT) emphasizes the link between computer science and real-life, while Brennan & Resnick's (2012) CT, focuses on developing concepts, practice and perspectives. With these in mind, we have chosen OOP because it enables study of decomposition, algorithmic logic, pattern recognition and abstraction, in diverse similar yet differentiated scenarios.

The main objective of this project is to develop and test the efficacy of the design of an interactive prototype, which provides students with opportunities to engage via practice exercises, quizzes and gamified challenges. The lessons cover fundamental OOP principles, and concepts e.g. classes, objects, inheritance and class diagrams. Simple personalized guidance supports students' specific needs and learning pace. The voice chatbot guides students through learning materials and explains syntax errors using non-technical terms.

2.RELATED WORK

2.1. Challenges to Learning OOP

Several research papers are analysed to identify the challenges, that students normally face in learning OOP.

2.1.1. Issues in Modelling Class Diagrams

Object-oriented programming and class diagrams are interrelated, as the class diagram presents a comprehensive view of the classes and their relationships, including association, aggregation, composition and inheritance, in a single picture (Nikiforova, Sejans, & Cernickins, 2011; Shmallo, & Shrot, 2020).

However, students often encounter issues when it comes to modelling complete and accurate class diagrams, due to the diagrams' size and complexity during the design, analysis and maintenance stages. These result in inaccuracies, redundancies and omissions when modelling a class diagram.

These errors stem from incorrect understanding, which is often a result of either overly general knowledge structures or unclear, faulty or missing knowledge components, or inconsistency in adhering to conventions (Shmallo & Shrot, 2020). More specifically, Kayama, Ogata, Asano, and Hashimoto's (2016) study on 174 beginner-level university students' understanding when modelling class diagrams, has revealed four types of errors (Table 1), i.e. syntactic, class-related, attribute-related and association-related errors.

Table 1. Common Types of Errors Made when Modelling Class Diagrams and their Respective Error Categories

Error type	Criteria	Error type	Criteria
Syntactic	Inadequate notation	Attribute-related	Same attributes are used in > than two classes
	Lack of association, multiplicity		Attribute name defined as value, not property
	Lack of unique names for class, attribute, association		Attribute that describes an action is used
Class-related	Classes with different abstraction levels		Attribute indicating multiplicity is used.
	>=2 classes whose names or attributes have the same meaning in one diagram		Duplicated attributes are used
Association-related	Inadequate association name	Association-related	Inadequate multiplicity

2.1.2. Lack of Individualized Learning Support

In traditional learning settings, it is difficult to provide individualized learning support or address each student's requests (Hobert, & Wolff, 2019). This is true, especially in universities, where large classes pose additional challenges for lecturers in terms of workload and opportunities for meaningful interaction (Cunningham-Nelson, Boles, Trouton, & Margerison, 2019). This will negatively impact students' academic performance and overall satisfaction.

2.1.3. Difficulty in Coping with OOP Learning Tools

Students also find it challenging when coping with the tools utilized when teaching OOP. Some of these teaching tools are initially designed for professional software engineers. This results in programming languages and programming environments, that are too complex for learners to handle and comprehend, especially for beginners (Kölling, 1999).

2.2. Addressing Challenges in Learning OOP

2.2.1. Virtual Learning Companion (VLC)

By analysing the challenges associated with learning OOP, a major portion of these challenges stems from insufficient support available to students during their learning journey. Building on Self's (1988) Intelligent Tutoring Systems (ITS), Chan and Chou (1995) have introduced Learning Companions (LC), for primary schools. At the highest level of the LC, a reciprocal tutoring mode is adopted, i.e., the student and the virtual learning companion, take turns to be tutor and tutee. An overlay student modelling approach enables mapping of the student and LC's performance to the expert teacher's performance. This will enable learning/ refinement of the rules in the LC. More well-known LCs are Google Assistant, and IBM's Jill Watson (Wang, Jing, Camacho, Joyner & Goel, 2020).

2.2.2. Educational chatbots

Personalized educational chatbots provide an easy and cost-effective method to simplify and focus the processes of teaching and learning (Kuhail, Alturki, Alramlawi, & Alhejori, 2023; Okonkwo, & Ade-Ibijola, 2021). Through chatbot conversations, lecturers can identify areas where students struggle, and assess their learning abilities (Okonkwo & Ade-Ibijola, 2021b). In another example, Latham, Crockett, McLean, and Edmonds' (2012) study finds that, students provided with a learning path tailored to their individual learning styles, achieve 12% more accurate answers, compared to those who utilized chatbots without personalized learning materials. Personalized educational chatbots can generate learning materials customized to meet the specific individual needs of students, resulting in increased motivation and engagement (Baskara, 2023). Thus, by establishing a more assisted learning environment, students can pursue their studies, at their preferred pace, and in their preferred style. This greatly helps in facilitating a thorough comprehension of the topics being studied, ultimately resulting in better knowledge retention.

2.2.3. Types of Chatbot

There are different types of chatbots, e.g. rule-based chatbots which are more structured with pre-defined input, processing and output, AI (unsupervised or supervised learning) chatbots, which learns over time, linguistic chatbots e.g. Messenger, and contextual chatbots. Contextual chatbots are the most advanced, as they reference prior conversations, and their history, to identify the progression of a conversation, and possibly what may be next (Chaturvedi, Srivastava, Rai, & Cheema, 2020). These chatbots can involve only text, or include voice as well.

3. METHODOLOGY

This project is implemented using the iterative incremental prototyping methodology, where the system's functions are developed in increments, starting with the main functionalities. The chatbot system provides personalized guidance on learning OOP and class diagram concepts by allowing students to express their needs through a menu of response options. Subsequently, recap questions, exercises, quizzes and gamified challenges that cover the fundamental OOP and class diagram concepts are provided for students to engage in, so that they can assess their understanding. The prototype's context diagram is presented in Figure 1.

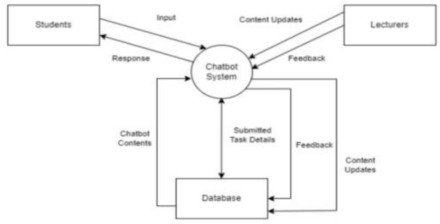


Figure 1. Prototype's Context Diagram

Moreover, Grover, Pea, and Cooper (2015) promote multiple forms of assessments, to develop a bigger picture of the students' CT development/learning outcomes. Hence, we have applied Schrepp, Hinderks, and Thomaschewski's.

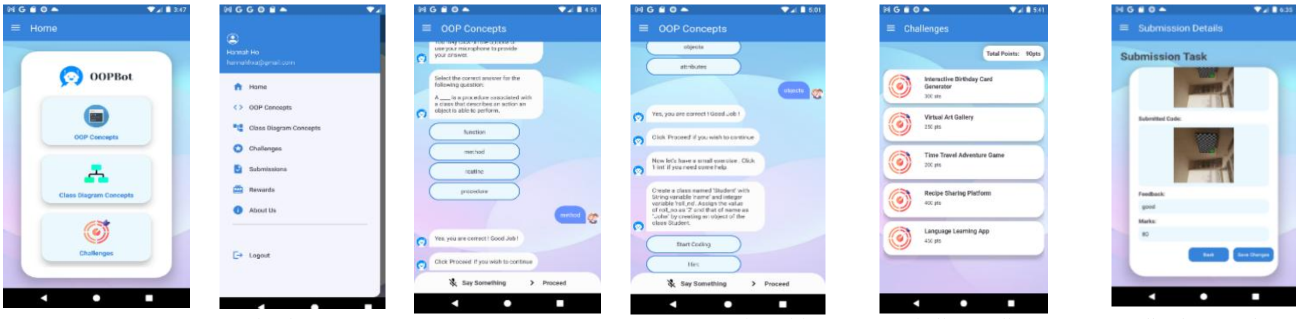
(2017) User Experience Questionnaire (UEQ) and Lewis'/IBM's Computer Systems Usability Questionnaire (CSUQ) for this pilot study. These questionnaires are carried out via Google Form. Hence, convenient sampling.

4. PROTOTYPE DEVELOPMENT AND TESTING

The chatbot system consists of 7 sections, i.e., the home page, OOP concepts content, class diagram content (20 classes), challenge content, submission page, rewards page, and the about us page. The BrainShop API acts as a service that works with Chat Natural Language Processing tasks (Brainshop, nd).

4.1. Screenshots (alpha user testing)

Some screenshots from the alpha prototype is presented in Figure 2 below. Feedback is factored in, and each reward is assigned a redeem point. The total points for students are displayed at the top. If the redeem points of a reward are lesser than a student's total points, the student can redeem.



Home page | Navigation Drawer (Student) | Recap Q & A | OOP Exercise with hints | Challenges list | Feedback to Students' Answers to Challenges

Figure 2. Alpha System's Recap, Exercise and Submit Code Screenshots

4.2. Alpha testing

4.2.1. Demographics

The demographics (Figure 3) shows that most of the respondents are female, around 18-24 years old, pre-university or university graduates, beginners in programming and object-oriented programming (OOP).

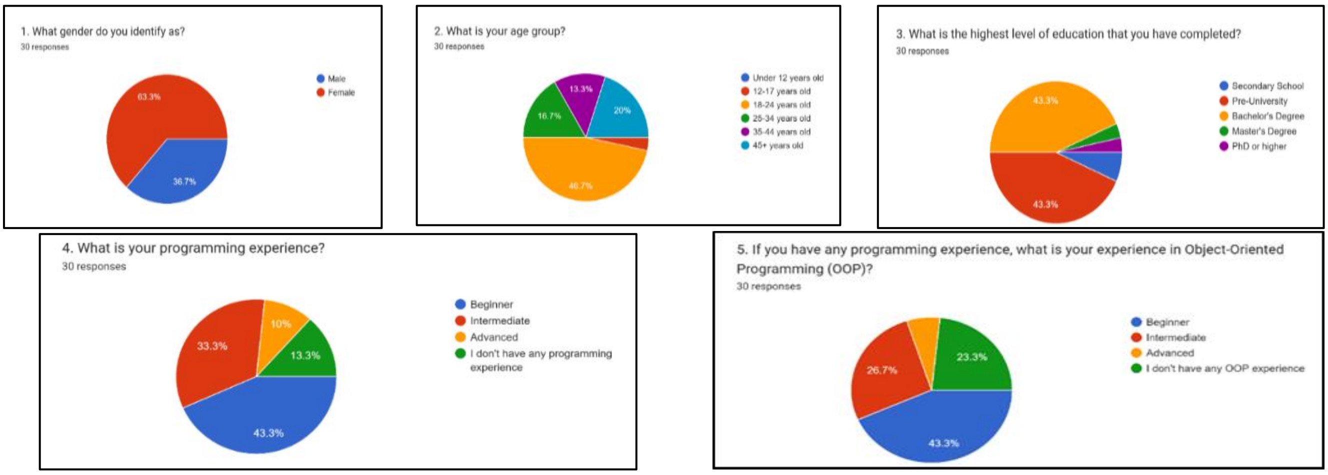


Figure 3. Demographics from Alpha Testing

4.2.2. User Experience Questionnaire (UEQ) testing

The UEQ alpha testing mean results all score above 3.5/5. "Confusing vs. Clear" has the highest mean of 4.13/5 (83%) (Table 2), followed by easy, interesting and inventive at 3.97/5 (79%). "Boring vs. Exciting" achieve a mean of 3.83/5 (77%) and "Usual vs. Leading edge" 3.73/5 (75%). The UEQ alpha testing results, shows that users are clear about the core idea of the application. Thus, interaction approaches within the application and the design of the application need to be improved.

Table 2. Alpha UEQ testing results

Constructs	Avg.	Constructs	Avg.	Constructs	Avg.	Constructs	Avg.
Confusing/Clear	**4.13**	(Un)interesting	**3.97**	Obstructive/Supportive	3.93	Boring/Exciting	3.83
Complicated/Easy	**3.97**	Conventional/Inventive	**3.97**	Inefficient vs. Efficient	3.87	Usual/Leading edge	3.73

4.2.3. CSUQ testing

From Table 3, we find that majority of the higher averages are in the affective domains (highlighted in yellow), e.g. *easy, need, pleasant, comfortable, satisfied, effective, simple, like.* These indicate that more needs to be done to improve productivity, in terms of effectiveness and efficiency.

Table 3. CSUQ Alpha Testing Results (Averages in Descending Order)

Questions	Avg.	Questions	Avg.
The **information** provided is **easy** to understand.	5.83	It was **simple** to use this system	5.47
It is **easy** to find the **information** I **needed**	5.73	I like using the **interface** of this system	5.47
The **interface** of this system is **pleasant**	5.70	Overall, I am **satisfied** with how **easy** it is to use.	5.40
I feel **comfortable** using this system	5.63	The system has all the **functions/capabilities** expected.	5.40
The **organization of information** on the screens is clear	5.63	I am able to **complete my work quickly** using this system.	5.37
Overall, I am **satisfied** with this system	5.63	I believe I became **productive quickly** using this system	5.27
It was **easy** to learn to use this system	5.57	I can **effectively complete my work** using this system	5.23
The **information** is **effective** in helping me complete the tasks	5.50	I am able to **efficiently complete my work** using this system	5.23

Based on Table 3, a hierarchy/dependency among factors relevant to the sample respondents, is derived (Figure 4).

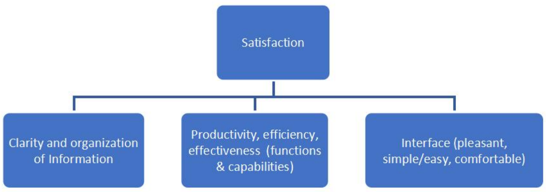


Figure 4. Hierarchy/dependency in Value Propositions

4.3. Beta iteration (refinements from alpha testing)

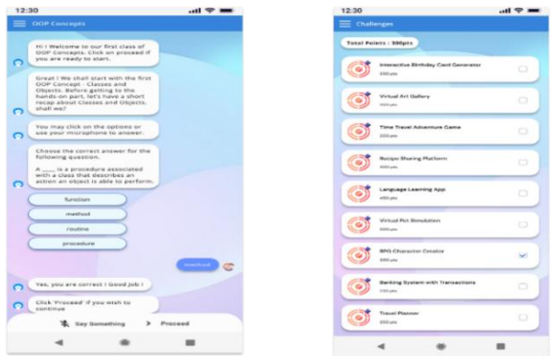
- Recap questions are provided to assess students' knowledge. This is more engaging and interactive.
- A rewards page is added for students to redeem points for rewards, if they score >=70 for the challenge.
- A list of more exciting real-life scenario challenges is added. These test understanding of logic, class diagram.

Sample screenshots are presented in Figure 5.

4.3.1. Beta testing

4.3.1.1. Demographics

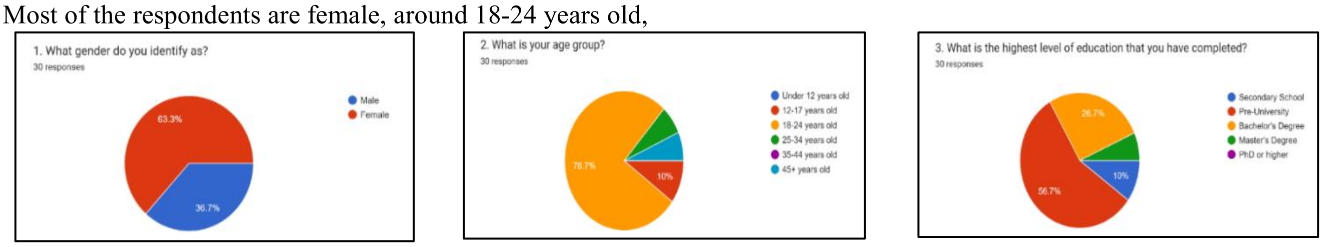
Most of the respondents are female, around 18-24 years old, pre-university graduates. Among these, 30% are beginners or have intermediate programming experience and 26.7% do not have any OOP experience (Figures 6a, b, c, d). Compared to alpha's user testing (Figure 7a), for beta, there is an increase of 13.3% beginners, 3.4% intermediate, a decrease of 3.3% advanced, and a decrease of 13.4% respondents without OOP experience (Figure 7b).



More interaction approaches (Recap questions, Rewards page)

Added challenges mostly related to real-life scenarios

Figure 5. Screenshots from the Beta Prototype



Figures 6a, b, c. Beta Testing Demographics

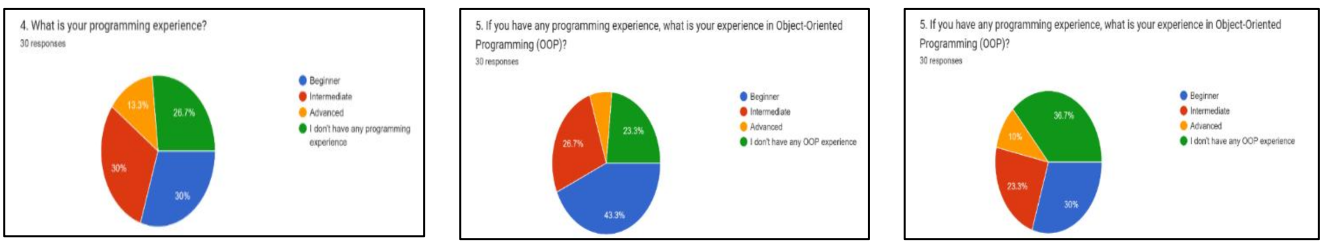


Figure 6d, Figure 7a. Alpha Testing Respondents' Experiences, *Figure 7b.* Beta Testing Respondents' Experiences

4.3.3.2. Learning engagement (UEQ) testing

Table 4 presents the UEQ beta testing results, in descending order. The 3 highest improvements between alpha-beta (AB) testings are leading edge, interesting, and efficient. There is room for improvement in terms of inventiveness

Table 4. UEQ Beta Testing Results (Averages in Descending Order and Degree of Improvement)

Construct	Avg.	Construct	Avg.	Construct	Avg.	Construct	Avg.
Confusing/Clear	4.23 (+0.10)	Complicated/Easy	4.13 (+0.16)	Usual/Leading edge	4.10 **(+0.37)**	Boring/Exciting	4.06 (+0.23)
Uninteresting/ Interesting	4.23 **(+0.26)**	Inefficient/ Efficient	4.13 **(+0.26)**	Obstructive/ Supportive	4.06 (+0.13)	Conventional/ Inventive	4.00 (+0.03)

4.3.3.3. CSUQ testing

Similar to alpha testing's findings, the affective factors (satisfied, easy, pleasant, effective, simple, like, comfortable), have higher averages compared to productivity factors (Table 5). As noted from Figures 7a, b, for beta testing, there are 13.3% more beginners, 3.4% more intermediate level, 3.3% less advanced, and 13.4% less respondents without OOP experience. This indicates that prior programming experience mediates usability.

These findings complement Poursaed and Lee's (2010) prior study on factors which contribute towards more effective problem-solving skills and thus, more meaningful learning in mobile and ubiquitous learning, in terms of user experience and satisfaction. The difference is Poursaed and Lee's (2010) study is for the learning of Java and without chatbot. Thus, design-CT-self-efficacy are complementary, to diverse demographics, objectives, phases of learning, and platforms (formal/informal, distance/flipped learning).

We thus agree with Grover, Pea, and Cooper (2015) and Zhang and Specht (2023) that more attention should be placed on designing suitable assessments, to develop more holistic assessments. We agree with Kong's (2019) assessment modes, due to the common base, i.e., Brennan and Resnick's (2012) concepts, practice and perspectives. Our research is however, scoped to Higher Education within computer science/computing groundings only.

As for perspective developments, as the findings have indicated, we need to first enable easier search of texts in the chat or to enable bookmarking, to reduce cognitive load. Greater decomposition of lessons/chat sessions and stronger AI, may help, in developing Kong's (2019) programming empowerment. Other assessment criteria reviewed in Lee and Jiang (2019) are exemplary when localizing rubrics.

Table 5. CSUQ Beta Testing Results (Averages in Descending Order and Degree of Improvements)

Questions	Avg.	+	Questions	Avg.	+
Overall, I am **satisfied** with this system	6.10	**+0.70**	The **organization of information** on the screens is clear	6.00	+0.37
It was **easy** to learn to use this system	6.10	+0.37	This system has all the **functions and capabilities** I expect	6.00	**+0.60**
The **interface** of this system is **pleasant**	6.06	+0.36	I am able to **efficiently complete my work** using this system	6.00	**+0.77**
The **information** is **effective** in helping completing tasks, scenarios	6.06	**+0.56**	I am able to **complete my work quickly** using this system	5.97	**+0.60**
It was **simple** to use this system	6.06	**+0.59**	Overall, I am **satisfied** with how **easy** it is to use this system	5.87	+0.47
I like using this system's **interface**	6.06	**+0.59**	It is **easy** to find the **information** needed	5.84	+0.11
The **information** provided for the system is **easy** to understand.	6.03	+0.20	I believe I became **productive quickly** using this system	5.84	**+0.57**
I feel **comfortable** using this system	6.00	+0.37	I can **effectively complete my work** using this system	5.77	**+0.54**

5. CONCLUSION

The main goals of this project are to develop interactive learning opportunities to engage students more actively, and to address the difficulties of learning OOP. OOP is chosen as it enables study of CT's decomposition, algorithmic logic, pattern recognition and abstraction in diverse similar/ differentiated scenarios. The chatbot system has provided activities, e.g. multiple-choice recap questions, which the students can answer via clicks or voice, exercises, quizzes and gamified challenges. On-demand hints generate learning materials, customized to meet specific student needs. Students can also earn points and redeem rewards by scoring 70 or more than 70 points, for a challenge.

There are limitations to this humble study, as the current personalization and remedial strategies are at the weak AI level, as it is an undergraduate capstone. Furthermore, our sample size is small. Hence, findings cannot be generalized.

6.ACKNOWLEDGEMENT

We thank the anonymous reviewers for their invaluable constructive feedback. The second author reframed the first author's undergraduate capstone project to design-computational thinking, extended the analysis/interpretation, added the hierarchy/dependency among value propositions. Thanks to Dr. K. Daniel Wong, SG, for past collaboration on Dym et. al.'s (2005) design thinking, Prof. Bo Jiang for past collaboration on CT-HCI scaffolds (China's Natural Science Foundation grant no.71704160), Prof. Ashok K. Goel (ex-Fulbright collaborator) for promoting IBM Jill Watson on LinkedIn, Laiye for its exemplary personalized CT, Prof. Siu-Cheung Kong and Prof. Hal Abelson for promoting CT since 2017. This paper is funded by the past Fulbright Visiting Scholar Fellowship.

7.REFERENCES

Baskara, F. X. R. (2023). Chatbots and flipped learning: Enhancing student engagement and learning outcomes through personalised support and collaboration.

International Journal of Recent Educational Research, *4* (2), 223-238.

BrainShop API. https://autonlp.ai/apis/brain shop. (Accessed Dec. 7, 2023).

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Annual American Educational Research Association Meeting*, BC, Canada, 1–25.

Chan, T. W. & Chou, C. Y. (1995). Simulating a learning companion in Reciprocal Tutoring System. *International Conference on Computer-assisted Collaborative Learning,* Bloomington, Indiana, October 17-20, 1995, 49-56.

Chaturvedi, A., Srivastava, S., Rai, A. and Cheema, A. S. (2020). A hybrid approach to develop and integrate chatbot in health informatics systems. *International Conference on Health Informatics*, 774-781.

Cunningham-Nelson, S., Boles, W., Trouton, L. & Margerison, E. (2019). A review of chatbots in education: Practical steps forward. *Australasian Association for Engineering Education: Educators Becoming Agents of Change: Innovate, Integrate, Motivate.* Australia, 299-306.

Dym, C. L., Agogino, A.M., Eris, O., Frey, D. D, & Leifer, L. J. (2005). Engineering Design Thinking, Teaching, and Learning. *Journal of Engineering Education*, *94*, 103-120.

Google Assistant. https://assistant.google.com

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school learners. *Computer Science Education, 25* (2), 199–237.

Hobert, S. & von Wolff, R. M. (2019). Say hello to your new automated tutor – A structured literature review on pedagogical conversational agents. *Wirtschaftsinformatik*, Germany, 301-314.

Kayama, M., Ogata, S., Asano, D. K., & Hashimoto, M. (2016). Educational criteria for evaluating simple class diagrams made by novices for conceptual modelling. *International Conference on Cognition and Exploratory Learning in Digital Age*, Germany, October 28-30, 2016, 319-323.

Kuhail, M. A., Alturki, N., Alramlawi, S. and Alhejori, K. (2023). Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, *28* (1), 973-1018.

Kölling, M. (1999). The problem of teaching object-oriented programming, Part I: Languages. *Journal of Object-Oriented Programming*, *11* (8), 8-15.

Kong, S. C. (2019). Components and Methods of Evaluating Computational Thinking for Fostering Creative Problem-Solvers in Senior Primary School Education. In Kong, S. C. & Abelson, H. (Eds.). *Computational Thinking Education*. Springer, 119-140.

Laiye https://laiye.com/en/blog/laiye-uibot-and-wulai-chatbot (Accessed: Jan 21, 2024).

Latham, A., Crockett, K., McLean, D. & Edmonds, B. (2012). Adaptive tutoring in an intelligent conversational agent system. In *Transactions on Computational Collective Intelligence VIII*. Springer, 148-167.

Lee, C. S. & Jiang, B. (2019). Assessment of Computational Thinking (CT) in Scratch fractal projects: Towards CT-HCI scaffolds for analogical-fractal thinking. *International Conference on Computer-Supported Education*, *1,* Crete, Greece, May 2-4, 2019, 192-199.

Lewis, R. (1995). Computer System Usability Questionnaire. Psychometric Evaluation and Instructions for Use. *International Journal of Human-Computer Interaction, 7* (1), 57-78.

Nikiforova, O., Sejans, J. & Cernickins, A. (2011). Role of UML class diagram in object-oriented software development. *Scientific Journal of Riga Technical University*, *44* (1), 65-74.

Okonkwo, C. W. & Ade-Ibijola, A. (2021). Chatbots applications in education: A systematic review. *Computer and Education: Artificial Intelligence*, *2*, 100033, 1-10.

Poursaeed, B., & Lee, C. S. (2010). Self-initiated curriculum planning, visualization and assessment in improving meaningful learning: A comparison between mobile and ubiquitous learning. *International Conference on Technology for Education*, Mumbai, India, July 1-3, 2010, 107-113.

Self, J. (1988). Bypassing the intractable problem of student modelling. *International Conference of Intelligent Tutoring Systems.* Montreal, Canada, 18-24.

Shmallo, R. & Shrot, T. (2020). Constructive use of errors in teaching the UML class diagram in an IS engineering course. *Journal of Information Systems Education*, *31* (4), 282-293.

Schrepp, M., Hinderks, A., & Thomaschewski, J. (2017). Design and Evaluation of a Short Version of the User Experience Questionnaire. *IJIMAI, 4* (6), 103-108.

Wang, Q., Jing, S., Camacho, I, Joyner, D. & Goel, A. K. (2020). Jill Watson SA: Design and evaluation of a virtual agent to build communities among online learners. *CHI Conference on Human Factors in Computing Systems.*1-8.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49* (3), 223.

Zhang, X. & Specht, M. (2022). A Review of Reviews on Computational Thinking Assessment in Higher Education. *CTE-STEM*, Netherlands, June 15-17, 2022, 98-103.

Unveiling the Associations between Print Color of cCTt and Types of Errors

Ben Avital[1], Arnon Hershkovitz[2]
[1,2]Tel Aviv University, Israel
benavital@mail.tau.ac.il, arnonhe@tauex.tau.ac.il

Abstract

This research investigates the impact of color scheme on elementary school computational thinking (CT) assessments. The study involved 116 4th-grade Israeli students from diverse socio-economic backgrounds, comparing performance between full color and grayscale versions of the competent Computational Thinking Test (cCTt). Overall, students in the grayscale group performed better than those in the color group. Analysis of error types indicated that grayscale printing reduced errors that depict lower levels of understanding and had no associations with responses that depict higher levels of understanding. The study contributes valuable insights into the role of color in CT assessments, emphasizing the importance of optimizing assessment tools for young learners.

Keywords

Assessment, Pen-and-Paper test, Color, Computational Thinking, cCTt, Elementary School

1. Introduction

One of the most common approaches to teach and assess computational thinking (CT) is to present students with visual blocks, with each block represents a different action, and to use these blocks to construct a solution to a puzzle. This approach is inspired by the way CT was initially introduced to young learners, i.e. via Scratch block-based programming. Today, blocks are closely associated with CT, particularly at a young age, and have migrated from digital learning environments to pen-and-paper assessment tools.

To make these environments and tools more friendly to young students, colors are used to code different types of actions. In the context of comprehension and learning, color has been long suggested to play an important role. As for the importance of color in learning and assessment, its role in CT assessment needs to be addressed. Nonetheless, there remains an evident gap in the existing literature regarding the direct associations between CT and color; this is indeed the focus of the current study.

1.1. The Relationship Between Color and Learning

Color perception is fundamental to human comprehension of the world; understanding relationships between colors and objects are vital mechanisms for higher-level color processing (Derefeldt et al., 2004; Siuda-Krzywicka & Bartolomeo, 2020) . Therefore, the expression of color in learning resources--e.g., in text-based materials, visualizations, or videos, is crucial for learning and development (Liu et al., 2021).

Moreover, color has exhibited relevance in diverse contexts that can be linked to Computational Thinking (CT). For example, color scheme was found to be important in learning to program (Liu et al., 2021) . In a study investigating the correlation between two video lecture conditions (color-coded vs. grayscale) and the efficacy of learning programming, color coding was observed to enhance programming comprehension more effectively than grayscale. Also, color was found to play a role in complex problem solving when subgoals are color-labeled, specifically with regarding to affect and mood (Ramos-Nuñez et al., 2018).

CT platforms and assessment tools—from block-based platforms like Scratch to assessment environments like Bebras to pen-and-paper tests like cCTt—often utilize tasks presented in various color shades. Therefore, it is important to explore the role of color in CT.

1.2. Measuring Computational Thinking

Computational thinking (CT) evaluation methods encompass various areas, emphasizing problem-solving abilities without necessarily focusing on programming (Ezeamuzie & Leung, 2021; Ogegbo & Ramnarain, 2022). Moreover, pen-and-paper assessments are more accessible and still promote learning (Sun et al., 2021).

For elementary students, the Beginner's CT test (BCTt) has been effective within the age range of 5-7 years, while the competent CT test (cCTt) is tailored for third and fourth graders (ages 7-9), demonstrating effectiveness in assessing CT skills (El-Hamamsy, Zapata-Cáceres, Barroso, et al., 2022; El-Hamamsy, Zapata-Cáceres, Marcelino, et al., 2022). These assessments comprise of a set of puzzles that include a grid on which a chick and its mother hen are positioned; students should choose the solution that leads the chick to the hen under the conditions of the puzzle. The possible paths—4 for each puzzle—are decoded in blocks which represent different operations (see under Methodology section for more details). The utilization of colors aids in distinguishing between different operations (blocks).

1.3. Research Goals

The purpose of the present study is to compare elementary school students' cCTt performance between color and grayscale test versions.

1.4. Research Questions

1. What are the relationships between computational thinking and print type?
2. What are the relationships between types of errors in cCTt and print type?

2. Methodology

2.1.Population

The study comprised of 116 fourth-grade students (ages 9-10 years old) from three elementary schools in Israel. The recruitment process involved reaching out to school administrators and participation upon their approval. The participating schools were located within a large city which represents diverse socio-economic backgrounds ranging from low to medium-high. Of the participants, 45% were girls (52 out of 115), 55% were boys (63 out of 115); and one participant did not indicate their gender. The data collection sessions were incorporated into the school schedule, and participation was voluntary. Of the whole grade population in these schools, an overall of fifty-two students abstained from participation due to various reasons like absence, lack of consent, participation in off-site activities, etc. Additionally, we excluded six students who responded to less than half of the questions.

2.2.Design and Procedure

The study received approval from both the Ministry of Education, which was necessary because data collection took place in school, and the authors' Institutional Review Board (IRB). Parents and relevant school staff were informed about the study's objectives and were provided with detailed information to address any potential objections or concerns regarding data collection. Participation was managed on an opt-out basis.

Data collection was held in June 2023. The class session began with a 5-minute brief explanation on how to complete the questionnaire and an overview of the types of questions. This was followed by approximately 40 minutes during which students filled out the questionnaires; students who needed more time kept working on the questionnaire until completing it. The research team was available to address individual questions aimed at clarifying the content of the questionnaire.

2.3.Instruments

The competent Computational Thinking Test (cCTt): The cCTt (Computational Thinking Test) is a pen-and-paper multiple-choice questionnaire designed to evaluate computational thinking skills among third and fourth-grade students. The questions were sourced from a recent study (El-Hamamsy, Zapata-Cáceres, Barroso, et al., 2022) which assessed and validated a computational thinking questionnaire tailored to elementary school students in these grades. The decision to adapt a questionnaire specifically for this age group was supported by theoretical and research justifications (El-Hamamsy, Zuuerey, et al., 2022; El-Hamamsy, Zapata-Cáceres, et al., 2022). Initially containing 25 questions, a recommendation was made to utilize a subset of 15 questions covering fundamental concepts from the original questionnaire (sequences, simple loops, nested loops, conditional and while statements), excluding the most challenging questions involving combinations, deemed difficult for the students (El-Hamamsy, Zapata-Cáceres, et al., 2022).

Each questionnaire item is accompanied by a diagram illustrating various potential response types (El-Hamamsy, Zapata-Cáceres, Barroso, et al., 2022), including the correct answer that exemplifies computational thinking and three types of mistakes. The questionnaire underwent translation into Hebrew and underwent review and refinement by the first and second authors of this paper. Subsequently, the test was administrated to two 3rd- and 4th-grade students who evaluate the questionnaire's clarity and comprehensibility; adjustments were made based on their feedback. The questionnaire comprises 15 progressively challenging questions, each presenting a scenario of guiding a chick to its mother hen while potentially accomplishing side-goals like picking up a flower or avoiding a cat. Students are presented with four possible answers per question and required to select one (refer to

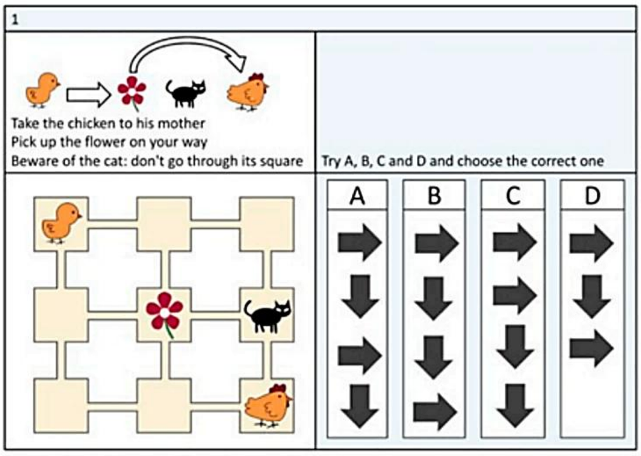


Figure 1 for an example).

Figure 1. Questions of the Sequences of the cCTt

2.4.Research Variables

Personal Characteristics*:* Our questionnaire included two additional self-reported items: gender and age.

Print Type: 55 questionnaires (47%) were printed in color, and 61 questionnaires (53%) were printed in grayscale.

Computational Thinking: Overall Computational Thinking score was calculated by taking the number of correct solutions (normalized by 15). Correct Solutions (M=0.71, SD=0.19, N=116). For each question, we calculated for each student the correctness of the answer. We then averaged this count for each student across all questions.

Types of incorrect answers (profiles): For each cCTt question, there are four potential responses of which one is correct, and the others are categorized into three types of errors, known as profiles (El-Hamamsy, Zapata-Cáceres, Barroso, et al., 2022). Profile 1 entails an incorrect target or a primary focus on arrow directions, disregarding repetition and condition. Profile 2 involves ignoring objectives or misinterpreting repetition and condition. Profile 3 consists of misinterpreting either objectives or non-arrow-related restrictions. Frequency for each profile was calculated as the count of incorrect responses under this profile divided by the total number of questions (15).

To better understand the profiles, we will use the first cCTt questions, which is focused on simple sequences, see Figure 1. Profile 1 corresponds to option D: Incorrect target. The student got confused about the target, and chose the cat as the target. Profile 2 is represented by option C: The student reaches the target, but ignoring objectives – failing to collect the flower and avoiding crossing the cat's square. Profile 3 is reflected in option A: Misinterpreting objectives – in this case, the student does collect the flower, and reaches the target, but forgets to avoid passing through the cat's slot. Option B signifies the correct answer– the student succeeds in all the terms of the task.

For another illustration of profiles, we will examine

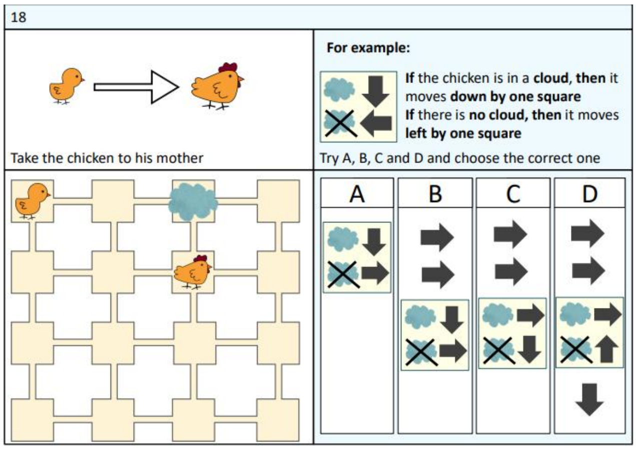


Figure 2. Questions of the Conditional Statements of the cCTt

question number 18, which is focused on if-else conditionals, see Figure 2. Profile 1 is mirrored in option D: Wrong arrow directions – ignoring the box with the clouds in it. Profile 2 is conveyed by option C: Misinterpreting condition – The box with the clouds was misinterpreted as expressing one downward progression. While in this case, the player can only go down if there is no cloud. Profile 3 is portrayed in option A: Misinterpreting restrictions – the interpretation of the box with the cloud as a kind of loop that can be repeated several times.. The correct answer is option D.

2.5.Analysis

Analysis was conducted using JASP version 0.17.3. Since some of our variables did not exhibit a normal distribution, we opted to employ the Spearman test and Mann-Whitney U test for conducting correlations in our analyses.

To assess the normality of the variable distribution, we applied the methodology developed by Kim (2013), which tests whether the Z-scores of Skewness and Kurtosis are above an a-priori threshold which depends on the population size. Z-scores are calculated by dividing the Skewness or Kurtosis values by their Standard Error. The relevant threshold for our study is 3.29, and if either of the Z-scores exceed it – we should declare for deviation from normality. Indeed, the Z-score values for the CT values were indicative of a non-normal distribution (see Table 1). Moreover, effect size is given by the rank biserial correlation (RBC); it is commonly agreed that RBC values between 0.1-0.2 denotes a small effect size, between 0.3-0.5 denotes a medium effect size, and between 0.8-1 denotes a large effect size (Cureton, 1956; Mann & Whitney, 1947).

Table 1. Z-scores, Means, Standard Deviations, and Standard Error for the 5 Response Profiles (N=116)

Variable	Z-skewness (Z-kurtosis)	mean (SD)	SE
Missing Value	10.67 (13.90)	0.04 (0.08)	0.01
Profile 1	4.67 (0.64)	0.08 (0.11)	0.01
Profile 2	2.8 (0.38)	0.11 (0.09)	0.01
Profile 3	6.32 (3.96)	0.06 (0.08)	0.01
Overall CT Score	4.63 (3.24)	0.71 (0.19)	0.02

3. Findings

No significant difference was found between gender and computational thinking (U=1757, at p=0.50). Therefore, we conclude that the entire population can be seen as a unified whole.

3.1.Computational Thinking, Demographics, and Print Type (RQ1)

A notable correlation was identified between computational thinking and print type; notably, among the students who filled-up a color printed version of cCTt, the average was lower compared to those who filled-up the grayscale printing version; this difference had a medium effect size (Table 2).

Table 2. Comparison between Computational Thinking and Print Type

Variable	mean (SD)	U	p	RBC
Print Type		1137.5	0.003	-0.32
color	0.66 (0.19)	(N=55)		
grayscale	0.75 (0.18)	(N=61)		

3.2.Types of Errors in cCTt and Print Type (RQ2)

The values of the three profile variables represent the frequency of errors of each type; values were calculated as the count of incorrect responses under each profile divided by the total number of questions (15). On average, students had 11% of Profile 2 errors, 8% of Profile 1 errors, and 6% of Profile 3 errors. The examination of error types revealed a significant association between CT and printing type. The grayscale printing group displayed fewer Profile 1-type errors, with a small-medium effect size (statistically significant with a small effect size), and fewer Profile 2-type errors (marginally significance, small-medium effect size). No significant difference was found regarding Profile 3-type errors. Findings are summarized in Table 3.

Table 3. Error Type Comparison between cCTt and Print Type

Variable	mean (SD)	U	p	RBC
	Profile 1			
Print Type		2114.0	0.01	0.26
color	0.11 (0.10)	(N=55)		
grayscale	0.06 (0.09)	(N=61)		
	Profile 2			
Print Type		2009.0	0.06	0.21
color	0.12 (0.09)	(N=55)		
grayscale	0.09 (0.08)	(N=61)		
	Profile 3			
Print Type		1769.0	0.59	..
color	0.07 (0.01)	(N=55)		
grayscale	0.06 (0.08)	(N=61)		

4. Discussion

In this study, we explored the associations between CT—as measured by the pen-and-paper cCTt—among 4th-grade children (N=116) and print type (color or grayscale). Overall, we found that the grayscale group performed better than the color group. It is plausible that this finding may be attributed to an increased cognitive load associated with information processing, specifically that color processing added another layer of required encoding activity in the brain (Gnambs et al., 2015; Wenjie et al., 2021). This additional effort may have hindered problem-solving abilities (Beddow, 2018) ; indeed, reducing cognitive load was shown to improve students' performance (Ben-Haim et al., 2019; Gillmor et al., 2015). Importantly, colors can be used as a means to reduce cognitive load and to improve performance; this requires an association between colors and required cognitive operations, and should be accompanied by getting students used to these associations (Ekman & Waliullah, 2019).

Moreover, we took a step forward, and explored the types of errors students made, in order to understand whether color has a unique impact on certain error types. In a sense, the profiles that correspond to types of errors represent the level of understanding, ranging from Profile 1 (low level of understanding) to Profile 3 (high level of understanding albeit still incorrect answer). Grayscale printing appeared to decrease Profile 1 and Profile 2 errors; Profile 3 errors showed no color-related difference. This may suggest that color was mostly harmful to students with lower levels of understanding.

These findings echo previous studies of the different effects multiple information channels have on students with low vs. high levels of knowledge (Kastaun et al., 2021; Kuldas et al., 2014). Although those studies were mostly focused on multimedia and multiple representations of information, the main insight from them still holds: Students of different levels of knowledge may react differently to cognitive load, hence instructional design should be better adapted to level of knowledge. This is an important lesson to assessment as well, and here we highlight the role color plays in CT assessment.

Another explanation to our findings—that goes beyond cognitive load—relates to affective aspects of learning. The utilization of color may trigger emotional processes, consequently influencing learning processes; that is, color adds to the cognitive load and can make the learning process more difficult. It seems that among female learners, color caused poorer performance compared to gray. (Liew et al., 2022) . On the other hand, bright colors that are interpreted in an emotionally positive way, may reduce cognitive load and positively impact their ability to solve problems (Le et al., 2021). Furthermore, color may impact various sub-groups differently than others, as was shown in the context of solving puzzles: Girls demonstrated a superior ability to solve problems presented in colors compared to boys (Honrales, 2020) . This non-cognitive impact of color on CT should be further explored.

Considering that color can impact student performance in various ways (Al-Ayash et al., 2016; Amarin & Al-Saleh, 2020; Kumi et al., 2013) , it is recommended to carefully consider color schemes while designing learning- or assessment-related materials. We should note that colors in cCTt are used mostly to distinguish between different objects and shapes that serve as either obstacles on the grid or triggers for conditionals; for example, there is a black cat, a bourdeaux flower, a red heart, a light blue cloud, a cyan triangle, a yellow star, etc. Contrary to that, the directing arrows, and their related symbols (e.g., numbers, frames), are presented mostly in grayscale. That is, color codes are not fully necessary for the CT-related objects, hence omitting them may not meaningfully harm the measurement of the task performance.

5. Limitations and Further Research

This study is, of course, not without limitations. First, we only measured CT by using a single tool (cCTt), and our findings may be somehow biased due to the unique characteristics of this tool. Second, the analysis was based on students from a single country (Israel), which is characterized by a set of educational, technological, and cultural beliefs and practices that may have biased our findings. Finally, it's important to note that our research population was limited and may not be considered representative. Therefore, it is suggested to replicate this study in other geographically and culturally varied populations, and to use further CT measuring tools, both physical and digital.

6. References

Al-Ayash, A., Kane, R. T., Smith, D., & Green-Armytage, P. (2016). The influence of color on student emotion, heart rate, and performance in learning environments. *Color Research & Application*, *41*(2), 196–205. https://doi.org/10.1002/COL.21949

Amarin, N., & Al-Saleh, A. (2020). The effect of color use in designing instructional aids on learners' academic performance. *Journal of E-Learning and Knowledge Society*, *16*(2), 42–50. https://doi.org/10.20368/1971-8829/1135246

Beddow, P. A. (2018). Cognitive load theory for test design. *Handbook of Accessible Instruction and Testing Practices: Issues, Innovations, and Applications*, 199–211. https://doi.org/10.1007/978-3-319-71126-3_13/COVER

Ben-Haim, E., Cohen, A., & Tabach, M. (2019). *Types of graphic interface design and their role in learning via mathematical applets at the elementary school*. *TWG16*(3). https://hal.science/hal-02422151

Cureton, E. E. (1956). Rank-biserial correlation. *Psychometrika*, *21*(3), 287–290. https://doi.org/10.1007/BF02289138/METRICS

Derefeldt, G., Swartling, T., Berggrund, U., & Bodrogi, P. (2004). Cognitive color. *Color Research & Application*, *29*(1), 7–19. https://doi.org/10.1002/COL.10209

Ekman, H., & Waliullah, F. (2019). *Reducing cognitive load on the working memory by signaling primed colors: Can color improve the learning of mathematics?*. https://www.diva-portal.org/smash/get/diva2:1327786/FULLTEXT02

El-Hamamsy, L., Dehler Zuuerey, J., Martín-Barroso, E., Zapata-Cáceres, M., Marcelino, P., Bruno, B., Román-González, M., Papadakis, S., Korkmaz, O., Leoste, J., Zuuerey, D. J., Copyright, fpsyg, & Zuuerey, D. (2022). *Comparing the psychometric properties of two primary school Computational Thinking (CT) assessments for grades 3 and 4: The Beginners' CT test (BCTt) and the competent CT test (cCTt)*.

El-Hamamsy, L., Zapata-Cáceres, M., Barroso, E. M., Mondada, F., Zufferey, J. D., & Bruno, B. (2022). The

Competent Computational Thinking Test: Development and Validation of an Unplugged Computational Thinking Test for Upper Primary School. *Journal of Educational Computing Research*, *60*(7), 1818–1866. https://doi.org/10.1177/07356331221081753

El-Hamamsy, L., Zapata-Cáceres, M., Marcelino, P., Bruno, B., Dehler Zufferey, J., Martín-Barroso, E., & Román-González, M. (2022). Comparing the psychometric properties of two primary school Computational Thinking (CT) assessments for grades 3 and 4: The Beginners' CT test (BCTt) and the competent CT test (cCTt). *Frontiers in Psychology*, *13*, 1082659. https://doi.org/10.3389/FPSYG.2022.1082659/BIBTEX

Ezeamuzie, N. O., & Leung, J. S. C. (2021). Computational Thinking Through an Empirical Lens: A Systematic Review of Literature. *Https://Doi.Org/10.1177/07356331211033158*, *60*(2), 481–511. https://doi.org/10.1177/07356331211033158

Gillmor, S. C., Poggio, J., Embretson, S., & Embretson, S. ". (2015). Effects of Reducing the Cognitive Load of Mathematics Test Items on Student Performance. *Numeracy*, *8*(1), 4. https://doi.org/http://dx.doi.org/10.5038/1936-4660.8.1.4

Gnambs, T., Appel, M., & Kaspar, K. (2015). The effect of the color red on encoding and retrieval of declarative knowledge. *Learning and Individual Differences*, *42*, 90–96. https://doi.org/10.1016/J.LINDIF.2015.07.017

Honrales, C. C. (2020). Improving Solving Problem Ability with Tower of Hanoi Puzzle. *Solid State Technology*, 204–219. https://www.solidstatetechnology.us/index.php/JSST/article/view/1524

Kastaun, M., Meier, M., Küchemann, S., & Kuhn, J. (2021). Validation of Cognitive Load During Inquiry-Based Learning With Multimedia Scaffolds Using Subjective Measurement and Eye Movements. *Frontiers in Psychology*, *12*, 703857. https://doi.org/10.3389/FPSYG.2021.703857/BIBTEX

Kim, H.-Y. (2013). Statistical notes for clinical researchers: assessing normal distribution (2) using skewness and kurtosis. *Restorative Dentistry & Endodontics*, *38*(1), 52–54. https://doi.org/10.5395/RDE.2013.38.1.52

Kuldas, S., Satyen, L., Ismail, H. N., & Hashim, S. (2014). Greater Cognitive Effort for Better Learning: Tailoring an Instructional Design for Learners with Different Levels of Knowledge and Motivation. *Psychologica Belgica*, *54*(4), 350. https://doi.org/10.5334/PB.AW

Kumi, R., Conway, C. M., Limayem, M., & Goyal, S. (2013). Research article learning in color: How color and affect influence learning outcomes. *IEEE Transactions on Professional Communication*, *56*(1), 2–15. https://doi.org/10.1109/TPC.2012.2208390

Le, Y., Chen, Z., Liu, S., Pang, W., & Deng, C. (2021). Investigating the effectiveness of emotional design principle to attenuate ego depletion effect. *Computers & Education*, *174*, 104311. https://doi.org/10.1016/J.COMPEDU.2021.104311

Liew, T. W., Tan, S. M., Gan, C. L., & Pang, W. M. (2022). Colors and Learner's Gender Evoke Different Emotional and Cognitive Effects in Multimedia Learning. *Human Behavior and Emerging Technologies*, *2022*. https://doi.org/10.1155/2022/1235732

Liu, Y., Ma, W., Guo, X., Lin, X., Wu, C., & Zhu, T. (2021). Impacts of Color Coding on Programming Learning in Multimedia Learning: Moving Toward a Multimodal Methodology. *Frontiers in Psychology*, *12*, 773328. https://doi.org/10.3389/FPSYG.2021.773328/BIBTEX

Mann, H. B., & Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Source: The Annals of Mathematical Statistics*, *18*(1), 50–60.

Ogegbo, A. A., & Ramnarain, U. (2022). A systematic review of computational thinking in science classrooms. *Studies in Science Education*, *58*(2), 203–230. https://doi.org/10.1080/03057267.2021.1963580

Ramos-Nuñez, P., Schaeffer, L., & Catrambone, R. (2018). Problem Solving with Color: Color's Effect on Affect and Problem Solving with Subgoal Labels. *Https://Doi.Org/10.1177/1541931218621268*, *2*, 1166–1170. https://doi.org/10.1177/1541931218621268

Siuda-Krzywicka, K., & Bartolomeo, P. (2020). What Cognitive Neurology Teaches Us about Our Experience of Color. *Neuroscientist*, *26*(3), 252–265. https://doi.org/10.1177/1073858419882621/ASSET/IMAGES/LARGE/10.1177_1073858419882621-FIG3.JPEG

Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity*, *42*, 100926. https://doi.org/10.1016/J.TSC.2021.100926

Wenjie, Z., Liqun, D., & Jinhong, D. (2021). Neural mechanism underlying the effects of object color on episodic memory. *Acta Psychologica Sinica*, *53*(3), 229. https://doi.org/10.3724/SP.J.1041.2021.00229

Research on the Current Status and Pathways for Improving Computational Thinking Levels in Upper Elementary School Students
—Using Selected Representative Elementary Schools in City K as Examples

Yuxi Chen [1],Yang Zhao[2*]
[1,2]School of Information,Yunnan Normal university,Kunming,China
2223100060@ynnu.edu.cn, 42755044@qq.com

ABSTRACT
Computational thinking is a core competency for problem-solving in the digital age. This study focuses on 344 upper elementary school students from four different-level primary schools in City K. Descriptive statistical analyses, including Spearman correlation, differences, and multilevel regression, were conducted to examine the computational thinking levels of upper elementary school students. Additionally, a structural equation model (SEM) was employed to identify the factors influencing their computational thinking levels. The results indicate that the overall computational thinking levels of upper elementary school students in City K are good. Gender, grade level (fifth or sixth grade), internet usage frequency, and attitudes towards information technology courses do not significantly influence computational thinking levels. However, attitudes towards internet usage, programming experience, information technology course content, math scores, and science scores show significant positive correlations with computational thinking levels. Finally, based on the data analysis results and a comparison with the requirements set by the Ministry of Education, recommendations are provided to promote the development of computational thinking in upper elementary school students.

KEYWORDS
Upper elementary school students, computational thinking, current status,influencing factors

1. INTRODUCTION
With the advent of the digital age, the ability to solve problems using computers has become increasingly important. In response, the concept of computational thinking has emerged and is considered as an "attitude and skill that everyone should possess"(The New Media Consortium,2017). Countries and regions such as Europe, the United States, Australia, and Singapore have incorporated the cultivation of computational thinking into their K-12 talent development plans(Angeli C et al,2016). In China, the 2022 revised version of the "Compulsory Education Information Technology Curriculum Standards" designates computational thinking as one of the four core competencies in the field of information technology(Beijing Normal University Press, 2022). It is evident that the cultivation of computational thinking is becoming more focused on younger age groups.Upper elementary school students have a certain level of knowledge and cognitive foundation, making it a critical period for cultivating computational thinking. However, there is currently limited research on the factors that influence computational thinking abilities in primary school students. Therefore, this study utilizes a questionnaire survey to analyze the current status of computational thinking levels among upper elementary school students in City K. It aims to deeply explore and analyze the influencing factors and investigate the correlations between these factors and computational thinking. The study strives to provide valuable insights for the development of computational thinking abilities in upper elementary school students.

2. Conceptual Explanation and Problem Statement

2.1. Definition of Computational Thinking Concept
Definition of Computational Thinking The concept of computational thinking was first elaborated by Professor Jeannette M. Wing of Carnegie Mellon University in 2006, stating that "computational thinking is the process of applying fundamental concepts of computer science to solve problems, design systems, and understand human behavior"(Wing J M,2006). In 2011, Professor Wing further summarized computational thinking as "a thinking process that can be effectively executed by information processing agents"(Liu m&Zhang Q,2018), indicating a shift from emphasizing operational skills to focusing on thinking processes. Subsequently, more scholars from both domestic and international contexts have joined the research on computational thinking. For example, the International Society for Technology in Education (ISTE) in the United States proposed in 2015 that computational thinking is a set of mental tools that effectively combines digital technology with human thinking to solve complex real-world problems(Liu m&Zhang Q,2018). In the 2022 revised version of the "Compulsory Education Information Technology Curriculum Standards" in China, computational thinking is defined as the use of thinking methods from the field of computer science, involving activities such as abstraction, decomposition, modeling, and algorithmic thinking in the process of problem-solving.(Beijing Normal University Press, 2022) Based on the above, this study defines computational thinking as a series of thinking processes centered around problem-solving, including dimensions such as problem-solving ability, creativity, critical thinking, algorithmic thinking, and collaboration skills.

2.2. Influencing Factors
In terms of inherent individual attributes, grade level and gender are the first two variables that should be considered. Existing research has yielded conflicting results regarding the impact of these two factors on computational thinking. For example,(Atmatzidou S&Demetriadis S,2016)found no significant correlation between grade level, gender, and computational thinking. Similarly, (Crews T&Butterfield

J,2013)suggested that girls need to invest more time to achieve computational thinking levels equivalent to boys, indicating that the influence of grade level and gender on computational thinking levels remains inconclusive.

In terms of external factors, with the widespread use and development of the internet, students' attitudes and frequency of internet usage for learning and entertainment can also influence their problem-solving thinking processes. Programming learning plays a role in the development of students' abstraction and algorithmic thinking abilities, making programming skills a potential influencing factor on students' computational thinking levels. The primary school information technology curriculum is an important subject for cultivating students' computational thinking abilities. Therefore, students' satisfaction with the information technology curriculum and the novelty of the curriculum may also be factors influencing computational thinking levels [8]. Subjects such as mathematics and science also revolve around problem-solving, and students' computational thinking levels may change during the learning process. However, there is limited research exploring the impact of other subject grades on computational thinking abilities.

Based on the above, this study aims to investigate the influence of grade level, gender, attitudes and frequency of internet usage, programming experience, attitudes and content of information technology courses, as well as mathematics and science grades on computational thinking levels.

3. Research Design and Procedure

3.1. Research Instruments

In the assessment of computational thinking levels, this study utilized the "Computational Thinking Scale for Elementary School Students" developed by Zhang(Zhang Y et al,2020). The scale consists of 23 items and can be administered in both programming and non-programming environments. To better align with the reading habits of upper elementary school students, the language expressions of some items in the scale were optimized. In terms of reliability, the overall Cronbach's Alpha coefficient and split-half reliability coefficient of the scale were 0.890 and 0.858, respectively, both exceeding 0.8, indicating high reliability. In terms of validity, the KMO value of the scale was 0.896, indicating strong correlations among the items and good questionnaire validity.For the factors influencing computational thinking, the grade and gender of the participants were recorded as unordered categorical variables. The attitudes and frequency of internet usage, programming experience, attitudes and content of information technology courses, as well as mathematics and science grades were recorded as ordered categorical variables. Attitudes were assessed using a Likert five-point scale, ranging from negative to positive with scores of 1-5. Frequency and programming experience were scored based on the duration of time. Grades were scored in ascending order within the specified range.

3.2. Participant Selection

To ensure a good level of coverage in the research data, this study selected three primary schools in City K with different quality levels. A total of 360 students from the fifth and sixth grades were randomly chosen as participants. The research team distributed 360 questionnaires on computational thinking to these upper elementary school students, and successfully collected all 360 questionnaires, resulting in a response rate of 100%. Among them, 16 questionnaires were deemed invalid, leaving a total of 344 valid questionnaires, with an effective response rate of 95.5%. The participants included 179 fifth-grade students (52% of the sample) and 165 sixth-grade students (48% of the sample). The gender distribution consisted of 169 male students (49.1%) and 175 female students (50.9%).

3.3. Data Processing

The self-assessment scale for computational thinking adopts a Likert-type five-point distribution: ranging from "completely not applicable" to "extremely applicable," with scores of 1-5 assigned accordingly. The scale consists of 23 items, with a total score of 115. To facilitate the research, the overall score for computational thinking is converted into a percentage score (calculation method: actual score/115*100). Scores below 60 are considered as failing, 60-79 as passing, 80-89 as good, and 90-100 as excellent. The obtained data were analyzed using SPSS 27.0 for descriptive statistics, differences, correlations, and multiple regression analysis. Additionally, Amos 26.0 software was used to construct a structural equation model to explain and predict the relationships between various factors and computational thinking levels. This further facilitated the construction of a relationship model between the factors and the development of computational thinking.

4. Research Findings

4.1. Descriptive Statistical Analysis of Computational Thinking

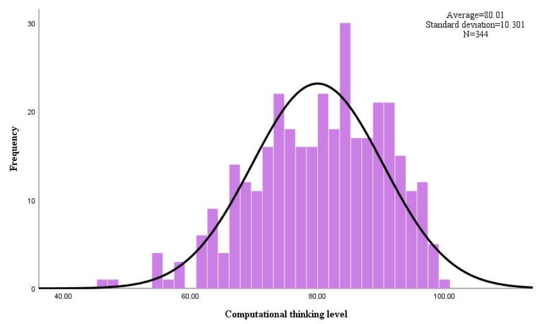


Figure 1. Histogram of the overall scores of computational thinking.

In the overall sample, the lowest score for computational thinking among upper elementary school students was 46.33, the highest score was 98.67, and the mean score was 80.01 (SD=10.301). From the normal P-P plot, it can be observed that the distribution of the sample data does not perfectly follow a normal distribution. Therefore, the median score (80.83) was chosen to represent the overall situation. The median indicates that the collected sample group has a good level of computational thinking, but there is still room for improvement. The histogram in Figure 1 shows that most students' computational thinking scores fall within the range of 75-85, which is above the passing score of 60.

The scale used to measure the computational thinking abilities of upper elementary school students adopts a Likert 5-point scoring method, with a theoretical midpoint of 3. After statistical analysis, the overall average score for computational thinking abilities in the sample data was found to be 3.99, which is greater than 3 (as shown in Table 1). This indicates that upper elementary school students, as a whole, possess computational thinking abilities that are slightly above average.From Table 1, it can be observed that the factor levels for the five dimensions of computational thinking are as follows: creativity > critical thinking > collaboration > problem-solving > algorithmic thinking. Among these dimensions, creativity, critical thinking, problem-solving, and algorithmic thinking exhibit a left-skewed peaked distribution, with the mean values underestimating the central tendency. On the other hand, the collaboration dimension shows a right-skewed peaked distribution, indicating a tendency for overestimation in the mean score for collaboration. Overall, the dispersion trend is relatively weak.

Table 1. Current levels of CT in each dimension.

Dimension	MV	SD	Skewness	PV
Creativity	4.11	0.58	-0.47	-0.55
Critical thinking	4.09	0.66	-0.62	-0.20
Problem solving	3.92	0.64	-0.44	-0.20
Algorithmic thinking	3.77	0.78	-0.24	-0.76
Collaboration	4.00	0.79	-0.94	0.69
Total	3.96	0.52	-0.47	-0.15

4.2. Analysis of Differences in Computational Thinking Levels

4.2.1. Independent Samples t-test

The computational thinking levels of students may be influenced by individual factors. Independent samples t-tests were conducted to compare the differences in computational thinking levels based on grade and gender (a binary variable).In terms of grade distribution, the overall computational thinking level of fifth-grade students (M=78.97±10.32) was slightly lower than that of sixth-grade students (M=80.85±10.28), but the difference was not significant (t=-1.68, p=0.093). However, in terms of specific dimensions, sixth-grade students showed significantly higher creativity levels compared to fifth-grade students (t=-2.10, p=0.037).Regarding gender, there was a small difference in the overall computational thinking level between male students (M=79.90±10.65) and female students (M=79.85±10.03), but this difference was not significant (t=0.048, p=0.96). No significant differences were observed in the specific dimensions between genders.Overall, the analysis suggests that there are no significant differences in computational thinking levels based on grade or gender, except for a higher level of creativity in sixth-grade students compared to fifth-grade students.

4.2.2. One-Way ANOVA test

One-way ANOVA tests were conducted to examine the differences in computational thinking levels among students based on the following influencing factors: attitudes and frequency of internet usage, programming experience, attitudes and content of information technology courses, and mathematics and science grades (all variables with three or more levels).The results showed significant differences in computational thinking levels among students with different attitudes towards internet usage (F=5.15, p=0.002), programming experience (F=3.13, p=0.004), attitudes towards information technology courses (F=3.59, p=0.007), content of information technology courses (F=6.12, p<0.001), mathematics grades (F=9.73, p<0.001), and science grades (F=9.26, p<0.001).

Specifically, in terms of attitudes towards internet usage, students who considered internet usage for learning as moderately important (M=77.47±10.74) had significantly lower computational thinking levels compared to those who considered it as relatively important (M=80.71±10.11) or very important (M=82.92±8.23).

Regarding programming experience, students who had learned programming for more than two years through various methods (M=81.79±17.64) had significantly higher computational thinking levels compared to those who had no exposure to programming courses (M=72.89±15.37).

In relation to information technology courses, students who highly enjoyed the courses (M=81.20±10.60) had significantly higher computational thinking levels compared to those who did not enjoy the courses (M=76.10±14.31). Students who perceived the content of information technology courses as uninteresting and outdated (M=75.92±11.17) had significantly lower computational thinking levels compared to those who found the content interesting and up-to-date (M=82.25±9.69).

Regarding academic performance, students who scored above 90 in their most recent mathematics exam (M=82.75±10.04) had significantly higher computational thinking levels compared to students in any other score range, such as 70-80 (M=72.87±8.30). Similarly, students who scored above 90 in their most recent science exam (M=81.79±10.18) had significantly higher computational thinking levels compared to students in the 60-70 (M=70.99±10.33) and 70-80 (M=77.14±9.48) score ranges.

4.3. Correlation Analysis between Variables and CT Levels

Spearman's correlation coefficient was used to explore the correlation between two types of variables. The analysis results are shown in Table 2.In the sample data, gender and internet usage frequency did not show a significant correlation with students' computational thinking levels. However, grade, programming experience, attitudes towards internet usage for learning, attitudes and content of information technology courses, and mathematics and science grades were found to be significantly correlated with computational thinking levels.The correlation coefficients (r values) ranged from small (r=0.107) to moderate (r=0.329), indicating varying degrees of correlation. Moreover, all variables that showed a correlation with computational thinking had positive r values greater than 0, indicating a positive linear relationship with computational thinking levels.

Table 2. Spearman's correlation analysis between variables.

Variables	1	2	3	4	5	6	7	8	9	10
1.Grade	—	-0.011	.206**	.194**	0.057	.189**	.126*	.170**	.430**	.107*
2.Gender		—	-.196**	-0.089	0.048	-.108*	0.003	-0.066	0.005	-002
3.Programming experience			—	.146**	0.089	.143**	0.103	.135*	.245**	.165**
4.Attitudes towards internet usage for learning				—	.273**	.229**	.130*	0.093	.119*	.186**
5.Frequency of internet usage					—	0.023	-0.048	0.029	0.072	0.008
6.Attitudes towards IT courses						—	.500**	0.095	.157**	.191**
7.Content of IT courses							—	0.02	0.074	.231**
8.Mathematics grades								—	.426**	.329**
9.Science grades									—	.252**
10.Total score of computational thinking										—

4.4. Multiple Regression Analysis of Variables and CT Levels

After conducting the correlation analysis, further investigation of the causal relationship between variables and computational thinking levels requires regression analysis to identify the influencing factors of computational thinking levels. Based on the results of the correlation analysis, it was found that grade and frequency of internet usage did not have a significant correlation with computational thinking levels. Therefore, these two factors were treated as control variables, while other potential influencing factors were considered as core independent variables.Model 1 represents the impact of control variables (gender and internet usage frequency) on the dependent variable (computational thinking levels), while Model 2 includes the core independent variables in addition to the control variables. The F-value for Model 2 is 9.060, which is significant at the $p<0.001$ level, indicating the effectiveness of Model 2. The results of the multiple regression analysis further confirm that attitudes towards internet usage for learning ($\beta=1.928$, $p<0.01$), programming experience ($\beta=1.827$, $p<0.01$), content of information technology courses ($\beta=2.159$, $p<0.001$), mathematics grades ($\beta=2.362$, $p<0.001$), and science grades ($\beta=1.830$, $p<0.001$) have a positive impact on computational thinking levels. However, gender, grade (fifth or sixth), frequency of internet usage, and attitudes towards information technology courses do not have a significant impact on computational thinking levels.

4.5. Construction and Analysis of Structural Equation Model (SEM)

*Table 3.*Model Fit Test

	X^2/df	GFI	CFI	NFI	TLI	RMSEA
Model	2.315	.967	.921	.875	.858	0.062
Standard	1-3	>0.8	>0.8	>0.8	>0.8	<0.08

Using Amos, a structural equation model was constructed to examine the relationships between variables and reveal the structural relationships through path coefficients(Sun L,&Hu L,2021). First, a model fit index test was conducted, and the results are shown in Table 3. Comparing the values with the standard values, it indicates that the model has a good fit.

The results of the model path coefficients, as shown in Figure 2 and Table 4, indicate that the relationships between grade, gender, frequency of internet usage, and attitudes towards information technology courses with computational thinking levels did not pass the significance test. The path coefficients between attitudes towards internet usage for learning, programming experience, content of information technology courses, mathematics grades, and science grades with computational thinking levels varied within the range of 0.14 to 0.25. The validation of the structural equation model further confirms the results of the multiple regression analysis, indicating that in the sample data, students' attitudes towards internet usage for learning, programming experience, content of information technology courses, and mathematics and science grades have an impact on computational thinking levels. However, gender, grade (limited to upper elementary school in this study), frequency of internet usage, and attitudes towards information technology courses do not have an impact on computational thinking levels.

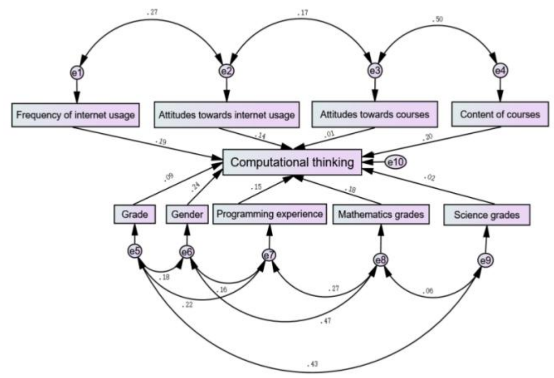


*Figure 2.*SEM Path Coefficient Diagram.

Table 4. Hypothesis testing

Hypothesis	Path Coefficient	Acceptance
Attitudes towards internet usage→CT	$.140^{**}$	Accepted
Programming experience→CT	$.151^{**}$	Accepted
Content of IT courses→CT	$.200^{***}$	Accepted
Mathematics grades→CT	$.235^{***}$	Accepted
Science grades→CT	$.181^{**}$	Accepted

5. Conclusion and Recommendations

5.1. Research conclusions

The statistical results indicate that upper elementary school students have a generally good level of computational thinking. In terms of sub-dimensions, students have the highest level of creativity, while problem-solving ability and algorithmic thinking are relatively weaker. Within the fifth and sixth grades and between genders, there is no significant impact on students' computational thinking levels. In various sub-dimensions, girls have slightly higher levels of creativity, critical thinking, and collaboration abilities compared to boys. However, their problem-solving abilities and algorithmic thinking are slightly lower than those of boys, but these differences are not statistically significant. Students' attitudes towards internet usage have a significant positive impact on their computational thinking levels. Students with programming experience have significantly higher computational thinking levels than those without, especially in the sub-dimensions of problem-solving ability and algorithmic thinking. The attractiveness and novelty of information technology course content have a significant positive impact on computational thinking levels.

5.2. Research Recommendations

China started relatively late in cultivating computational thinking abilities among K-12 students. It was only in 2017 that it was first included in the information technology curriculum standards and gradually integrated into classroom teaching(Bai X&Gu X,2019). With the implementation of the Education Informatization 2.0 initiative, interdisciplinary STEAM education, programming instruction, and the widespread use of intelligent teaching environments have contributed to the development of students' computational thinking. Upper elementary school students are in a transitional stage from concrete operations to formal operations(Huang R&Liu X,2016). They can classify, sort, and engage in rule-based games, but their abstract logical thinking, hypothetical thinking, and analogical reasoning abilities are still in the early stages of development. Currently, teaching in various subjects in primary schools mainly focuses on imparting declarative knowledge, and the main focus of the information technology curriculum, which is the main field for cultivating computational thinking, is on operational skills(Wang X,2022). This is not conducive to students applying knowledge to solve complex real-life problems. Therefore, in future teaching, attention should be given to creating suitable environments, guiding students in problem decomposition, and gradually developing abstract thinking through concrete examples, in order to comprehensively enhance students' computational thinking abilities.

Learners in the fifth and sixth grades of primary school have a high degree of similarity in terms of age characteristics, cognitive development, and existing knowledge foundation. Therefore, for the development of computational thinking levels in upper elementary school students, it is important to formulate continuous and spiral training programs. In daily teaching activities, family education, and social discourse, there should be no differentiation between boys and girls. Equal learning resources should be provided, and both boys and girls should be encouraged to actively participate in computational thinking-related activities and projects, fostering confidence and providing personalized and diverse learning approaches.

With the advent of the information age, the internet, as a technology, medium, and cognitive tool, has an impact on the cognitive development and academic achievement of primary and secondary school students(Chen J et al,2018). When students approach internet learning with a positive attitude, they become more open to accepting new information and perspectives, which helps foster creativity and problem-solving abilities. Upper elementary school students already possess certain information retrieval and filtering skills, and engaging in online learning and following social trends on the internet can promote the development of their computational thinking. However, schools and parents should also pay attention to positive guidance. Upper elementary school students tend to process information at a superficial level(Chen P,2022). Without proper supervision, they may use the internet for entertainment or rely on it for learning, which is not conducive to the development of their independent thinking abilities.

Students with more than two years of programming experience performed well in various sub-dimensions, while students with 8-12 months of learning experience had higher levels of computational thinking compared to those with 1-2 years of learning experience. This may reflect the current focus on programming learning for competition purposes(Xie Z et al,2019), where teachers and parents prioritize the acquisition of knowledge and skills, overlooking the cultivation of computational thinking. Research by Özcan(Özcan M Ş et al,2021) suggests that programming education has a positive impact on the development of computational thinking, albeit a relatively small one. Therefore, further exploration is needed to understand how the design of programming courses and the process of programming learning influence the development of computational thinking levels.

The information technology curriculum covers knowledge and skills related to computer hardware and software, programming, and data processing, which are of significant importance in cultivating and enhancing students' computational thinking abilities(Xu X&Dang B,2023). The attractiveness and novelty of the information technology curriculum can stimulate interest in learning, provide the ability to translate abstract knowledge into concrete operations, inspire creativity, and promote sustained learning. When the curriculum content is engaging, students are more likely to actively participate in learning and maintain their motivation to learn.

Information technology is not the only subject that cultivates computational thinking(Sun L&Hu L,2021). Mathematics and science performance can predict computational thinking levels. Additionally, the improvement of computational thinking plays an important role in common skills used in mathematics and science. The essence of computational thinking lies in its interdisciplinary integration with problem-solving at its core. Therefore, teachers can create scenarios, adopt problem-based approaches, and utilize teaching methods such as cooperative learning and brainstorming in primary schools. Encouraging self-assessment and peer assessment can also be combined in the evaluation process.

6. REFERENCES

Angeli C, Voogt J, et al.(2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge . *Educational Technology & Society*, *19(3):* 47-57.

Atmatzidou S, Demetriadis S.(2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences . *Robotics & Autonomous Systems*, *75(1):* 661-670.

Bai X, Gu X.(2019). Construction and application of computational thinking assessment tools for K-12 students. *China Educational Technology*, *2019(10):* 83-90.

Crews T, Butterfield J. (2013).Gender differences in beginning programming: an empirical study on improving performance parity. *Campus Wide Information Systems*, *20(5):* 186-192.

Chen P. (2022).*Research on the current situation and influencing factors of computational thinking level of primary school students*.Wuhan, Huazhong University of Science and Technology.

Chen J, Yan J, Gao L.(2018). Research on the relationship between internet usage preferences and academic performance of primary and secondary school students. *China Health Education*, *34(04):* 347-351.

Huang R, Liu X. (2016).Maker education and the cultivation of students' innovation ability. *Modern Educational Technology*, *26(04):* 12-19.

Hill C, Corbett C, et al. (2020).*Why so few? Women in science, technology, engineering, and mathematics.* https://files.eric.ed.gov/fulltext/ED509653.pdf

Liu M, Zhang Q.(2018). Research progress on computational thinking education abroad. *Open Education Research, 24(1):* 41-53.

Ministry of Education of the People's Republic of China. (2022).I*nformation Technology Curriculum Standards for Compulsory Education (2022 Edition).* Beijing:Beijing Normal University Press.

Özcan M Ş, Çetinkaya E, Göksun T, et al. (2021).Does learning to code influence cognitive skills of elementary school children? Findings from a randomized experiment. *British Journal of Educational Psychology*, *91(4):* 1434-1455.

Sun L, Hu L. (2021).What influences elementary students' computational thinking: A cross-sectional study based on five provinces and cities. *China Educational Technology, 2021(07):* 52-61.

The New Media Consortium. (2017).*The NMC/CoSN Horizon Report: 2017 K-12 Edition*.

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

Wang X. (2022).STEM education innovation and practice: A study on the construction of robotics curriculum in primary and secondary schools. *Educational Research, 2022(11):* 91-95.

Xie Z, Cao Y, Li Y. (2019).Exploration of the content design of artificial intelligence courses in primary and secondary schools. *China Educational Technology*, *2019(04):* 17-22.

Xu X, Dang B.(2023). Integrated learning design model for information technology subject units targeting core competencies. *Modern Educational Technology, 33(06):* 73-81.

Zhang Y, Mo W, Zhang Y, et al.(2023). Development and application of a computational thinking scale for primary school students in China. *China Educational Technology, 2020(10):* 49-57.

Teachers as Architects of Multi-Level CT Experiences: A Phenomenological Exploration

Hangling YU[1*], Jinbao ZHANG[2]
[1,2] Beijing Normal University, China
202221010182@mail.bnu.edu.cn, zhangjb@bnu.edu.cn

ABSTRACT

Computational thinking (CT), as the core quality of information technology in China, has been highly concerned, but there are still ambiguities and controversies in computational thinking education (CTE). Teachers are the designers and participants in the class, and they have rich experience in CTE. From the perspective of teachers, we can find the commonness, essence and significance of CTE in primary schools, and provide new perspectives and suggestions for the development of CTE. Therefore, this study conducts in-depth interviews with teachers and adopts the phenomenological method to explore how teachers understand and practice CTE, discover its theme and reveal its significance. It is found that "creating multi-level experience of CT for students" is the common theme of teachers in the practice of CTE in primary schools. Teachers aim to gain participation, recognition and interest by designing experience activities, and finally cultivate students' computational thinking. This study holds that "multi-level experience" reflects the concern for students' pondering space and emotional experience in primary school CTE, and reflects the special role of teachers in creating experience environment and maintaining internal motivation for students in primary school CTE.

KEYWORDS

computational thinking education, phenomenological method, primary school, experience, teacher

1. INTRODUCTION

Computational thinking is highly valued in both research and teaching practice. However, at present, computational thinking still faces several challenges, including a vague connotation, unclear meaning (Li & Yang, 2023), and a disconnect between theory and practice (Zhang, 2019), etc. Many related research discusses the concept, connotation, educational content, and teaching methods of CT from the perspective of what it "ought to be". Nevertheless, in actual school classrooms, teachers must carry out appropriate CTE tailored to specific student characteristics, teaching conditions and needs. The methods and strategies proposed in literature research are often challenging to learn from and apply (Zhang, 2019).

Computational thinking, as the core literacy of information technology in compulsory education in China, has generated rich practical experience in classroom teaching. As a crucial role of teaching activities, teachers not only play a key role in implementing CT but also possess an intuitive understanding and experience of CTE. By adopting a teacher-centric perspective and utilizing phenomenological methods, we can explore how teachers carry out CTE in information technology courses. This approach can unearth commonalities and characteristics in teachers' practices across different teaching situations, can reflect the unique features and significance of CTE. Moreover, this exploration can help determine the essence and importance of what computational thinking "is". Such insights can provide enlightenment and a new perspective for research in CTE.

2. LITERATURE REVIEW

Phenomenology insists on the attitude of "going back to the thing itself" and emphasizes the significance of people's intuitive experiences. It takes people's experience and consciousness as the method for uncovering the essence and meaning of things. Phenomenology holds that life experiences have a definite essence, which can be clearly identified through introspection. Furthermore, life experience has the characteristics of intersubjectivity, which can transcend individuals and obtain general characteristics. Unlike empirical positivism, which focuses on discovering rules to control behavior, phenomenology obtains profound significance and essential characteristics of something by describing and explaining its experience (Loren, et.al., 2010, 49). In the field of education, educational phenomenology centers on educational experience. It adopts the perspective of pre-reflection and pre-theory to explore and question the unique behavior and experience of individuals in specific educational situations. From these experiences, we can get some useful reflections on the teaching content, and explore the teaching wit and significance (Li, 2005).

In the field of computational thinking education, there have been numerous studies and analyses on the key elements and compositional structure of computational thinking. The most representative ones include the five elements of CT (Selby & Woollard, 2013) and the three-dimensional framework of CT (Brennan, et.al., 2012). Many studies have demonstrated the role of specific teaching content or teaching modes in cultivating students' CT from the perspective of scientific demonstration. However, limited attention has been given to teachers' experiences with CTE in actual classroom settings and the educational wisdom and significance it reflects.

Scientism-oriented research often treats "computational thinking" as an objective and scientific object. However, education is a complex system involving multiple subjects, and its significance can only be revealed through diverse and situational experiences related to CT. Shi (2018) have explored the characteristics of children's experiences in programming and their educational significance through grounded research. Additionally, some scholars have delved into the teaching concepts of computational thinking held by different types of teachers using the phenomenological method. These studies are of great significance in promoting the teaching practice of computational thinking. However, generally speaking, research on CTE from the perspective of practical experience is still in its early stages. The purpose of phenomenology is to clarify and find the common core of meaning (Van, 2003, 23). Studying teachers who have experienced and felt computational thinking education through phenomenological methods, while paying attention to teachers' behaviors and experiences, will help discover how computational thinking is understood by teachers and transmitted to students in real teaching. This approach aids in a thorough understanding of the characteristics, essence, and significance of computational thinking education. Perhaps only through the life world and practical experiences can we genuinely grasp the educational meaning of computational thinking.

3. RESEARCH DESIGN

3.1. Research Questions

Actually, teachers make different choices based on students' situations, the teaching context, and the characteristics of the teaching content. They carry out personalized classroom designs to introduce diverse teaching activities to students. The choices and designs made by teachers reflect their understanding of computational thinking teaching in primary schools, providing insights into the characteristics and connotations of computational thinking teaching in primary schools. This study focuses on how teachers bring computational thinking experiences to primary school students and explores their characteristics and significance. Specifically, the research addresses the following three issues:

Q1. What experiences do teachers have regarding computational thinking education?

Q2. What are the characteristics and commonalities of teachers' experiences in computational thinking education?

Q3. What structural features does this commonality reflect in the education of computational thinking in primary schools, and what is its educational significance?

3.2. Research Methods

Researchers conducted in-depth interviews to gather teachers' descriptions of their teaching practices and experiences in computational thinking. Guided by the phenomenological concept, the textual data of these experiences were processed and analyzed using a combination of Van Manen's phenomenology of education and Giorgi's phenomenology of experience. In the actual implementation, the study referred to the explanation and refinement of Giorgi's method by scholar Lee (2009) and also drew on Colaizzi's (1978) framework for theme extraction. This allowed the identification of themes related to primary school teachers' experiences in computational thinking teaching practices and the revelation of their significance. The specific steps are illustrated in the figure below.

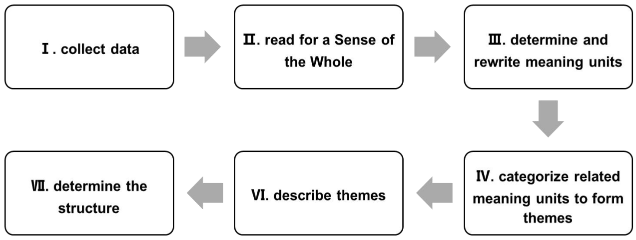


Figure 1. Specific Operation Steps in This Study.

3.3. Study Participants

On the one hand, we need to consider whether the selected research subjects can provide experiential information related to the studied phenomenon. On the other hand, the research subjects should be accessible. To gain insights into the design, implementation process, and experiences of computational thinking teaching, this study followed the purposive sampling method and selected four teachers who clearly expressed their experiences in computational thinking teaching as the research subjects. These four teachers come from different regions, with varying genders and teaching experiences. They possess unique understandings and concepts of computational thinking, and the specific situations in teaching computational thinking also vary. Through interviews with these teachers, we aim to capture more representative and diverse practical experiences in computational thinking teaching and identify commonalities within the differences. The specific information about the research subjects is presented in the following figure.

Teacher	Level	CT in Teachers' Understanding	Representative CTE content	Methods of CTE
YB	Grade 1-2	The essence of computational thinking is algorithm, a strategic mechanism to solve problems.	Algorithm theory (sorting algorithm) Innovative production (multimedia)	Through the "queue formation" sports activities, experience the process of computer sorting.
YZ	Grade 5	Thinking activities such as abstraction, decomposition, modeling and algorithm involved in the process of solving problems.	Basic concept (abstract shortest path)	By drawing the shortest path from the school library to the classroom, we can experience and discover the essence of "abstraction" and apply this method to solve problems in life.
WS	Grade 5	Think like a computer scientist.	Programming Design (stage role design in scratch)	Think like a computer scientist through role-playing simulation.
LY	Grade 5-6	Thinking activities such as abstraction, decomposition, modeling and algorithm involved in the process of solving problems.	Programming design (Four operations) Innovative Production (IoT)	Solve problems with computers through the cycle experience of "finding problems-designing algorithms-programming verification-reflection and modification".

Figure 2. Participants in This Study.

3.4. Data Processing and Analysis

Researchers conducted one-on-one in-depth interviews with four typical and representative primary school information technology teachers, focusing on the theme of the "typical implementation process and experience of computational thinking teaching". After the interviews, the researcher repeatedly read the collected interview texts. On this basis, the research rewrote the key paragraphs and extracted the meaning units (MU).

Teacher	A Brief Description of Computational Thinking Teaching Practice	Key Meaning Unit
YB	I don't want to tell the students what the sorting algorithm is, but I hope they can experience the algorithm, so I designed an activity // (YB-1-MU1) about the sorting algorithm. First, I told the students that the rules of the game are that as long as I shout "queue formation", two adjacent students will compare each other and then exchange orders, and finally they will be arranged in the order from high to low from chaotic order // (YB-2-MU2). It's hard for students to understand why queuing has become so complicated, so I will tell them, "This is a step for computers to deal with problems, and that's how computers sort them..." After talking, they are still quite cooperative // (YB-3-MU3). This whole process actually takes a long time, but I will find that students are more likely to accept the knowledge of sorting algorithm // (YB-4-MU4). Before class, they didn't know the sorting algorithm at all, but after class, I asked them to review activities // (YB-5-MU5) and answer how many times to compare from an unordered sequence computer to an ordered sequence computer, they had an epiphany //(YB-6-MU6). I hope I can provide a good space and environment for students, and students can be interested in my class. I hope they can actively use computers to create their own works, think and do it themselves, so as to cultivate their computational thinking //(YB-7-MU7).	YB-1-MU1：Hope that students can get intuitive experience and feelings about the algorithm in the activity. YB-2-MU2：Making rules of the game enables students to do specific actions according to instructions. YB-3-MU3：Help students understand that the meaning of action is related to computers. YB-4-MU4：Activities make it easier for students to accept the knowledge content of computer sorting algorithm. YB-5-MU5：Guide the students to review the experience activities before thinking about the problems. YB-6-MU6：After reflection, students can gain a new understanding of the relevant content, better understand the key content of the ranking algorithm and successfully complete the test. YB-7-MU7：Arouse students' interest and then develop computational thinking.
YZ	Teaching abstraction to primary school students relies heavily on experience. In the course of abstraction, I took a real map of the campus for students to abstract and asked them to draw the shortest path from the warehouse to the classroom // (YZ-1-MU1). I will find that the pictures drawn by the students are uneven //(YZ-2-MU2). After the students finish painting for the first time, I will show some representative works to the physical booth in the classroom, so that they can observe and compare them first. Then I will ask the students how we will show the way if a stranger comes to school, so that some students will react that the painting just now is wrong // (YZ-3-MU3). Children's abstract ability is not strong, and they are usually exposed to concrete things. Therefore, we should start from familiar scenes in life, let them experience them step by step, guide them to reflect and modify after painting, and finally they can understand that abstraction should delete unnecessary content and keep necessary content //(YZ-4- Unit 4). I hope that after this class, students can also use abstract methods to simplify many complicated problems in their lives, which will be helpful to their lives //(YZ-5- Unit 5). In addition, I will expand it at the end of the class. For example, this simplified diagram with a combination of points and lines is actually related to graph theory, and I will also choose something interesting like "Seven Bridges Problem", which students are also very interested in //(YZ-6- Unit 6).	YZ-1-MU1: Teaching about abstraction starts with students' familiar learning and life situations. YZ-2-MU2: Drawing is a process of students' abstraction, which can reflect students' different understanding of abstraction. YZ-3-MU3: Guide students to reflect and improve their abstract process and results by observing and asking questions. YZ-4-MU4: Guide students to gain an essential understanding of abstraction in action. YZ-5-MU5: Hope that students can migrate and apply abstract methods and ideas to solve real-life problems. YZ-6-MU6: The related knowledge of computer science will be told to students as an expanding content, which will stimulate students' interest in further study and exploration of information technology disciplines.
LY	I choose programming to realize four operations as the main content of the course. The program itself is not difficult, so students can spend more time on algorithm design //(LY-1-MU1). In addition, this is a familiar problem for students, so they can easily abstract the rules and then realize programming //(LY-2-MU2). First of all, I started with the simplest "addition", took the students to define and analyze the problem, abstracted the key elements, established the first model to solve the problem, then designed the algorithm, and finally verified it by programming. After using these steps once to solve the problem and students have a preliminary experience //(LY-3-MU3), let students analyze "subtraction problem", build the model-design algorithm-program verification for the new problem again, and execute this cycle. This cycle will be executed at least 2-3 times in class //(LY-4-MU4). Actually, I didn't do this in a class. This method of "building model-designing algorithm-programming verification" runs through programming teaching for a long time. After repeated many times, students can get some experience and find out the advantages of this method and the advantages of solving problems with computers, so that the next time they encounter problems, they will unconsciously want to use this method to design algorithms for computers to solve problems //(LY-5-MU5). After a period of training in analyzing and solving problems, I found that the students' programming ability was greatly improved, which also verified the improvement of their computing thinking //(LY-6-MU6).	LY-1-MU1: Computational thinking teaching should pay more attention to algorithm design and use programming as a tool to test algorithms. LY-2-MU2: Solve familiar problems with computers. LY-3-MU3: Take students through the process from analysis to design to inspection, so that students can get a preliminary experience of this problem-solving method. LY-4-MU4: Strengthen students' understanding of this problem-solving method through independent practice and repetition. LY-5-MU5: Continuous training and related experience make students realize the advantages of using computers to design algorithms and solve problems. LY-6-MU6: After a period of time, students' computational thinking is expressed through programming level.
WS	My idea in this class is to show the final work to students first, and then solve the problems one by one by computer to cultivate their computational thinking. // (WS-1- MU1) But the teacher commented that my class didn't realize the cultivation of computational thinking. I later reflected that it was really that I replaced the students' thinking process with my own process of decomposing problems. //(WS-2-MU2) Students actually have no relevant real life experience and no knowledge base, so they can't abstract the key to the problem smoothly and then realize it step by step like me.// (WS-3-MU3) Later, I also asked students to role play, pretend to be computer scientists, or pretend to be parking lot technical administrators, etc. to design the system, but in fact, these experiences are far away from the students, so they have no interest and it is difficult to cultivate their computational thinking through such activities so that they can really think like computer scientists. //(WS-4-MU4) So up to now, I still think it is very difficult to cultivate students' computational thinking.	WS-1-MU1: WS thinks that the reasonable design of computational thinking teaching. WS-2-MU2: Teachers use their own experience instead of students' experience and thinking process, which is not conducive to the cultivation of computational thinking. WS-3-MU3: Teaching activities need to be carried out on the basis of students' real experience and experience. WS-4-MU4: Arouse students' interest and curiosity to stimulate students to think actively.

Figure 3. Rewrite and Determine Meaning Units.

The study initially categorizes the meanings associated with semantic words as potential categories. It then conducts a screening process, excluding repeated meaning categories or merging them. Subsequently, each category is defined. Finally, the theme is determined to obtain a description of the general structure of computational thinking teaching in primary schools. The study reveals that the term "experience" holds a rich structure and profound connotation in teachers' descriptions of computational thinking teaching practices (as presented in the following figure).

Theme	Dimension	Description of Meaning Units (3-5 items are extracted for each category)
teachers create "*multi-level computational thinking experience*" for students	body-based CT experience	YB-2: performing specific actions YZ-1: drawing as required LY-3: executing specific steps WS-3: familiar life experience
	cognition-based CT experience	YB-3: understanding the significance behind actions YB-5: reflecting after reviewing the experiential process YZ-3: observing, comparing, and reflecting for improvement YZ-4: gaining essential understanding through experience LY-4: independently attempting to enhance understanding
	emotion-based CT experience	YB-7: creating a positive teaching atmosphere and environment YZ-6: novel experiences captivate student interest LY-5: sense of achievement in experience leads to method acceptance
	evaluation of experience	WS-4: mobilizing active participation and independent thinking YB-4: students are more receptive to knowledge YB-6: acquiring new understanding and experiences YZ-5: transferring to real-life issues LY-6: reflected in subsequent works

Figure 4. Meaning Units and Core Theme.

4. MULTI-LEVEL CT EXPERIENCE: RESULTS FROM PHENOMENOLOGICAL ANALYSIS

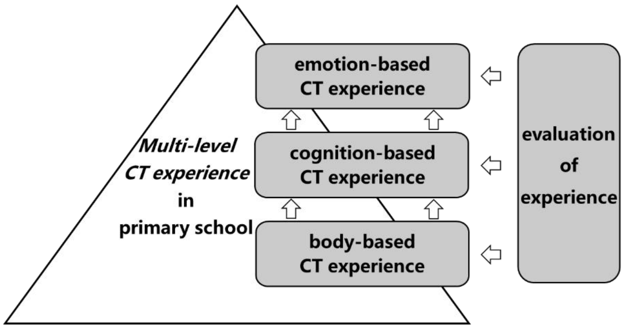


Figure 5. Multi-level CT Experience in Primary School.

4.1. Body-based CT Experience

"Experience" begins with a hands-on approach. When describing typical CTE activities they have conducted, all four teachers mentioned attracting students to participate in the activities by creating scenarios, constructing problems, and other methods. They emphasized gaining intuitive experiences related to CT through actions and interactions. Whether designing game activities, asking students to imitate hands-on tasks, or inviting students to role-play, these four teachers demonstrated a focus on students "doing" and gaining direct experiences. They consciously delayed the introduction and teaching of knowledge, concepts, principles, etc., expecting students to obtain natural and direct understanding and experiences of a specific object through participation and action.

4.2. Cognition-based CT Experience

"Experience" begins with perceptual knowledge and leads to rational thinking. The design of teachers' computational thinking teaching involves not only students' subjective understanding and experience but also introduces key elements of CT based on these experiences, enabling students to establish a connection between intuitive feeling and rational understanding. After activities, teachers guide students to observe, reflect, and summarize the experiential process. Subsequently, they introduce abstract and objective knowledge and methods to help students transition from perceptual interactive behavior and personal experience to rational and objective laws. Moving from unconscious perception to conscious thinking allows for a profound understanding of the key contents of CT, including important knowledge, process methods, the ability to solve problems with computers and so on.

4.3. Emotion-based CT Experience

Get positive emotions in "experience" and boost students' active and sustained actions. In teachers' statements, the significance of positive emotions such as a sense of accomplishment, curiosity, and interest in the development of students' CT is emphasized. During the implementation of CTE, teachers create opportunities for students to experience positive emotions by adjusting difficulty levels, allowing room for free play, and introducing novel and interesting knowledge content.

In CTE, teachers not only assist students in acquiring knowledge and skills through creating "multi-level experience of CT", but also provide a space for the emergence of emotions. Transitioning from cognition to behavioral occurrence requires an intentional system with emotion as the core (Zhu, 2019, 9), and emotion plays a crucial role in shaping behavior, will, goals, and motivation. Equipped with certain knowledge and skills, students, with the aid of positive emotions, acknowledge the efficiency and advantages of using computers to solve problems. This fosters an interest in information technology, enabling them to shift from passive experience to active action. Students actively engage in learning, explore problems, reflect on the process, and subsequently develop their computational thinking.

4.4. Evaluation of Experience

Judging from the teachers' statements, students' participation is an important factor in evaluating the effectiveness of "experiencing it". YB and LY think that the students' participation in their own design activities is high. *Students will be confused at first, because they don't know why they want to sort it like this, but their enthusiasm is still very high, and they are willing to cooperate with you* (YB), *They are quite interested in this kind of hands-on activity* (LY). WS did not feel the enthusiasm of students to participate in the experience, so it's difficult for WS to teach the following content. *They are not very interested, even role-playing seems to be far away from their lives, so it is hard to say what they have learned* (WS).

In addition to students' participation, the help of "experience" in follow-up teaching is also a key factor to evaluate its effectiveness. YB and YZ pay more attention to students' help in learning relevant knowledge of computational thinking after the experience in class. *I feel that after they have experienced the computer sorting process, they will understand it much better later. At first, they had no idea about these, but they all answered the last exercises in the class better* (YB). *They can finally draw the shortest road map after correct abstraction. However, I hope that if someone really asks them the way in life, they can solve the problem with this abstract idea, which means that my class is effective* (YZ). WS and LY pay more attention to the students' performance after a similar experience lasts for a period of time. *When I point out their problems in class, their behavior has changed, but I can't tell whether their thinking has changed. I can only watch their performance in the later class* (WS). *After a semester like this, it is obvious that the students' ability to write programs has improved. Many times, they have some difficult problems before I say them* (LY).

5. THE SIGNIFICANCE OF "MULTI-LEVEL EXPERIENCE" TO TEACHERS AND STUDENTS

5.1. Teachers' Roles and Functions in "Multi-level Experience"

This study discovered that in primary school information technology course, students often grasp CT through "experience", subsequently prompting active cognitive engagement. "Thinking like a computer scientist" is a concept detached from the daily life experiences of primary school students, and the associated principles and working methods of computers are even more abstract and challenging to comprehend directly. Therefore, teachers' representative teaching practices in understanding CT always commence with direct experiential activities.

Starting from physical participation and action, progressing to the development and reflection of knowledge, and culminating in the heartfelt recognition of subject value and ways of thinking, teachers create an optimal learning and thinking environment for students through the design of "multi-level experiences". They assist students in perceiving and gradually forming computational thinking through progressive approaches. Eventually, students come to recognize the value of information technology disciplines and actively apply the knowledge and methods of information technology disciplines to think and problem-solve "like computer scientists".

Creating a "multi-level experience of CT" in primary schools requires careful design and planning by teachers, who play a crucial role in the process. And teachers' design of "multi-level experience" also reflects their own CT. Teachers' comprehension and understanding of CT directly impact the richness and interest of the "experience", consequently influencing the development of students' CT. In this study, teachers LY, YB, and YZ all formulated their own understandings and beliefs regarding computational thinking. Through in-depth contemplation of the key elements and essence of CT, coupled with considerations of the teaching and actual situations, they were able to create multi-level and enriching experiential activities, gradually guiding the development of students' CT.

Simultaneously, upon reviewing and reflecting on the "multi-level experience", teachers gained new insights and understanding of computational thinking teaching. This ongoing process allows them to deepen their comprehension of computational thinking and further optimize their classes.

5.2. Students' Thinking and Emotion in "Multi-level Experience"

It's found that cultivating primary school students' CT places greater emphasis on students' active participation and experiential learning in action, cognition, and emotion. It not only pays attention to students' mastery of knowledge and skills, but also considers students' emotions and attitudes during the learning experience. In "multi-level experience", abstract concepts originally associated with computers are integrated into students' classroom activities and daily life experiences. This integration allows students to gain direct insights into the key elements of computational thinking in familiar and relaxed experiences, fostering a deeper understanding under the guidance of teachers, then eliciting positive emotions such as interest, curiosity, excitement, and confidence.

Emotion and thought develop in tandem. Guided by positive emotions, students can actively explore and think critically. They not only acquire relevant knowledge and methods but also engage in innovative explorations using computers, creating unique works with personal characteristics. This process fosters respect and recognition for information technology disciplines, gradually developing thinking and problem-solving methods based on information technology, leading to the germination and development of CT.

The cultivation of CT in primary school has the trend of being closer to daily life. In primary school, students are not required to master profound computer knowledge or complicated programming methods, but pay more attention to the cultivation of students' problem-solving ability. In the "multi-level experience of CT", students gain new knowledge and new insights about computers from their familiar experiences, and then provide methods and wisdom for solving daily life problems.

6. CONCLUSION AND DISCUSSION

Through phenomenological analysis, it's found that different teachers prioritize whole-hearted participation, experiences, and multi-dimensional development in cultivating CT in primary school students. Recognizing it as a gradual, long-term process, teachers continually observe students' development in both classroom works and daily behavior. The design of the "multi-level experience" in the classroom integrates key computational thinking elements, from body-based experience to cognition-base experience, finally upon to emotion-based experience, teachers guiding students from participation to understanding and recognition, facilitating the progressive formation of computational thinking.

The "multi-level experience", embracing the idea of embodied cognition, actively involves students in constructing cognition through various perceptual channels. Teachers initiate diverse experience activities, allowing students to engage with CT elements visually, auditorily, and tactually, and this interactive process fosters direct perceptions of abstract information. Guided by teachers, students establish connections between perceptual knowledge and rational thinking, promoting cognitive expansion and thinking development. Ultimately, this nurtures an interest in information technology disciplines and enhances students' problem-solving abilities with computers.

While there's a common perception that cultivating CT in primary school students is challenging, teachers' practical experiences demonstrate its effectiveness through "multi-level experience". These activities provide intuitive experiences, expanding students' thinking and cognition, better preparing them to adapt to the information society and solve daily life problems flexibly through knowledge transfer.

7. 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 (Vol. 1, p. 25).

Colaizzi, P.F. (1978). Psychological Research as the Phenomenologist views it. In R. S. Valle, & M. King (Eds.), Existential Phenomenological Alternatives for Psychology. New York: Oxford University Press.

Lee, W.L., & Lai, Y.H. (2009). Phenomenological Methodology as an Existential Move. Chinese Journal of Guidance and Counseling, 25, 275-321.

Li, M., & Yang, Y.X. (2023). Human is not Machine: the Understanding of Computational Thinking. Open Education Research, 29(02), 55-60.

Li, S.Y. (2005). Phenomenological Pedagogy: A New Paradigm of Pedagogy. Open Education Research, 11(03), 4-7.

Loren, S. B., Ton, B., Hans, B., & Karel, M. (2010). A Handbook for Phenomenological Research in Education (Liu, J., Trans.). Educational Science Publishing House. (Original work published 1983)

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. University of Southampton (E-prints) 6pp.

Shi, J.N. (2018). Research on children's programming learning experience. [PhD thesis, Nanjing Normal University]. Retrieved from https://kns.cnki.net/kcms2/article/abstract?v=1fhXvtqifPI4JyZHpzbjd3fLPhVUG9mScA3aoRAhz6DOwh-z_GtqRv7lr1AThgV88ySQVcqlYMQAcNJQBKKQfheXfKzhpcDrxcOUnf-KchkGVFicsCiJ81U7eOso_Juy4efBU677NjFz-dfuypLCsw==&uniplatform=NZKPT&language=CHS

Van Manen, M. (2003). Researching lived experience: Human science for an action sensitive pedagogy (Song, G.W., Trans.). Educational Science Publishing House. (Original work published 1990)

Zhang, J.B. (2019). Computational Thinking Education: Concept Evolution and Challenges. Modern Distance Education Research, 31(06), 89-101.

Zhu, X.M. (2019). Outline of Emotional Education (3rd ed.). Nan Jing Normal University Press.

A New Digital Test for Assessing Computational Thinking in Chinese Preschool Children: Its Construct Validity and Reliability

Hao LI[1*], Xiao ZHANG[2]
[1,2] Faculty of Education, The University of Hong Kong, Hong Kong
u3562613@connect.hku.hk, xzhang1@hku.hk

ABSTRACT

The importance of computational thinking (CT) in K-12 education is widely recognized. Given its significance, there has been a growing need and interest in exploring how to measure children's CT skills. However, existing tools predominantly targeted primary or secondary school children, with very limited measures available for preschool children. Moreover, there is a lack of valid and reliable CT assessments that can comprehensively evaluate different CT skills of preschool children. In this study, we developed and validated a new digital CT test for assessing preschool children's six CT skills: algorithms, representation, modularity, pattern recognition, conditional logic, and debugging. Two-hundred-and-twelve Chinese preschool children (age: M ± SD = 53.96 ± 9.38 months) completed the new digital CT test. Confirmatory factor analysis showed that a six-factor model fit the data best. The ordinal coefficient alphas of the test and each sub-scale were good, indicating sufficient reliability of the test. Overall, the results demonstrate that the new digital CT test has adequate construct validity and reliability for assessing CT in preschool children aged four to five years. The study offers a new tool for future researchers and educators to comprehensively assess different CT skills of preschool children.

KEYWORDS

Computational thinking, preschool children, confirmatory factor analysis, construct validity, reliability

1. INTRODUCTION

In recent years, computational thinking (CT) has received increasing attention from worldwide researchers, educators, and policymakers, particularly in K-12 education. Jeanette Wing (2006), who popularized the concept of CT, underscored the fundamental role of CT by arguing that it should be valued as the fourth "R," akin to reading, writing, and arithmetic. Furthermore, it has been highlighted that early education of CT can better prepare children to harness the power of computing in the digital society (International Society for Technology in Education [ISTE] & Computer Science Teachers Association [CSTA], 2011) and has been associated with lower costs and more long-lasting effects than introducing CT to children at a later stage (Bers, 2022).

While the importance of CT has been emphasized and numerous efforts have been devoted to studying and fostering CT, debates on CT conceptualization have continued. Some scholars defined CT in a broad sense and conceptualized it as a set of skills that can be used to solve problems systematically across contexts (e.g., ISTE & CSTA, 2011; Shute et al., 2017). However, other scholars used a narrow scope by defining CT as domain-specific skills that are applicable to solve problems in computational contexts (e.g., Barr & Stephenson, 2011; K-12 Computer Science Framework Steering Committee, 2016). In this study, we followed the second branch of CT definitions and conceptualized CT as a set of cognitive skills for solving computational problems by using computers or robots. Defining in this way allows for a clear distinction between CT and other thinking skills or domain-general problem-solving skills. Such clarity is crucial as it can avoid ambiguity regarding what CT is and is not, which is essential for designing effective assessments.

To understand CT in preschool children, it is important to identify the CT components that are age-appropriate for them. Bers' (2021) seven powerful ideas of CT offered a valuable framework to explore young children's CT. This framework suggests seven powerful ideas of CT that are critical and age-appropriate to young children, including algorithms, modularity, control structure, representation, hardware/software, design process, and debugging. According to Bers (2021), algorithms refer to the ability to solve a problem by taking sequential steps. Modularity refers to the ability to decompose a larger task into smaller modules. Control structure determines the logical order of instruction to be followed in a program and includes loops and conditionals. Representation refers to the ability to transform and manipulate information in various ways. Hardware/software is the factual understanding of what hardware and software is. The design process describes the engineering design cycle and includes several steps (i.e., ask, imagine, plan, create, test and improve, and share). Debugging is the ability to identify and fix problems.

In this study, we further refined Bers' framework based on previous literature and our CT conceptual definition. We first separate the control structure into pattern recognition and conditional logic, as these two skills have been considered foundational cognitive skills underlying loop and conditional concepts embedded in the control structure and bedrock competencies in the field of computer science (CS). The design process was excluded from our framework as it was grounded in engineering thinking (Bers, 2021), instead of being fundamental to CS (Wing, 2006), and its broadness makes it not feasible for operationalization. Hardware/software was also excluded as it describes the factual knowledge of children, rather than domain-specific PS skills for computational problem solving. The rest CT components were included as they aligned with the domain-specific definition of CT, were feasible for operationalization, received consensus across previous studies (e.g., Bers et al., 2022; Shute et al., 2017), and were age-appropriate for young children (e.g., Kazakoff et al., 2013; Saxena et al., 2019). Together, we proposed a six-component model of CT in our study, including algorithms, representation, modularity, pattern recognition, conditional logic, and debugging. The conceptual definitions of the six CT components in this study are shown in Table 1.

Table 1. Conceptual Definitions of Six CT Components.

CT Component	Conceptual Definition
Algorithms	The ability to develop a sequence of codes to solve computational problems.
Representation	The ability to use data in various forms to solve computational problems.
Modularity	The ability to decompose codes into smaller modules that can be combined to solve computational problems.
Pattern Recognition	The ability to identify the similarities and regularities of the codes.
Conditional Logic	The ability to make decisions based on certain conditions using codes.
Debugging	The ability to identify and fix the errors in the codes.

There are increasing numbers of instruments for assessing CT (For a review, see Tang et al., 2020). However, the majority are designed for primary or secondary children, with relatively few tailored to preschool children. In general, CT assessments for young children typically fall into two categories (Tang et al., 2020): portfolio-based and traditional assessment. Portfolio-based assessments measure children's CT skills through the examination of their work products, often using predetermined rubrics. Examples are Strawhacker and Bers' (2014) "Solve-Its" tasks and Bers's (2010) 5-point "SSS" rubric along with robotic programming tasks. Traditional assessments evaluate children's CT skills through closed-ended or selective-response tasks. Examples include Marinus et al.'s (2018) Coding Development (CODE) Test 3-6 and TechCheck-K (Relkin & Bers, 2021). Nevertheless, existing tools for assessing young children's CT have several limitations. First, most have undergone rigorous validation; therefore, whether they are reliable or valid measures is yet to be confirmed. Second, the available tools predominantly indicate the overall CT skills. Hence, a comprehensive measure that can comprehensively evaluate multiple components of CT skills is lacking. Third, a more fundamental issue is that many of these tools were developed without a sound conceptual framework, resulting in a relatively weak theoretical foundation on CT structure and components that these instruments aim to measure.

To address these gaps, this study developed a new digital CT test based on the six-component model of CT and examined the construct validity and reliability of this test in Chinese preschool children. We hypothesized that the new digital CT assessment would have good construct validity and reliability.

2. METHOD

2.1. Participants and Procedures

A total of 212 children were recruited from two preschools in Guangdong Province, China, to participate in this study. The mean age of the participating children was 53.96 months (SD = 9.38). There were 112 boys and 100 girls. Most mothers (87.9%) and fathers (90.5%) obtained an associate degree or higher diploma degree. The median monthly incomes for mothers and fathers were RMB 5,000 to RMB 5,999 (equivalent to USD 704 to USD 845) and RMB 8,000 and RMB 8,999 (equivalent to USD 1,127 to USD 1,267), respectively.

Data collection was completed in the fall semester of 2023. Children were tested individually in a quiet room of their preschools. During the assessment, children were read aloud the instructions by trained research assistants (RAs) and completed the new digital CT test on a tablet. The test took about 45 minutes for each child, and it was separated into two to three sessions to avoid children's fatigue. All the recruitment and data collection procedures were approved by the institutional review board of the authors' university.

2.2. Measures

Children's CT skills were assessed by a new digital CT test that was developed by the first author. The test was designed in the "drag-and-drop" graphical programming context, in which children could produce graphical code response to each item. Such programming context has shown age-appropriateness for preschool children (e.g., Flannery et al., 2013; Papadakis et al., 2016).

The original test comprised 54 items measuring six CT components (nine items per component): algorithms, representation, modularity, pattern recognition, conditional logic, and debugging. In the algorithms task, children were asked to generate a set of arrow codes (i.e., Forward, Backward, Left, and Right) in correct sequence to direct a moving agent (i.e., a cartoon insect character, such as a caterpillar) to a target position on the 5×5 grid navigation map. In the representation task, children were asked to select one out of three new symbols – each symbol representing a set of arrow codes (e.g., "moon" symbol represents a Right arrow) – to complete a code sequence that would direct the agent to reach a target position on the navigation map. In the modularity task, children were asked to identify which decomposed modules can be used to guide the agent to the target position on the navigation map. In the pattern recognition task, children were asked to identify the repeating pattern embedded in the given codes and generate the following three/four codes that allow the agent to move based on the specific pattern. In the conditional logic task, children were asked to use conditional codes employing two types of logics: "if-then" and "if-then-else," along with the arrow codes, to navigate the agent to a target position on the navigation map. In the debugging task, children were given a set of codes and asked to evaluate whether there is an error in the codes and fix the errors if any. One point was given for the correct response to each item. Total scores were calculated.

All aforementioned tasks of the new digital CT test were administered on the Microsoft Surface Go 3 tablet, which features a touch-screen interface. The touch-screen interface allows young children to perform the test by simply dragging and dropping the graphical codes using their fingers. Such interface is familiar to and comfortable for children, as they can easily access touch-based devices in daily life. By leveraging the affordances of touch-screen technology, the new digital CT test provides an accessible, engaging, and age-appropriate way to evaluate young children's CT skills. Figure 1 shows the home page and menu page of the new digital CT test. Figure 2 shows a sample item of the new digital CT test.

Figure 1. Home Page (Left) and Menu Page (Right).

Figure 2. A Sample Item of the New Digital CT Test.

2.3. *Data Analytic Strategy*

The construct validity of the test was evaluated using confirmatory factor analysis (CFA). We estimated a six-factor CFA model, which corresponds to the hypothesis that the test consists of six dimensions (i.e., algorithms, representation, modularity, pattern recognition, conditional logic, and debugging), and two competing models: a one-factor model, which corresponds to the hypothesis that the new digital CT test assesses a single dimension of CT, and a higher-order CFA model corresponds to the hypothesis that the test consists of six aforementioned dimensions as first-order factors and an overall CT as a second-order factor. The weighted least squares estimator with means and variance adjustments (WLSMV) for ordered categorical data was used because all the test items were dichotomous (i.e., 1 = correct and 0 = incorrect). Four fit indices were used to evaluate the goodness-of-fit of the model: the chi-square statistic (χ^2), comparative fit index (CFI; ≥.95), Tucker-Lewis index (TLI; ≥.95), and root mean square error of approximation (RMSEA; ≤.06) (Hu & Bentler, 1999). The chi-square difference test was conducted and ΔCFIs were reported for model comparisons. A non-significant chi-square difference test statistic and ΔCFI values lower than .01 indicate that a more complex model does not demonstrate a better fit to the data and a more parsimonious model should be selected. Further, ordinal coefficient alphas were computed to examine the reliability of the test. An ordinal coefficient alpha greater than .70 indicates acceptable reliability (Gadermann et al., 2012). All analyses were conducted in R software (R Core Team, 2022).

3. RESULTS

3.1. *Construct Validity*

Table 2 shows the model fit indices of all estimated CFA models and model comparison results. The hypothesized six-factor model demonstrated good fit to the data: $\chi^2(1362) = 1439.28$, $p = .07$, CFI = .98, TLI = .99, RMSEA = .02. We then estimated two alternative CFA models: a one-factor model and a higher-order model. Results showed that both models had a good fit to the data, with $\chi^2(1377) = 1566.76$, $p < .001$, CFI = .97, TLI = .97, RMSEA = .03 for the one-factor model and $\chi^2(1371) = 1444.67$, $p = .08$, CFI = .99, TLI = .99, RMSEA = .02 for higher-order model. Furthermore, model comparison results showed that the six-factor model had significantly better fit to the data than the one-factor model, $\Delta\chi^2 = 132.38$, $\Delta df = 15$, $p < .001$; ΔCFI = .02. However, the higher-order model did not fit significantly better than the six-factor model: $\Delta\chi^2 = 5.58$, $\Delta df = 9$, $p = .78$; ΔCFI = .001. Therefore, the six-factor model was selected. However, a further examination of the factor loading of the six-factor model showed that three items in the representation dimension had low factor loadings (i.e., < .30). We removed these items and estimated a modified six-factor model. Results showed that the modified six-factor model continued to have a good fit to the data: $\chi^2(1209) = 1286.25$, $p = .001$, CFI = .99, TLI = .99, RMSEA = .02. The standardized factor loadings of the modified six-factor model ranged from .33 to .92 and all loadings were statistically significant. Therefore, the modified six-factor model was selected as our final model.

Table 2. Confirmatory Factor Analysis Results.

Models	χ^2	*df*	CFI	TLI	RMSEA (90%CI)
Six-factor model (M1)	1439.28, $p = .07$	1362	.98	.99	.02 (.01 – .02)
One-factor model (M2)	1566.76, $p < .001$	1377	.97	.97	.03 (.02 – .03)
Higher-order model (M3)	1444.67, $p = .08$	1371	.99	.99	.02 (.00 – .03)
Revised Six-factor model ($M1_R$)	1286.25, $p = .06$	1209	.99	.99	.02 (.00 – .03)
Model comparisons	$\Delta\chi^2$		Δdf	ΔCFI	
M1 vs M2	132.38, $p < .001$		15	.02	
M1 vs M3	5.58, p = .78		9	.001	

3.2. *Reliability*

Ordinal coefficient alpha was computed to evaluate the reliability of the overall test and its six dimensions. Results showed good reliability of the new digital test. Specifically, the ordinal coefficient alphas were .97 for the overall test, .91 for algorithms, .75 for representation, .88 for modularity, .94 for pattern recognition, .86 for conditional logic, and .90 for debugging.

4. DISCUSSION AND CONCLUSION

This study developed and validated a new digital CT test for preschool children. Adapted from Bers' (2021) framework of CT for young children, we proposed a six-component CT model. The CFA results confirmed the six-factor structure of CT, suggesting that preschool children's CT consisted of components including algorithms, representation, modularity, pattern recognition, conditional logic, and debugging. Reliability analyses demonstrated that the new digital CT test and its six sub-tests had adequate reliability. Together, these findings provide empirical evidence on the construct validity of the new digital CT test and suggest that it is a reliable instrument for assessing CT in preschool children aged four to five years in China.

Our new digital CT test has several strengths. First, this test was based on a theoretically sound CT model. Most of the previous work provided little information about the theoretical justifications for CT structure and components in the development of CT assessments. Our test was designed grounding on the six-component model that was adapted from Bers' (2021) CT framework, which strengthens the theoretical bases of the new digital CT test. Second, this test has showed good construct validity and reliability. Only limited studies have validated the existing CT assessments; therefore, whether these CT assessments were valid and reliable is yet unknown. In our study, we examined a series of CFA models and evaluated the ordinal coefficient alphas of the test. Our findings offered empirical evidence supporting the adequate construct validity and reliability of the new digital CT test. Third, this test can indicate children's proficiency in different CT skills. Unlike most existing CT assessments, which use the total score to indicate children's overall CT skill, our test includes six dimensions (i.e., algorithms, representation, modularity, pattern recognition, conditional logic, and debugging). Each of these dimensions contains several items, allowing for a comprehensive evaluation of children's proficiency across multiple CT skills. Fourth, compared to non-digital tests, the new CT test uses digital format, which allows an autoscoring process that minimizes the risk of scoring and data entry errors. Meanwhile, the digital format of the test increases the standardization of the test procedures, thereby enhancing the internal validity of the test.

Despite these strengths, the new digital CT test requires further validation in future studies. For instance, future research can collect multiple-wave data to examine the longitudinal measurement invariance and check if the six-factor structure remains consistent over time in young children. It is also important to establish convergent validity by examining the associations between children's the new digital CT test and other criterion measures. Furthermore, future research can consider applying Item Response Theory (IRT) to analyze psychometric properties, such as item difficulty, which can enhance our understanding of each test item's quality. Lastly, given that this study only included Chinese preschool children, whether the new digital CT test is applicable to young children from different cultural backgrounds is unclear. Thus, cross-cultural adaptation and validation of the new digital CT test are essential steps to ensure that the test is effective for evaluating CT in diverse cultural contexts.

5. REFERENCES

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2), n2.

Bers, M. U. (2021). *Coding as a playground: Programming and computational thinking in the early childhood classroom* (2nd ed.). Routledge.

Bers, M., Strawhacker, A., & Sullivan, A. (2022). *The state of the field of computational thinking in early childhood education, OECD Education Working Papers, No. 274*, OECD Publishing, Paris.

Flannery, L.P., Kazakoff, E.R., Bontá, P., Silverman, B., Bers, M.U., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. In *Proceedings of the 12th international conference on interaction design and children (IDC '13)* (pp. 1–10). ACM, New York, NY, USA.

Gadermann, A. M., Guhn, M., & Zumbo, B. D. (2012). Estimating ordinal reliability for Likert-type and ordinal item response data: A conceptual, empirical, and practical guide. *Practical Assessment, Research, and Evaluation, 17*(3), 1-13.

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.

ISTE & CSTA (2011). *Computational thinking leadership toolkit*. https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf

K-12 Computer Science Framework Steering Committee. (2016). *K-12 computer science framework*. http://www.k12cs.org.

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal, 41*(4), 245-255.

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the cognition of coding in 3-to-6-year-olds: The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. In *Proceedings of the 2018 ACM conference on international computing education research* (pp. 133-141).

R Core Team. (2022). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Relkin, E., & Bers, M. (2021, April). *Techcheck-K: A measure of computational thinking for kindergarten children* [Paper presentation]. 2021 IEEE Global Engineering Education Conference (EDUCON). IEEE.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020). Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education. *The Asia-Pacific Education Researcher, 29*(1), 55-66.

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

Strawhacker, A. L., & Bers, M. U. (2014). "I want my robot to look for food": Comparing children's programming comprehension using tangible, graphical, and hybrid user interfaces. *International Journal of Technology and Design Education, 25*, 293-319.

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*, 103798.

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

Exploring the Relationship between Personality Traits and Computational Thinking among University Students

Qi Luo[1, 2], Shuhan Zhang[1*]
[1] Macao Polytechnic University, Macao
[2] Heyuan Polytechnic, China
keroroluo@outlook.com, shuhan@mpu.edu.mo

ABSTRACT
Computational thinking (CT) is an immensely important skill in the 21st century, and enhancing students' CT has become a focal point for educators and researchers. Numerous studies have identified cognitive abilities related to CT. However, relying solely on cognitive abilities does not fully explain the learning outcomes that are associated with CT. Additionally, research exploring the non-cognitive aspects of CT is scarce, especially in the context of university students. To address this gap, our study aims to explore the relationship between university students' personalities and CT, understanding the role of different personality traits in CT. The participants of this study were 133 sophomore university students, and data on personality traits and CT performance were collected. The findings reveal significant differences in Conscientiousness (C) and Agreeableness (A) among students with different levels of CT, with linear regression analysis further suggesting that Conscientiousness (C) can predict CT performance. The results highlight the importance of students' personalities in CT, implicating the necessity of incorporating these traits in future teaching practices.

KEYWORDS
computational thinking, programming, personality, higher education

1. INTRODUCTION
Computational Thinking (CT) is an essential skill for everyone (Wing, 2006), defined as a problem-solving approach that integrates technology with cognitive skills and is applicable across all disciplines (ISTE, 2015). It has now been comprehensively integrated into K–12 education (Rana et al., 2022). CT can be employed to solve various types of problems (Pelánek & Effenberger, 2023). While different activities can foster the development of CT in the classroom (Zapata-Caceres et al., 2021), programming is widely acknowledged as a fundamental way to bring CT to life (Lye & Koh, 2014). Consequently, coding, rooted in Computer Science (CS), has become the primary method educators use to teach CT (Grover & Pea, 2013) spanning from early childhood education (Bers et al., 2014) to higher education (Romero et al., 2017).

In this context, enhancing students' CT learning outcomes has become a focal concern for educators and researchers (Zhang & Wong, 2023b). Consequently, many scholars have dedicated themselves to identifying factors related to CT (Gerosa et al., 2021; Marinus et al., 2018; Zhang & Wong, 2023a). Current research indicates that on a cognitive level, skills such as mathematics, spatial ability, and reasoning have been confirmed to be associated with CT (Scherer et al., 2019; Zhang & Wong, 2023a), suggesting that fostering these skills can further enhance students' CT development. However, relying solely on cognitive abilities is insufficient to fully explain students' learning outcomes (Leeson et al., 2008; Meyer et al., 2024), necessitating continued exploration of the impact of non-cognitive factors on students. On the non-cognitive level, it has been found that personality traits can significantly influence learning outcomes, as different characteristics affect learners' behaviors and motivations, thereby impacting their academic performance (Tlili et al., 2023). Consequently, researchers have begun investigating the impact of personality on CT. For instance, Zhang and Wong (2023b) studied the relationship between personality traits and CT in elementary school students, discovering that Conscientiousness (C) notably predicts students' CT performance. Similarly, Román-González et al. (2016) explored the correlation between middle school students' personality traits and CT, finding that CT is associated with Extroversion (E), Conscientiousness (C), and Openness (O). These studies collectively indicated that personality indeed affected students' CT performance. However, research on the relationship between personality and CT among university students remains limited.

To address this gap, this study explores how personality influences CT in the context of university students. It seeks to provide in-depth insights that can assist educators and curriculum designers in developing more personalized teaching methods and learning strategies. These tailored approaches aim to cater to students with different personality types, thereby enhancing their learning efficiency and effectiveness in the field of CT. The research questions for this study are as follows:

RQ1. How do personality traits differ across students with different levels of CT performance?

RQ2. To what extent do personality traits predict CT performance?

2. METHOD

2.1. Participants
The research sample was drawn from three sophomore classes in the Electronic Information Engineering Technology program at a public university in Southern China, comprising a total of 133 students. Among them, 103 were male and 30 were female, with an average age of 19 years.

2.2. Measurements

2.2.1. Personality

Since Fiske identified the five major personality traits in 1949 (Fiske, 1949), researchers have consistently replicated and refined their findings over the subsequent decades. It was not until McCrae and Jogn proposed what is now widely recognized as 'the Big Five Personality Traits' or the 'Five-Factor Model' (McCrae & John, 1992). This model has become a foundational framework in the field, representing a comprehensive understanding of personality traits.

In this study, the Big Five Personality Inventory utilized was a Chinese version translated from the Big Five Inventory (BFI-2) (Soto & John, 2017), with its validity and reliability already established (Zhang et al., 2022). This scale designed to assess individual personalities consists of 60 items across five dimensions (Roccas et al., 2002):

- **Extraversion (E):** Characterized by sociability and activity, people high in extraversion tend to exhibit talkativeness, assertiveness, and lively energy.
- **Agreeableness (A):** Reflecting qualities such as kindness, cooperativeness, and modesty, agreeable individuals often display a compliant and gentle nature in social contexts.
- **Conscientiousness (C):** Marked by a sense of responsibility and organization, they are generally meticulous, reliable, and display a strong adherence to duty.
- **Negative Emotionality (N):** This trait is linked with emotional challenges like anxiety, depression, and anger, suggesting a predisposition towards feelings of insecurity.
- **Open-Mindedness (O):** Those who are typically intellectual and imaginative, showing a marked openness to new experiences and a sensitivity to diverse perspectives.

Students completed the self-report questionnaire using a 5-point Likert scale, ranging from 1 (strongly disagree) to 5 (strongly agree), to rate the occurrence of behaviors described in each item. The scale demonstrated good reliability for all five factors in our research, with Cronbach's Alpha values of .771, .786, .861, .827, and .706 for Extraversion (E), Agreeableness (A), Conscientiousness (C), Negative Emotionality (N), and Open-Mindedness (O), respectively, all of which exceed the threshold of 0.7 (Nunnally, 1994).

2.2.2. Computational thinking

At present, there are no specialized tools specifically for assessing CT in university students, except for the Computational Thinking Scale (CTS) (Korkmaz et al., 2017). However, since CTS is a self-report scale, it may not align precisely with students' actual CT performance. Recognizing that computer programming is a fundamental medium for applying CT (Lye & Koh, 2014), this study utilized the overall assessment scores from a Python programming course, undertaken in the first semester of the sophomore year, as a proxy for CT. The assessment for this course comprises both theoretical and practical components, with a total test duration of 1.5 hours.

2.3. Procedure

The Python programming test was conducted by the course instructor in a university-organized setting. One week after completing this test, the first author distributed the Big Five Personality Inventory to the students through an online survey platform. The students were required to complete the questionnaire simultaneously during a designated period. All participants were informed of the study's purpose and consented to participate in the research.

2.4. Data analysis

After data preprocessing, 120 valid questionnaires were obtained for final analysis. Initially, Pearson's correlation coefficient was used to measure the strength and direction of the linear relationship between students' personality and their CT performance. To answer RQ1, a one-way ANOVA was conducted to compare whether there were significant differences in the mean values of personality traits among three independent groups with different levels of CT performance. For RQ2, a linear regression model was employed to observe the extent to which personality can predict students' CT performance.

3. RESULTS

3.1. Descriptive statistics and correlation analyses

The descriptive statistics and correlations among the variables are presented in Table 1. The results indicated that, based on the Likert scale, the mean values for all personality traits exceeded the midpoint of 3, except Negative Emotionality (N). The overall CT performance of the students was at a good level, with an average score of 80.58 out of a total of 100. Pearson's correlation coefficient was used for the correlation analysis, which revealed that CT performance was significantly positively correlated with Conscientiousness (C) and moderately positively correlated with Agreeableness (A). However, no correlation was found with the other three personality traits.

Table 1. Descriptive statistics and correlations.

	M	SD	(E)	(A)	(C)	(N)	(O)
(E)	3.04	.49					
(A)	3.71	.46	.511**				
(C)	3.42	.52	.538**	.641**			
(N)	2.78	.56	-.441**	-.514**	-.466**		
(O)	3.38	.46	.578**	.318**	.342**	-.243**	
CT	80.58	6.80	.115	.196*	.257**	.010	.125

$^{**}p < 0.01$; $^{*}p < 0.05$.

3.2. Personality traits of groups with different levels of CT performance

To observe the personality traits of students with varying levels of CT, we have roughly divided students into three categories based on their grades. Students scoring 85 or above are classified as having excellent CT performance, while those scoring 78 or below are considered to have poorer CT performance. Students with scores falling within the remaining range are categorized as having medium CT performance. Table 2 presents the average personality trait scores and the results of a one-way ANOVA for each CT performance level.

The table reveals that students with high CT performance have higher average scores in Extraversion (E) and Conscientiousness (C). Additionally, the ANOVA revealed statistically significant differences in Agreeableness (A) (F (2,117) = 4.757, p < .05) and Conscientiousness (C) (F (2,117) = 6.500, p < .01) among groups with different levels of CT.

Table 2. ANOVA of the personality factors across the CT performance levels.

	CT	N	M	SD	F	P
(E)	High	33	3.14	.48	1.872	.158
	Medium	55	3.06	.52		
	Low	32	2.90	.45		
(A)	High	33	3.77	.42	4.757*	.010
	Medium	55	3.79	.45		
	Low	32	3.50	.48		
(C)	High	33	3.56	.52	6.500**	.002
	Medium	55	3.48	.47		
	Low	32	3.15	.51		
(N)	High	33	2.80	.58	.265	.768
	Medium	55	2.73	.58		
	Low	32	2.80	.51		
(O)	High	33	3.43	.45	.825	.441
	Medium	55	3.40	.49		
	Low	32	3.29	.41		

**p < 0.01; *p < 0.05.

Table 3. Summary of multiple regression model on CT performance.

Variable	Standardized β	t	R	R^2
Model 1			.315	.099
(E)	-.047	-.374		
(A)	.126	1.008		
(C)	.270*	2.189		
(N)	.196	1.815		
(O)	.068	.626		
Model 2			.394	.155
(E)	-.030	-.245		
(A)	.122	1.009		
(C)	.261*	2.166		
(N)	.126	1.166		
(O)	.045	.422		
Gender [a]	.247**	2.744		

Note: [a] Effect coded: 0 = male, 1 = female.
*p < 0.05; **p <0.01.

3.3. Regression model of CT performance based on personality.

Using CT performance as the dependent variable, a two-stage multiple linear regression was conducted with five dimensions of personality traits as independent variables. The ENTER method was employed. Initially, students' personality data results from the Big Five personality test were inputted. Subsequently, gender was introduced as a control variable. Table 3 summarizes the results of the model estimation. In the first model, F (5,114) = 2.551, p = 0.034, the variables predicted 9.9% of the CT performance. In the second model, with the inclusion of gender as a control variable, F (6,113) = 3.467, p = .004. Conscientiousness (C) (β = 0.261, p = 0.032) and gender (β = 0.247, p =.007) were predictive of students' programming performance, and the overall regression model could predict 15.5% of the CT performance.

4. DISCUSSION & CONCLUSION

This paper explores the relationship between university students' CT performance and their personality traits. We observed a positive correlation between CT in students and two personality traits: Agreeableness (A) and Conscientiousness (C). Further categorizing students' CT performance into high, medium, and low groups, we noted that the differences in average values across these groups are significant in terms of Conscientiousness (C) and Agreeableness (A). A further linear regression analysis revealed that Conscientiousness (C) could significantly predict CT, suggesting its potential as a key factor influencing CT performance. Incorporating gender as a variable improved the model's explanatory power, indicating that gender may also be an influencing factor.

From the results of this study, Conscientiousness (C) within the Big Five personality model appears to have the most significant impact on students' CT. This finding aligns with the results of Zhang and Wong's (2023b) study on the relationship between CT and personality among elementary school students. Similarly, in the context of middle school students, Román-González et al. (2016) also found a significant correlation between Conscientiousness (C) and students' CT. Conscientiousness (C) comprises three subdimensions: Organization, Efficiency, and Responsibility. In our research, we additionally discovered that Organization significantly correlates (p <.001) with CT. Since CT is a skill for problem-solving (Wing, 2008), and organization is a critical component of problem-solving (Becker & Baloff, 1969), this may contribute to the effect of Conscientiousness (C) on students' CT performance.

Agreeableness (A) also appeared to play an important role in CT, which aligns with Durak et al. (2021). Studies have shown that Agreeableness (A) is associated with an increase in students' autonomous learning motivation, leading to enhanced goal progression (Levine et al., 2021). Autonomous motivation plays a crucial role in the learning process, as it can drive students to go beyond mere task completion and deeply understand and master complex concepts, such as CT. Nevertheless, similar findings were not observed in relevant studies in the K–12 context (Román-González et al., 2016; Zhang & Wong, 2023b), which could be attributed to the differences in the study population. Thereby, more research can be conducted in this area, expanding its reach to encompass higher education.

The findings consistently indicate that students' development in CT is associated with their personality traits. Considering these findings, it becomes essential for educators to go beyond merely focusing on cognitive knowledge in their teaching strategies. It is suggested that they could also give considerable attention to students' non-cognitive traits, such as personality. Understanding and tailoring approaches to accommodate individual personality differences can enable educators to develop more effective teaching methods and strategies, which may help improve learning outcomes for each individual. For example, Conscientiousness (C) is closely linked with self-regulated learning (Barros et al., 2022), self-discipline (Slovakia et al., 2021), persistence (Wilmot & Ones, 2019), and achievement striving (Cardoso et al., 2017), which can be incorporated into teaching practices to enhance the development of students' CT.

Some limitations of the study need to be noted. Firstly, due to convenient sampling, the sample size is relatively small, which may not be representative of the entire university student population. Future research should aim to expand the sample size and include students from a wider range of disciplines to replicate this study and explore potential new findings. Secondly, the research utilized students' programming course grades as a proxy for their CT performance. While programming is a primary tool for teaching CT in education, it is not synonymous with CT. Therefore, it is suggested that future studies develop appropriate CT assessment tools for university students.

5. REFERENCE

Barros, A., Simão, A. M. V., & Frisson, L. (2022). Self-regulation of learning and conscientiousness in Portuguese and Brazilian samples. *Current Psychology*, *41*(11), 7835–7842. Retrieved from https://doi.org/10.1007/s12144-020-01232-y

Becker, S. W., & Baloff, N. (1969). Organization Structure and Complex Problem Solving. *Administrative Science Quarterly*, *14*(2), 260. Retrieved from https://doi.org/10.2307/2391104

Berinšterová, M., Bozogáňová, M., Magdová, M., Kapová, J., & Fuchsová, K. (2021). *PROCRASTINATION AND SELF-CONCEPT IN MORE/LESS CONSCIENTIOUS STUDENTS*. 162–166. Retrieved from https://doi.org/10.36315/2021inpact034

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. Retrieved from https://doi.org/10.1016/j.compedu.2013.10.020

Cardoso, A. P., Ferreira, M., Abrantes, J. L., & Seabra, C. (2017). *CONSCIENTIOUSNESS AND INTRINSIC MOTIVATION AS PREDICTORS OF PERCEIVED LEARNING AND ACADEMIC ACHIEVEMENT*.

Fiske, D. W. (1949). Consistency of the factorial structures of personality ratings from different sources. *The Journal of Abnormal and Social Psychology*, *44*(3), 329–344. Retrieved from https://doi.org/10.1037/h0057198

Gerosa, A., Koleszar, V., Tejera, G., Gómez-Sena, L., & Carboni, A. (2021). Cognitive abilities and computational thinking at age 5: Evidence for associations to sequencing and symbolic number comparison. *Computers and Education Open*, *2*, 100043. Retrieved from https://doi.org/10.1016/j.caeo.2021.100043

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. Retrieved from https://doi.org/10.3102/0013189X12463051

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. Retrieved from https://doi.org/10.1016/j.chb.2017.01.005

Leeson, P., Ciarrochi, J., & Heaven, P. C. L. (2008). Cognitive ability, personality, and academic performance in adolescence. *Personality and Individual Differences*, *45*(7), 630–635. Retrieved from https://doi.org/10.1016/j.paid.2008.07.006

Levine, S. L., Milyavskaya, M., Powers, T. A., Holding, A. C., & Koestner, R. (2021). Autonomous motivation and support flourishes for individuals higher in collaborative personality factors: Agreeableness, assisted autonomy striving, and secure attachment. *Journal of Personality*, *89*(5), 899–914. Retrieved from https://doi.org/10.1111/jopy.12622

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. Retrieved from https://doi.org/10.1016/j.chb.2014.09.012

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018). Unravelling the Cognition of Coding in 3-to-6-year Olds: The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. *Proceedings of the 2018 ACM Conference on International Computing Education Research*, 133–141. Retrieved from https://doi.org/10.1145/3230977.3230984

McCrae, R. R., & John, O. P. (1992). An Introduction to the Five-Factor Model and Its Applications. *Journal of Personality*, *60*(2), 175–215. Retrieved from https://doi.org/10.1111/j.1467-6494.1992.tb00970.x

Meyer, J., Lüdtke, O., Schmidt, F. T. C., Fleckenstein, J., Trautwein, U., & Köller, O. (2024). Conscientiousness and Cognitive Ability as Predictors of Academic Achievement: Evidence of Synergistic Effects From Integrative Data Analysis. *European Journal of Personality*, *38*(1), 36–52. Retrieved from https://doi.org/10.1177/08902070221127065

Pelánek, R., & Effenberger, T. (2023). The Landscape of Computational Thinking Problems for Practice and Assessment. *ACM Transactions on Computing Education*, *23*(2), 1–29. Retrieved from https://doi.org/10.1145/3578269

Rana, D. S., Dimri, S. C., Malik, P., & Dhondiyal, S. A. (2022). Impact of Computational Thinking in Engineering and K12 Education. *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, 697–701. Retrieved from https://doi.org/10.1109/ICIRCA54612.2022.9985593

Roccas, S., Sagiv, L., Schwartz, S. H., & Knafo, A. (2002). The Big Five Personality Factors and Personal Values. *Personality and Social Psychology Bulletin*, *28*(6), 789–801. Retrieved from https://doi.org/10.1177/0146167202289008

Román-González, M., Pérez-González, J.-C., & Moreno-León, J. (2016). *Does computational thinking correlate with personality? The non-cognitive side of computational thinking*.

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. Retrieved from https://doi.org/10.1186/s41239-017-0080-z

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology*, *111*(5), 764–792. Retrieved from https://doi.org/10.1037/edu0000314

Soto, C. J., & John, O. P. (2017). The next Big Five Inventory (BFI-2): Developing and assessing a hierarchical model with 15 facets to enhance bandwidth, fidelity, and predictive power. *Journal of Personality and Social Psychology*, *113*(1), 117–143. Retrieved from https://doi.org/10.1037/pspp0000096

Tlili, A., Sun, T., Denden, M., Kinshuk, Graf, S., Fei, C., & Wang, H. (2023). Impact of personality traits on learners' navigational behavior patterns in an online course: A lag sequential analysis approach. *Frontiers in Psychology*, *14*, 1071985. Retrieved from https://doi.org/10.3389/fpsyg.2023.1071985

Wilmot, M. P., & Ones, D. S. (2019). A century of research on conscientiousness at work. *Proceedings of the National Academy of Sciences*, *116*(46), 23004–23010. Retrieved from https://doi.org/10.1073/pnas.1908430116

Wing, J. M. (2006). *Computational thinking*.

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. Retrieved from https://doi.org/10.1098/rsta.2008.0118

Yildiz Durak, H., Saritepecı̇, M., & Aksu Dünya, B. (2021). Examining the Relationship between Computational Thinking, Lifelong Learning Competencies and Personality Traits Using Path Analysis. *Bartın Üniversitesi Eğitim Fakültesi Dergisi*, *10*(2), 281–292. Retrieved from https://doi.org/10.14686/buefad.888374

Zapata-Caceres, M., Martin-Barroso, E., & Roman-Gonzalez, M. (2021). Collaborative Game-Based Environment and Assessment Tool for Learning Computational Thinking in Primary School: A Case Study. *IEEE Transactions on Learning Technologies*, *14*(5), 576–589. Retrieved from https://doi.org/10.1109/TLT.2021.3111108

Zhang, B., Li, Y. M., Li, J., Luo, J., Ye, Y., Yin, L., Chen, Z., Soto, C. J., & John, O. P. (2022). The Big Five Inventory–2 in China: A Comprehensive Psychometric Evaluation in Four Diverse Samples. *Assessment*, *29*(6), 1262–1284. Retrieved from https://doi.org/10.1177/10731911211008245

Zhang, S., & Wong, G. K. W. (2023a). Exploring the underlying cognitive process of computational thinking in primary education. *Thinking Skills and Creativity*, *48*, 101314. Retrieved from https://doi.org/10.1016/j.tsc.2023.101314

Zhang, S., & Wong, G. K. W. (2023b). Unravelling the underlying mechanism of computational thinking: The mediating role of attitudinal beliefs between personality and learning performance. *Journal of Computer Assisted Learning*, jcal.12900. Retrieved from https://doi.org/10.1111/jcal.12900

K–12 Pre-service Teachers' Perspectives on AI Models and Computational Thinking: The Insights from an Interpretative Research Inquiry

Muhammad ALI[1*], Gary K.W. WONG[2], Ming MA[3]
[1,2,3]The University of Hong Kong, Hong Kong (S.A.R.)
akula@connect.hku.hk, wongkwg@hku.hk, mingma@connect.hku.hk

ABSTRACT
Computational thinking (CT) has emerged as a pivotal component of K–12 education for fostering problem-solving skills among the next generation of learners. However, CT integration remains an arduous challenge for K–12 teachers due to their limited preparation, prior knowledge, and relevant expertise in CT. To respond to this challenge in Hong Kong, we designed and implemented an introductory CT course employing plugged and unplugged CT approaches alongside AI technology to prepare pre-service teachers. To inform the design of our future course iterations, we conducted an interpretative research inquiry within the course to explore how these teacher trainees learn CT through different teaching and learning activities. Our data analysis accentuated the emergence of three core themes, encompassing numerous subthemes within our data. The three core themes are delineated as themes of (1) CT Knowledge, (2) CT Perspectives, and (3) Potential Barriers. This paper disseminates part of our findings on the trainees' CT Perspectives *only*: It delves into their post-course perspectives on AI models and CT, seeking to elucidate the pedagogical implications of integrating AI models and CT into K–12 education. These perspectives provide new insights into teaching and learning CT through prompt engineering, which could emerge as a novel approach to democratizing CT education and could be the conduit to bridge the divide between CT and general education.

KEYWORDS
AI models, computational thinking, K–12 education, pre-service teachers, perspectives

1. INTRODUCTION
The cultivation of computational thinking (CT) in K–12 education has been consistently emphasized, yet it remains a formidable challenge, primarily attributed to the limited knowledge, training, and preparedness of teachers in designing and implementing appropriate teaching and learning (T&L) activities within their specific K–12 contexts (Kong et al., 2020; Ung et al., 2022). Concurrently, with AI technology becoming ubiquitous, a growing corpus of research focuses on integrating AI with traditional CT tools. These studies typically aim to acquaint K–12 students with AI and its sub-domains via leveraging CT tools such as autonomous and programmable educational robotics, block-based programming, and specifically designed games (e.g., Park & Shin, 2022; Priya et al., 2022). Regardless, a research gap persists in exploring such integrations from a teacher's perspective, owing to its nascent nature, particularly in the context of T&L of CT.

The rapid proliferation of AI models (e.g., large-scale pre-trained and large language models) is being extensively debated in higher education settings (Michel-Villarreal et al., 2023; Xia & Li, 2022; Yilmaz & Karaoglan Yilmaz, 2023). However, these models' full potential and implications remain predominantly unexplored within K–12 education and teacher education. The study under consideration, therefore, endeavored to contribute to bridging this burgeoning research gap. It explored how CT T&L activities, employing both plugged (i.e., with computers, programming, or digital) and unplugged (i.e., without computers, programming, or non-digital) approaches alongside AI technology, influenced pre-service teacher trainees' perspectives on applying AI models for their prospective T&L to cultivating CT in K–12 education.

2. METHODS

2.1. Research Participants & Context
The research inquiry was conducted in a unique introductory CT course at the undergraduate level at a prominent university in Hong Kong. This course was a free elective for undergraduate students across the participating institution who aspired to become K–12 teachers. The course cohort comprised twenty-eight students, including twenty-five pre-service teachers and three part-time in-service teachers. The in-service teachers did not participate in the research. Altogether, fifteen (n=15) pre-service teachers participated in the inquiry; the sampling criteria are described in the next section. The pre-service teachers, who were undergraduate students, came from several faculties and departments within the institution. As a result, the cohort showcased a significant diversity in terms of educational backgrounds and disciplinary expertise, including applied artificial intelligence, computer science, mechanical engineering, information management, quantitative finance, economics, chemistry, biological sciences, molecular biology, biotechnology, education, social sciences, and education psychology.

The course was designed and implemented with T&L activities to facilitate CT knowledge construction, which was identified based on the past foundation of research. Firstly, this encompassed the theoretical and conceptual aspects of CT (Kong et al., 2020; Rich et al., 2021; Shute et al., 2017), with learning content including introductions to CT practices (e.g., decomposition, pattern recognition, abstraction, algorithm design, debugging), formal logic, technology integration, the history of computing and algorithms, and pedagogical approaches such scaffolding, active learning, and constructionism. The trainees were engaged through various passive and active T&L activities, such as lectures, individual Q&A sessions, group discussions, and interactive demonstrations.

Secondly, the course entailed T&L activities purposefully addressing the technical and applied aspects of CT (Angeli, 2022; Tedre et al., 2021; Ung et al., 2022), with learning content including CT concepts and constructs (e.g., initialization, functions, variables, conditionals, iteration, and arrays), learning of these concepts and constructs using unplugged platforms such as LEGO patterns, and applications of these concepts and constructs with plugged platforms using block-based programming languages (e.g., *Blockly* and *Snap!*). The trainees utilized *Blockly* to program Micro-Bits to develop peripheral devices and DIY (Do-It-Yourself) projects. More importantly, they designed and interacted with rudimentary chatbots leveraging AI models (e.g., ChatGPT, GPT-3, Cohere, DALLE-2, and Stable Diffusion) using Snap! (see Figure 1).

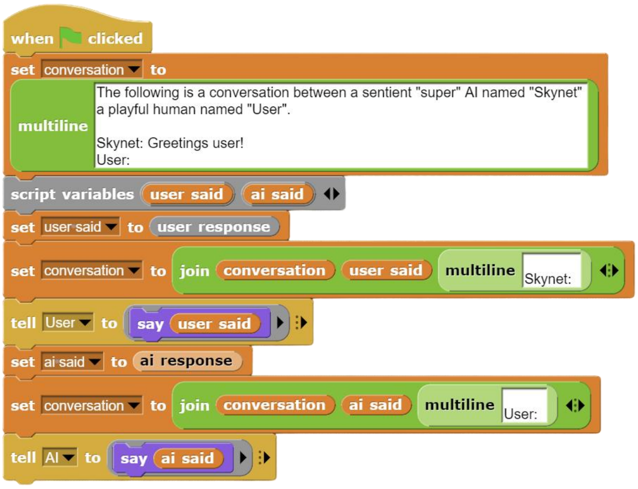


Figure 1. A Rudimentary AI Chatbot Powered by GPT-3 Model and Developed Using *Snap!*

Lastly, the course engaged the trainees in collaboratively developing learning designs for their preferred K–12 context: kindergarten, primary, or secondary (Tucker-Raymond et al., 2021). This involved designing intended learning outcomes based on the revised Bloom's Taxonomy (Krathwohl, 2002) and developing age-appropriate T&L activities with the CT tools they had previously practiced.

2.2. Data Collection & Analysis

We conducted an interpretative research inquiry exploring the pre-service teacher trainees' learning experiences of CT during the course (Merriam & Tisdell, 2015b). The trainees were sampled purposively with the following criteria: (i) Undergraduate students who aspire to be K–12 teachers, (ii) have no formal teaching experience, (iii) participated in all the T&L activities, (iv) completed all their weekly reflections, and (v) consented to participate in the study. Fifteen trainees met the criteria and were engaged in the data collection. We used multiple qualitative data sources for triangulation and generating a *think* description (see Figure 2). This included (a) Participant Observations (e.g., field notes, photographs, video records, and learning artifacts), (b) Participant Reflections (e.g., reflections, personal insights, and comments), and (c) Participant Interviews (e.g., response to semi-structured interview questions). Concurrently, we engaged in the inductive content analysis following a two-phase iterative coding process (Merriam & Tisdell, 2015a). During this, two coders worked independently and reported an inter-coder agreement greater than 89% afterward. The data analysis accentuated the emergence of three core themes, encompassing numerous subthemes within our data. The three core themes are delineated as themes of (1) CT Knowledge, (2) CT Perspectives, and (3) Potential Barriers (see Figure 2). Nevertheless, this paper disseminates part of our findings on the trainees' CT Perspectives *only*: It delves into their post-course perspectives on AI models and CT, seeking to elucidate the pedagogical implications of integrating AI models and CT into K–12 education, respectively.

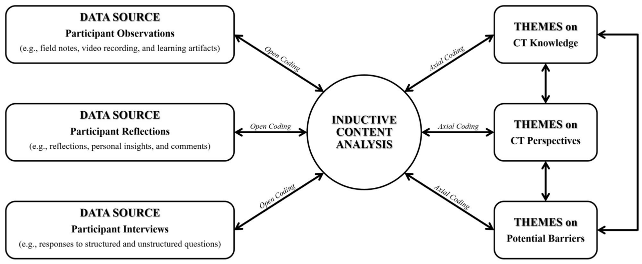


Figure 2. Iterative Process of Data Collection & Analysis.

3. FINDINGS

This section presents part of our findings on the pre-service teacher trainees' CT Perspectives. The inductive content analysis revealed that only *ten* of the fifteen participating trainees extensively evidenced meaningful perspectives on AI models and CT; Table 1 provides these trainees' background information. Subsequently, four major themes emerged within their perspectives based on the axial coding (Merriam & Tisdell, 2015a): (i) Embrace AI Models to Enhance T&L; (ii) Perceive AI Models as Computational Thinkers; (iii) Employ CT for Effective Prompting of AI Models; (iv) Recognize the Challenges of Adopting AI Models. These themes are discussed along with the qualitative evidence seriatim.

Table 1. Background Information of the Trainees.

Name	Gender	Major	Study Year
Bennett	Male	BEng in Mechanical Engineering	Year 3
Beckett	Male	BSc in Information Management	Year 4 (Final Year)
Blaine	Male	BSc in Chemistry and BEd	Year 4 (Final Year)
Carter	Male	BSc in Quantitative Finance	Year 2
Nolan	Male	BEng in Computer Science	Year 2
Jasper	Male	BASc in Applied AI	Year 2
Ethan	Male	BEng in Computer Science	Year 2
Orion	Female	BEng in Computer Science	Year 2
Graham	Male	BSc in Biological Sciences	Year 1
Mateo	Male	BSocSc in Education	Year 5

Psychology and BEd	(Final Year)

3.1. Embrace AI Models to Enhance T&L

Firstly, the trainees generally perceived AI models to have the capacity to enhance the T&L experience of K–12 students and beyond. These models can facilitate idea generation, provide students with new insights, and enable them to solve problems independently (Bywater et al., 2019; Holstein & Aleven, 2022; Kim et al., 2022). They can catalyze creativity, nudging students towards curiosity and inspiring them to think outside the box. For instance, Mr. Bennett, a Bachelor of Engineering (BEng) in mechanical engineering student, remarked,

These days, students are born into an AI generation. They're surrounded by AI, and it's [an] essential and very useful tool for them to tackle problems in the world for their future. So as a teacher, we must use [or] apply AI in our education, in our teaching. The role is vital. For example, we have to teach our students to get used to AI chat systems and AI language models. These are very useful tools for generating ideas in education.

Similarly, Mr. Beckett, a student of a Bachelor of Science (BSc) in information management, said,

Talking about AI, I think it can certainly help people do things, learn things, or manage things. Integrating these tools into education seems like a great opportunity. Education is about teaching, and AI tools also serve a similar purpose—to help someone find the right way.

Another trainee, Mr. Blaine, studying BSc in chemistry and Bachelor of Education (BEd), expressed,

Well, in my learning, it's a good way that I can learn with AI supplementing me with some very insightful ideas, perhaps some of the insights such as commonly mentioned [insights] or common sense that AI provides me. And also, it may give ideas that I've already thought of. But yes, I don't know why they always bring me some new insights.

3.2. Perceive AI Models as Computational Thinkers

Secondly, some trainees perceived AI models as capable of employing CT when tackling complex problems. These models can systematically approach an overarching, complex problem by breaking it down into smaller, more manageable parts (Kojima et al., 2023; Wei et al., 2022). Thus, they can give an impression of employing CT practices, notably decomposition and abstraction. They can simplify and analyze intricate problems methodically, akin to human reasoning (Huang & Chang, 2023; Qiao et al., 2023). For instance, Mr. Carter, pursuing a Bachelor of Science (BSc) in quantitative finance, observed,

Like you ask ChatGPT, "I have this math problem, how do I solve it?" It's going to break that problem up for you in an easier manner and try to make you understand it. So, it's also using its own sort of CT mindset, you could say, in order to solve the problem that you give it.

Mr. Nolan, a BEng in computer science student, noted, "But if you're talking specifically about AI models, since they're programmed and they are running on CT themselves, in a sense…"

3.3. Employ CT for Effective Prompting of AI Models

Thirdly, the trainees overwhelmingly perceived CT as critical for effectively prompting AI models, which entails instructing and interacting with the models (Arora et al., 2022; X. Liu et al., 2023). They recognized that strategic formulation of prompts (i.e., prompt engineering) could drastically improve the quality of AI-generated outputs. This can lead to more accurate and contextually relevant outputs, mirroring sophisticated, human-like interactions (Clavié et al., 2023; Y. Liu et al., 2023; White et al., 2023). Moreover, they stated that CT can give users an intuitive understanding of AI models, at least at a rudimentary level, helping them heuristically leverage the underlying mechanisms that drive the models' behavior and adaptability. For instance, Mr. Jasper, a student of Bachelor of Arts and Sciences (BASc) in applied AI, stated,

With the invention of ChatGPT and Bing AI, when you run into a problem, if you ask it a question like a human, you're not going to get the best possible answer… But if you're a programmer and you know that this is how it [AI] works, it looks at these words, it makes this connection… You probably will be able to give it a much better prompt and get a much better response. So, I think today, especially, it's more important for everyone in this field to learn CT.

Mr. Ethan, a trainee pursuing a BEng in computer science, suggested, "I think the concept of prompt engineering is a very CT-based concept. You design things in such a manner that the computer or the AI really understands it and does what you want."

Another trainee, Ms. Orion, doing a BEng in computer science, expressed,

I think that computational thinking is the base that is required for individuals to be able to use AI technology… I think prompt engineering is a lot about how to phrase what you say and how to understand the design behind the AI and to understand that it is AI at the end; it's not human. And, as a user, I need to know, like the programmer who designed that AI, "What thinking did they put into building that AI?" And by thinking that it's my CT, I did either knowingly or unknowingly.

Likewise, Mr. Graham, a BSc in biological sciences student, proffered,

In CT, we need to remove all noise. Similarly, when we talk to a chatbot, we need to remove all unnecessary information… I think the student with CT skills will have an advantage. Like, he understands better, at least like, how a computer processes answers, so he'll be able to refine his input better or refine them.

Mr. Mateo, studying a Bachelor of Social Sciences (BSocSc) in education psychology and BEd, remarked,

I think CT would empower students to make better use of AI or different types of chatbots. Like, because our AI is not as smart as a human being yet. When we expect the AI to provide a detailed answer, sometimes it makes mistakes or misunderstands our prompt. CT would help students understand why we should instruct the AI about our tasks in a certain way… How to decompose our task requirements, [and] abstract them, like how to *communicate them to the AI. We can explain our task requirements to the AI in different ways.*

Mr. Bennett said,

I think one aspect is that we need to have CT to understand how AI works. And if we understand how AI works, we can control it and make use of it. For example, we need to know how ChatGPT processes information so we can ask accurate questions and guide the AI model to the correct outcome. Another example is that if we have an AI that generates drawings, we need to understand the algorithm behind it in order to generate a photo or a picture accurately.

Likewise, Mr. Beckett mentioned, "CT can help you to think more like a computer. So, when you have learned CT, you can understand what kind of information to extract from ChatGPT."

Mr. Nolan noted,

I think with CT, you're better able to understand AI models, any machine at that… You don't know how this model was created. You don't know whether you should be afraid of it. You don't know whether you should be fond of it. But I think CT gives you a very basic idea of what's happening behind the scenes.

3.4. Recognize the Challenges of Adopting AI Models

The trainees perceived various challenges when asked about adopting AI models for their current and future T&L. These challenges include (a) limited guidance and experience in integrating AI models within educational settings, (b) conflicting reasons among students when it comes to utilizing AI models, (c) difficulties in designing assessment tasks that ensure transparent and honest use of AI models, and (d) adverse effects on learning transactions in classroom settings. These challenges are evident from the following quotes:

(a) Mr. Bennett acknowledged,

These [AI models] are very useful tools for generating ideas in education. But how to apply AI in our education? Currently, I do not have a very clear concept or idea because I haven't had the chance to apply it.

(b) Mr. Carter expressed,

I would argue that people who are less adept at solving problems would rely more on AI models, or these AI bots, to solve their problems… On the other hand, you could argue that the smartest students, who want to save their time, would also utilize AI. So, I think it just goes both ways, and it's a discussion that could continue indefinitely.

(c) Mr. Blaine relayed,

In teaching parts, it gets more challenging. Because I have to design an assessment that AI can, I won't say cannot, but it is more difficult for AI to finish. Because, yeah, we have ChatGPT. We also have more and more of those language models. So, if I still continue with very simple questions that students can copy and paste.

(d) Mr. Graham noted,

In the past, before when students had a question, they might raise their hand. Or ask teachers or use Google search. But now, I think they're more likely to ask just AI… But when they ask AI, like the AI might not be able to see these problems and [might] provide an answer the student wasn't needing.

4. DISCUSSION

4.1. Latent Synergies Between AI Models and CT

The pre-service teacher trainees' perspectives on AI models and CT reveal latent synergies or interrelationships between the two. On one side, their perspectives recognized that AI models can seemingly deploy CT-like problem-solving when dealing with complex problems (Kojima et al., 2023; Wei et al., 2022; Yao et al., 2023). Conversely, they accentuated the importance of CT for human users to instruct and interact with AI models effectively. These latent synergies are no mere coincidence; it stands to reason that the AI models directly resulted from the CT employed by computer scientists—they are the outcome of CT—writ large (Celik, 2023; Lin et al., 2023; Yilmaz & Karaoglan Yilmaz, 2023). Likewise, since AI models (i.e., transformers) are trained on copious amounts of human data, they have acquired human-like reasoning and problem-solving abilities, such as CT (Huang & Chang, 2023; Qiao et al., 2023). However, one question remains: Why did the trainees recognize CT for effectively instructing and interacting with AI models, or what is the interrelationship between prompting and CT? This question has not been researched (to the best of our knowledge) from either an a priori or a posteriori standpoint. This highlights a research gap, but the question is, why is answering this question significant from a real-world point of view?

4.2. Prompting AI Models: A Novel Approach for Cultivating CT

The current trends in CT education, especially at the K–12 level, can be divided into plugged or unplugged T&L approaches. The plugged approaches include educational robotics, block-based and text-based programming, and digital gamification (e.g., Kong et al., 2020; Rich et al., 2021; Umutlu, 2021). The unplugged approaches include algorithm assembly and pattern recognition puzzles, board games based on CT concepts (e.g., functions, conditionals, iteration), and physical programming robots (e.g., Bell & Lodi, 2019; Delal & Oner, 2020; Ung et al., 2022). Regardless, the elephant in the room remains, as these approaches are arduous to integrate within formal T&L environments across different K–12 subject areas and contexts (Ali, 2021; Angeli & Giannakos, 2020; Lodi & Martini, 2021; Shute et al., 2017). Subsequently, the cultivation of CT has been traditionally relegated to specialized STEM-based lessons, projects, or competitions. So, one may argue that prompting or prompt engineering could emerge as a novel approach that potentially addresses this issue and revolutionizes CT education.

The crux of prompting lies in its fluid adaptability and its necessity to be taught across K–12. As AI models become ever more prevalent, it becomes crucial for both students and teachers to learn how to harness them effectively (Casal-Otero et al., 2023; Chiu, 2023; Lozano & Carolina Blanco, 2023). Recognizing this, a promising opportunity exists to integrate CT into the formal curricula to teach prompting. This hinges on first investigating CT's role as a foundational skill for effective prompting. If rigorously evidenced, this could allow seamless integration of CT across computer-based and noncomputer-based subject areas, mainly thanks to increasingly affordable and ubiquitous access to AI models (ChatGPT, Bing AI, Stable Diffusion, etc.). This could democratize CT education and help overcome its dependence on expensive, specialized equipment. In this manner, CT and its practices could be cultivated regardless of a school's resources, becoming a proper egalitarian prerogative for all K–12 students and teachers.

In brief, prompt engineering's versatility as a novel approach could bridge the divide between CT and general education. It could foster a generation of learners trained to apply CT and its practices across diverse academic disciplines and real-world scenarios. Most importantly, it could prepare them to be AI-ready and excel in an increasingly digital and automated world.

5. CONCLUSION

The study reveals latent synergies between AI models and CT that beckon further exploration in future research. It highlights the potential of prompt engineering as a novel approach to cultivating CT in K–12 education and teacher education. The teacher trainees acknowledged the significance of CT in developing AI models and its necessity for effectively harnessing them. Nevertheless, future studies should pragmatically investigate this interplay, particularly the extent and roles of different CT practices during prompting and their implications across different prompt engineering techniques.

6. REFERENCES

Ali, M. (2021). State of STEM Education in Hong Kong: A Policy Review. *Academia Letters*, *Article 3680*. https://doi.org/https://doi.org/10.20935/AL3680

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*, 100329. https://doi.org/https://doi.org/10.1016/j.ijcci.2021.100329

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, *105*, 106185. https://doi.org/https://doi.org/10.1016/j.chb.2019.106185

Arora, S., Narayan, A., Chen, M. F., Orr, L., Guha, N., Bhatia, K., Chami, I., Sala, F., & Ré, C. (2022). Ask Me Anything: A simple strategy for prompting language models. *arXiv [cs.CL]*. http://arxiv.org/abs/2210.02441

Bell, T., & Lodi, M. (2019). Constructing Computational Thinking Without Using Computers. *Constructivist foundations*, *14*(3), 342-351. (Special Issue "Constructionism and Computational Thinking")

Bywater, J. P., Chiu, J. L., Hong, J., & Sankaranarayanan, V. (2019). The Teacher Responding Tool: Scaffolding the teacher practice of responding to student ideas in mathematics classrooms. *Computers & Education*, *139*, 16-30. https://doi.org/https://doi.org/10.1016/j.compedu.2019.05.004

Casal-Otero, L., Catala, A., Fernández-Morante, C., Taboada, M., Cebreiro, B., & Barro, S. (2023). AI literacy in K-12: a systematic literature review. *International Journal of STEM Education*, *10*(1), 29. https://doi.org/https://doi.org/10.1186/s40594-023-00418-7

Celik, I. (2023). Exploring the Determinants of Artificial Intelligence (AI) Literacy: Digital Divide, Computational Thinking, Cognitive Absorption. *Telematics and Informatics*, *83*, 102026. https://doi.org/https://doi.org/10.1016/j.tele.2023.102026

Chiu, T. K. F. (2023). The impact of Generative AI (GenAI) on practices, policies and research direction in education: a case of ChatGPT and Midjourney. *Interactive Learning Environments*, 1-17. https://doi.org/https://doi.org/10.1080/10494820.2023.2253861

Clavié, B., Ciceu, A., Naylor, F., Soulié, G., & Brightwell, T. (2023). Large Language Models in the Workplace: A Case Study on Prompt Engineering for Job Type Classification. *arXiv [cs.CL]*. http://arxiv.org/abs/2303.07142

Delal, H., & Oner, D. (2020). Developing Middle School Students' Computational Thinking Skills Using Unplugged Computing Activities. *Informatics in Education*, *19*(1), 1-13. https://doi.org/https://doi.org/10.15388/infedu.2020.01

Holstein, K., & Aleven, V. (2022). Designing for human-AI complementarity in K-12 education. *AI Magazine*, *43*(2), 239-248. https://doi.org/https://doi.org/10.1002/aaai.12058

Huang, J., & Chang, K. C.-C. (2023). Towards Reasoning in Large Language Models: A Survey. *arXiv [cs.CL]*. http://arxiv.org/abs/2212.10403

Kim, J., Lee, H., & Cho, Y. H. (2022). Learning design to support student-AI collaboration: perspectives of leading teachers for AI in education. *Education and Information Technologies*, *27*(5), 6069-6104. https://doi.org/https://doi.org/10.1007/s10639-021-10831-6

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., & Iwasawa, Y. (2023). Large Language Models are Zero-Shot Reasoners. *arXiv [cs.CL]*. http://arxiv.org/abs/2205.11916

Kong, S.-C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education*, *151*, 103872. https://doi.org/https://doi.org/10.1016/j.compedu.2020.103872

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory into practice*, *41*(4), 212-218. https://doi.org/https://doi.org/10.1207/s15430421tip4104_2

Lin, X.-F., Zhou, Y., Shen, W., Luo, G., Xian, X., & Pang, B. (2023). Modeling the structural relationships among Chinese secondary school students' computational thinking efficacy in learning AI, AI literacy, and approaches to learning AI. *Education and Information Technologies*. https://doi.org/10.1007/s10639-023-12029-4

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., & Tang, J. (2023). GPT Understands, Too. *arXiv [cs.CL]*. http://arxiv.org/abs/2103.10385

Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., & Liu, Y. (2023). Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *arXiv [cs.SE]*. http://arxiv.org/abs/2305.13860

Lodi, M., & Martini, S. (2021). Computational Thinking, Between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/https://doi.org/10.1007/s11191-021-00202-5

Lozano, A., & Carolina Blanco, F. (2023). Is the Education System Prepared for the Irruption of Artificial Intelligence? A Study on the Perceptions of Students of Primary Education Degree from a Dual Perspective: Current Pupils and Future Teachers. *Education Sciences*, *13*(7), 733. https://doi.org/https://doi.org/10.3390/educsci13070733

Merriam, S. B., & Tisdell, E. J. (2015a). Qualitative Data Analysis. In *Qualitative Research: A Guide to Design and Implementation* (4th ed., pp. 195-236). John Wiley & Sons.

Merriam, S. B., & Tisdell, E. J. (2015b). What is Qualitative Research? In *Qualitative Research: A Guide to Design and Implementation* (4th ed., pp. 3-21). John Wiley & Sons.

Michel-Villarreal, R., Vilalta-Perdomo, E., Salinas-Navarro, D. E., Thierry-Aguilera, R., & Gerardou, F. S. (2023). Challenges and Opportunities of Generative AI for Higher Education as Explained by ChatGPT. *Education Sciences*, *13*(9).

Park, Y., & Shin, Y. (2022). Text Processing Education Using a Block-Based Programming Language. *IEEE Access*, *10*, 128484-128497. https://doi.org/https://doi.org/10.1109/ACCESS.2022.3227765

Priya, S., Bhadra, S., Chimalakonda, S., & Venigalla, A. S. M. (2022). ML-Quest: a game for introducing machine learning concepts to K-12 students. *Interactive Learning Environments*, 1-16. https://doi.org/https://doi.org/10.1080/10494820.2022.2084115

Qiao, S., Ou, Y., Zhang, N., Chen, X., Yao, Y., Deng, S., Tan, C., Huang, F., & Chen, H. (2023). Reasoning with Language Model Prompting: A Survey. *arXiv [cs.CL]*. http://arxiv.org/abs/2212.09597

Rich, P. J., Mason, S. L., & O'Leary, J. (2021). Measuring the effect of continuous professional development on elementary teachers' self-efficacy to teach coding and computational thinking. *Computers & Education*, *168*, 104196. https://doi.org/https://doi.org/10.1016/j.compedu.2021.104196

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

Tedre, M., Toivonen, T., Kahila, J., Vartiainen, H., Valtonen, T., Jormanainen, I., & Pears, A. (2021). Teaching Machine Learning in K-12 Classroom: Pedagogical and Technological Trajectories for Artificial Intelligence Education. *IEEE Access*, *9*, 110558-110572. https://doi.org/10.1109/ACCESS.2021.3097962

Tucker-Raymond, E., Cassidy, M., & Puttick, G. (2021). Science teachers can teach computational thinking through distributed expertise. *Computers & Education*, *173*, 104284. https://doi.org/https://doi.org/10.1016/j.compedu.2021.104284

Umutlu, D. (2021). An exploratory study of pre-service teachers' computational thinking and programming skills. *Journal of Research on Technology in Education*, 1-15. https://doi.org/https://doi.org/10.1080/15391523.2021.1922105

Ung, L.-L., Labadin, J., & Mohamad, F. S. (2022). Computational thinking for teachers: Development of a localised E-learning system. *Computers & Education*, *177*, 104379. https://doi.org/https://doi.org/10.1016/j.compedu.2021.104379

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., & Le, Q. V. (2022). Finetuned Language Models Are Zero-Shot Learners. *arXiv [cs.CL]*. http://arxiv.org/abs/2109.01652

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. *arXiv [cs.SE]*. http://arxiv.org/abs/2302.11382

Xia, X., & Li, X. (2022). Artificial Intelligence for Higher Education Development and Teaching Skills. *Wireless Communications & Mobile Computing (Online)*, *2022*. https://doi.org/https://doi.org/10.1155/2022/7614337

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T. L., Cao, Y., & Narasimhan, K. (2023). Tree of Thoughts: Deliberate Problem Solving with Large Language Models. *arXiv [cs.CL]*. http://arxiv.org/abs/2305.10601

Yilmaz, R., & Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, *4*, 100147. https://doi.org/https://doi.org/10.1016/j.caeai.2023.100147

Assess Computational Thinking in K-12 Students' Mathematics Education: A Scoping Review with Cultural Sensitivity Focus

Yuting HUANG[1*], Yi SHEN[2], Ziyi HUANG[3]
[1,2,3]The University of Hong Kong, Hong Kong SAR, China
yutingh96@gmail.com, yishen_sy@163.com, sansiyuedejiyi@163.com

ABSTRACT
Since computational thinking is regarded as one of the essential skills of the 21st century, many studies have focused on assessing the development of CT in mathematics education. However, there is still no clear discussion on assessing the development of CT in mathematics learning in different cultural contexts and how different cultures influence the assessments of CT. We conduct a scoping review to emphasize how cultures influence CT assessment and how to describe the interplay between CT and assessment tools. Based on the PRISMA-ScR framework, we collected studies in three databases, Scopus, Web of Science, and ERIC, and selected the most relevant studies according to the inclusion and exclusion criteria. Finally, we charted the data in 10 studies and collated, summarized, and reported the results. In our findings, we categorized four categories of assessment tools, educational robot, unplugged activity, CT knowledge test, and interview, and four categories of culture sensitivity, language issue, ethnic issue, traditional culture, and education culture. We suggest that researchers pay much attention to describing and assessing how CT integrates with mathematics thinking, which could help clarify what happens in students' CT development. Besides, we encourage researchers to revise assessments, like translating into the mother language, improve traditional assessment tools, change items unsuitable for participants based on cultural, social, and sociocultural aspects, and be careful about sensitive issues such as ethnicity and equality.

KEYWORDS
Computational thinking, Mathematics education, Assessment, Cultural sensitivity

1. INTRODUCTION
As proposed by Jeannette M. Wing (2006), computational thinking (CT) is a term used for problem-solving and decision-making. It is a way of thinking and approaching complex problems that draws on principles from computer science. Since the concept of CT was popularized, research on how CT promotes mathematics learning has continued to emerge. CT and mathematical learning have already been shown to have a solid connection (Miller, 2019).

Since CT is regarded as one of the essential skills of the 21st century (Wing, 2014), many studies have focused on assessing the development of CT in mathematics education. Still, the assessment tools of CT are diverse, such as unplugged activities, robots, programming, questionnaires and interviews, and the assessment forms vary across different cultural contexts. Recently, reviews have discussed the interplay between CT and mathematics thinking (Ye, Liang, Ng, & Chai, 2023) and the approaches to integrating CT and mathematics (Chan, Looi, Ho, & Kim, 2023). However, there is still no clear discussion on assessing the development of CT in mathematics learning in different cultural contexts and how different cultures influence the assessment of CT. We chose the framework of CT developed by Grover and Pea (2017) mainly because the categories in their framework clearly describe what the cognitive concepts of CT are and what the practices related to CT, which helped us figure out CT assessed in different studies in which the definition of CT is diverse. In order to address this research gap, our study aims to discuss how different assessment tools assess the development of CT in mathematics learning in different cultural contexts.

In this study, we conduct a scoping rather than a systematic review. Although sharing similar protocol, the differences between a systematic review and scoping review are as bellow: systematic review is usually conducted by a group of experts and aims to provide rigorous evidence for making conclusions or suggestions in a particular field; systematic review is more likely to use statistical analysis while scoping review is more likely to synthesize the finding qualitatively (Munn et al., 2018). Therefore, the proper review type for our project should be a scoping review, which fits our research purpose.

In this review, we aim to emphasize and synthesize the assessment of CT in mathematics education from kindergarten to Grade 12 in different contexts. Moreover, we explore the cultural aspect and underline the significance of cultural sensitivity in research. We emphasize how cultures influence CT assessment and how to describe the interplay between CT and assessment tools. Therefore, we have two focuses: the assessment of CT and cultural sensitivity, which drive us to pose the following research questions (RQs):

RQ1: What assessment tools have been used to assess CT in K-12 students' mathematics education?

RQ2: What CT concepts and practices have been assessed?

RQ3: How do the studies discuss cultural sensitivity if they do?

By answering these questions, we want to determine the relationship between different CT concepts and assessment tools, providing suggestions for researchers when choosing assessment tools. Also, we want to concentrate on cultural sensitivity in those studies, trying to remind researchers of cultural aspects while conducting empirical research in the future.

2. METHOD

We conducted a scoping review, which is suitable for searching and identifying key characteristics of a certain topic and examining what earlier research has been done (Munn et al., 2018) to search for related studies that talk about the assessment of CT in mathematics education, synthesize the results and processes in those studies, and discuss our research questions. In this study, we wanted to identify prevalent assessment tools of CT in mathematics education, pay attention to cultural sensitivity that relates to research design, and contribute to future empirical research on the assessment of CT. We followed the five processes of conducting a scoping review: (1) identifying the research question, (2) identifying relevant studies, (3) study selection, (4) charting the data, (5) collating, summarizing, and reporting the results (Arksey & O'Malley, 2005).

2.1. Search and Selection Process

In this study, we used three databases, including Scopus, Web of Science, and ERIC, which cover our research interest in education and contain numerous popular journals and articles, to identify relevant studies. In order to meet our research purpose, we focused on four topics: assessment, computational thinking (CT), mathematics and education. The search string elements (Table 2) were performed in different search fields of different databases (Table 1). We established a publication date restriction of 5 years to ensure the relevance and quality of up-to-date studies, and the last source search was conducted on October 4, 2023. Besides, we choose peer-reviewed articles to ensure the quality of studies. We collected 260 studies from three databases using these research terms and removed 124 duplicated articles. After that, 136 articles were included in the selection process.

According to the PRISMA-ScR framework, we conducted the selection process in Covidence, a convenient website that helps researchers review. There were two rounds of screening, the title and abstract screening and the full-text review, for identifying the relevant empirical studies that talk about the assessment of CT in mathematics education and filtering out the inappropriate articles. The reason for choosing empirical research is that it is backed up by evidence of practice results rather than relying on theoretical derivation to reach conclusions. We developed inclusion criteria aligned to our research questions in this selection process: (a) available in English, (b) available in full paper, (c) presenting adequate data and discussion, (d) assessment design that included CT in mathematics or STEM education, and (e) serving participants are K-12 students (Table 3). In each screening round, two of us screened the same article independently. We negotiated for consensus with each other to make sure that these articles needed to be included in our discussion.

Table 1. Research Field of Database.

Database	Search Field
Scopus	Article title, abstract, and keywords.
Web of Science	Topic (Searches title, abstract, author, keywords, and keywords plus).
ERIC	Title, author, source, abstract and descriptor.

Note. When searching in ERIC, we click "peer-reviewed only". All articles included in Scopus are peer-reviewed articles.

Table 2. Search Terms.

Topic	Search String Elements
Assessment	Assess*
AND	
Computational thinking	"computational thinking"
AND	
Mathematics	mathematics OR math OR STEM
And	
Education	"primary school" OR "preschool" OR "kindergarten" OR "pre-k" OR "secondary school" OR "high school" OR "junior high school" OR "senior high school" OR "K-12" OR "pretertiary" OR "elementary school"
Publication year	2019 - 2023

Table 3. Inclusion and exclusion criteria.

Screen	Inclusion criteria	Exclusion criteria
Type of record	Full text available	Full text is not available
Population	K-12 (Kindergarten to grade 12)	post-secondary education, university students
Language	Written in English	Not written in English
Subject	Mathematics, STEM (focus on mathematics)	The study relates to other subjects, except mathematics
Data	Complete Data (means that the article is reliable)	Without complete data and analysis of the data
Research type	Empirical study	Not empirical study
CT	At least one of the CT concepts and practices are mentioned	Mention CT but do not mention specific concepts or practice
Assessment Tools	The study uses specific assessment tools to assess CT and has an analysis of it	The study mentions assessment, but it assesses mathematics, not CT

According to the inclusion and exclusion criteria, a total of 106 articles were filtered for exclusion in the first screening round. Then, we removed 20 studies that were not closely relevant to our research when reviewing the full texts of those studies. Finally, 10 studies were included in our discussion. The selection process is presented in Figure 1.

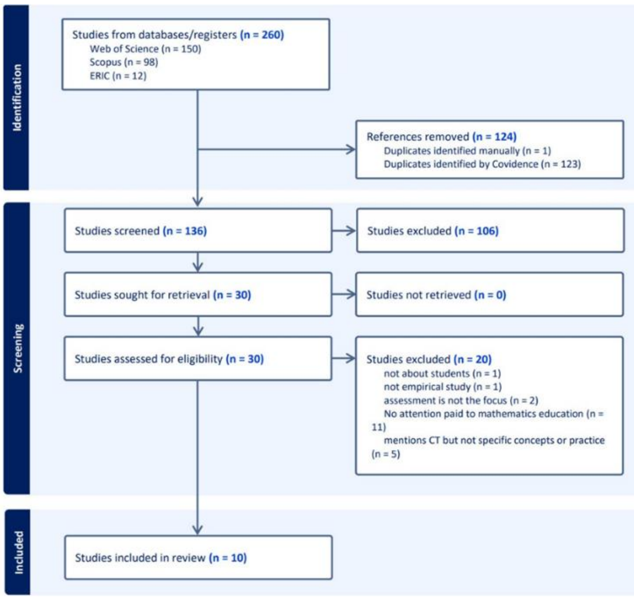


Figure 1. Paper Selection Procedure.

2.2. Data Extraction and Analysis

After selection, we finally got 10 articles for the analysis process. According to our research questions, we settled our focus on mathematics, CT, assessment tools and cultural sensitivity and extracted related information accordingly. Except for those four focus items, we extracted the basic information about every study, such as title, publication year, research design, and participant description, from 10 selected articles. Two of us extracted data from the same study independently and then negotiated with each other for consensus. Finally, we form the extraction data table, containing basic and important information relating to our research questions, and ready to do the next step: data analysis.

We conducted a "top-down" theoretical thematic analysis (Braun & Clarke, 2006) since we facilitated an existing framework of CT (Grover & Pea, 2017) to analyze the data we extracted. Meanwhile, we generated our categories about assessment tools and cultural sensitivity. Similar to the selection and extraction processes, disagreements were resolved through further review and discussion until a consensus was reached. First, we used descriptive codes to label data which could be the answers to our research questions. Then, we categorized CT, adapting the existing framework from Grover and Pea, and categorized assessment tools and cultural sensitivity by generating new categories according to the descriptions of assessment tools and cultural aspects mentioned in selected articles. Finally, we got three themes: CT, assessment tools, and cultural sensitivity (Table 4).

Table 4. Coding System

Theme	Category	Code
CT Concept	Logic and logical thinking	Logically analyzing and organizing data, Logical thinking, etc.
CT Practice	Algorithms and algorithmic thinking	Calculations, Algorithmic thinking, etc.
	Patterns and pattern recognition	Sequence, Pattern recognition, etc.
	Abstraction and generalization	Generalizations, Abstraction, etc.
	Automation	N/A
	Problem decomposition	Problem decomposition, Problem-solving, etc.
	Creating computational artefacts	Create new artifacts, Design complex systems, etc.
	Testing and debugging	Debugging, Repeat the reasoning, coding, and observation processes, etc.
	Iterative refinement	N/A
	Collaboration & Creativity	Create and express themselves. Creativity and collaboration.
Assess-ment	Educational robots	Robotics kits within educational settings, Lego Education WeDo 2.0 kit, etc.
	Unplugged activities	Unplugged robotics activities, Bebras Cards, etc.
	CT knowledge tests	Computational thinking ability test, Yune Tran's CT questionnaire, etc.
	Interview	Cognitive interview, Semi-structured interviews, etc.
Culture Sensitivity	Language issue	Attached to English notations, etc.
	Ethnic issue	Student population with racial issues.
	Traditional culture	Traditional Yupana, Validation in a Latin American population, etc.
	Education culture	Korean Nuri curriculum, Content covered under the Korean National Curriculum for kindergartens.

3. RESULTS

3.1. Assessment Tools Used to Assess CT in K-12 Students' Mathematics Education

Table 5 summarizes the assessment tools used in the 10 selected articles. Most of the assessment tools were designed for middle school (n=4) students, followed by primary school students (n=3) and kindergarten students (n=3). The assessment tools can be divided into four types: educational robot, unplugged activity, CT knowledge test, and interview, with almost all studies using two or more assessment tools.

Table 5. The Details of the Reviewed Articles.

Year	Article	Grade Level	Assessment Tool(s)	Mathematics
2023	Angraini et al. (2023)	Seventh-grade students	A CT ability test; Interviews.	Flat shapes.
2022	Luo et al. (2022)	Fourth grade students	Standardized paper-and-pencil test; Cognitive interviews.	Variables.
2023	Sung et al. (2023)	5- and 6-year-old students	Bebras Cards; TACTIC-KIBO.	Nuri curriculum (STEM).
2019	Chiazzese et al. (2019)	Third- and fourth-grade students	The Bebras tasks; Robotics kits (the Lego Education WeDo 2.0 kit).	STEM (angle and distance).
2021	Chan et al. (2021)	Secondary 1 students	Unplugged Math+C activities; Spreadsheet.	Number patterns.
2022	Alvarado et al. (2022)	Elementary students	Digital Yupana; A CT test by Roman Gonazalez et al. (2017).	Arithmetic operations.
2021	Polat et al. (2021)	Fifth- and sixth-grade secondary school students	CT Test (CTt); CT Levels Scale (CTLS).	Math courses
2023	Møller and Kaup (2023)	Three boys aged 13, 14 and 15; One girl aged 12	The DJI RoboMaster S1 educational robot; Semi-structured interviews.	Coordinate systems, translation, and rotation.
2021	Gerosa et al. (2021)	Level 5 students (kindergarten)	Yune Tran's CT questionnaire; Tablet based test; Educational robotics task	Sequencing skills and numerical abilities.
2019	Nam et al. (2019)	Kindergarten students (age from 5 to 6 years)	TurtleBot; Problem-solving performance instrument (test)	Categorization, patterns, numbering, measuring, statistics, diagram.

Educational Robot. Educational robots refer to robots used in learning activities, which can be physical robots, modular systems, or robots built for programmed activities (Gubenko, Kirsch, Smilek, Lubart, & Houssemand, 2021). Educational robots are one of the most widely used assessment tools in the selected articles (n=5) and are used in kindergarten, elementary, and middle school. Robots in preschool are generally shaped like animals, and students can control the movements of robots through action commands to complete different test questions, like problem-solving performance in mathematics (Nam, Kim, & Lee, 2019). Educational robots used in elementary and junior high school evaluations may include two major steps of robot assembly and programming, and the operation is relatively more complex and can be used in STEM courses (Chiazzese, Arrigo, Chifari, Lonati, & Tosto, 2019; Møller & Kaup, 2023).

Unplugged Activity. This kind of activity does not involve the use of computers and usually uses physical objects in the form of games or competitions to stimulate students' interest in learning (Tonbuloğlu & Tonbuloğlu, 2019), such as cards and worksheets with cartoon elements including different levels of tasks that evaluate CT concepts in STEM programs (Sung, Lee, & Chun, 2023).

CT Knowledge Test. This kind of assessment usually adopts more traditional testing methods (n=6), such as single-choice, multiple-choice, and open questions (Flanigan, Peteranetz, Shell, & Soh, 2017). Most CT knowledge tests are based on tests designed by others with validated validity, adapted or directly translated (Alvarado, Falcon, Gutiérrez-Cárdenas, & Romero-Romero, 2022; Gerosa, Koleszar, Tejera, Gomez-Sena, & Carboni, 2021; Nam et al., 2019; Polat, Hopcan, Kucuk, & Sisman, 2021), using a variety of media, some using traditional paper-and-pencil methods, and some using electronic devices to assess the basic arithmetic operations such as addition and subtraction.

Interview. It can help researchers deeply analyze the opinions and experiences of research subjects and better understand the research field of CT that has not been fully explored (Li, 2021). Few studies in the selected articles use interviews (n= 3), and they are generally used as an aid to other assessment methods, such as clarifying students' answers through interviews and more accurately analyzing their CT ability (Angraini, Yolanda, & Muhammad, 2023). Or it is used to analyze aspects of cognition that are difficult to measure with other assessment tools (Luo, Yan, Liu, & Israel, 2022).

In case two or more authors are from different institutions, type the corresponding author's institutional affiliation below the first author. In case two or more authors are at the same institution, use the same institutional affiliation just as it would for one author. Examples of three authors are demonstrated at the top of the page.

3.2. CT Concepts and Practices Being Assessed

Figure 2 shows that the selected articles evaluated CT concepts more frequently in terms of algorithms and algorithm thinking (n=10) and patterns and pattern recognition (n=8), while automation was not evaluated at all. The most evaluated CT practices are Problem composition (n=8) and Testing and debugging (n=5), while Iterative refinement has not been evaluated. Overall, the concept of CT is the main objective of mathematics learning intervention and evaluation in the K-12 stage, while CT practice has been relatively overlooked.

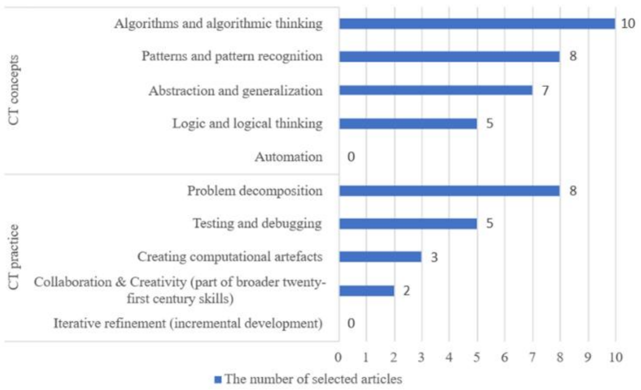


Figure 2. CT Concepts and Practices Assessed in the Selected Articles.

3.3. Cultural Sensitivity Discussed in Current Studies

According to data analysis, we categorize culture sensitivity discussed in selected articles into four categories: Language issue, Ethnic issue, Traditional culture, and Education culture.

Language. A number of studies have focused on cultural differences between languages and have therefore adapted assessment tools and the language used in the assessment process, i.e., Sung et al. (2023) utilized tasks translated into Korean.

Ethnic. According to some research, ethnicity may influence how assessments are conducted and designed. We realize that there is a gap in current research comparing different ethical populations. Especially using the same assessment tools to compare the outcomes and discuss whether ethnicity may affect the conduct of the assessment. Gerosa et al. (2021) found that it has undergone validation in a Latin American population and has shown stability across time and cultures. However, due to the lack of comparative studies of other ethnic groups, we cannot deduce whether this is a specific or a general phenomenon among Latin American ethnic groups. Though one study mentioned the diversity of its population groups (Luo et al., 2022), it did not adjust the assessment tools or discuss the outcomes. We think that it is especially important in multiethnic contexts.

Traditional culture. One research focus is on Tawa Pukllay (Alvarado et al., 2022), which is unique in Peru. Tawa Pukllay of Prem considers the basic arithmetic operations. Alvarado et al. (2022) utilized it to develop a digital Yupana for this study. Yet, they have not infused this traditional cultural factor into their assessment process or adapted their assessment tools. This study inspires us that traditional culture can be applied to CT teaching and assessment. For example, China has many traditional mathematics manipulatives and teaching methods of CT, i.e., Klotski and Abacus, which are overlooked, so researchers can integrate these traditional elements with cutting-edge research, making the CT assessment tool more culturally sensitive and improving its accuracy and intrigue.

Education culture. There are many variations in educational contexts in different settings, so the challenges posed by factors such as educational policy must be considered when assessing CT. The Turkish researchers adapted the assessment tool to their population and revised it by two IT teachers and four university academic experts (Polat et al., 2021). Korean researchers adapted items from the original instrument to represent the content covered under the Korean National Curriculum for kindergartens (Nam et al., 2019).

4. DISCUSSION AND CONCLUSION

CT is a broad term. The public may agree with the definition “the thought processes involved in formulating problems and their solutions”, mentioned by Wing (2011). However, due to many existing improved definitions and theoretical frameworks of CT, it becomes difficult to define the standard version of the contents of CT. When conducting data analysis, we also found that it is difficult to evaluate the concepts and practices of CT in practice accurately, and the evaluation elements of existing research are also challenging to summarize under the same framework. Besides, our framework has two components, automation and iterative refinement, which are hardly being assessed. Therefore, there is an urgent need to produce an authoritative standardized CT framework that could be assessed worldwide.

When an authoritative and widely accepted CT framework emerges, adjusting assessment tools with the new one is suitable. In our finding, there are four categories of assessment tools, and many researchers would conduct two or more of them. However, these assessments are mainly used quantitatively. The advantages of qualitative methods like interviews are not fully discussed in selected articles that conduct mixed-method research. We suggest that researchers pay much attention to describing and assessing how CT integrates with mathematics thinking, which could help clarify what happens in students' CT development. Besides, cultural sensitivity should be emphasized during the experiment design and assessment since the cultural aspect could influence the assessment result. We encourage researchers to revise assessments, like translating into the mother language, improve traditional assessment tools, change items unsuitable for participants based on cultural, social, and sociocultural aspects, and be careful about sensitive issues such as ethnicity and equality.

5. REFERENCES

Alvarado, L., Falcon, C., Gutiérrez-Cárdenas, J., & Romero-Romero, V. S. (2022). Teaching of the Yupana with the Tawa Pukllay method for developing the Computational Thinking in children. Presented at the EDUNINE - IEEE World Eng. Educ. Conf.: Rethink. Eng. Educ. After COVID-19: A Path New Normal, Proc., Institute of Electrical and Electronics Engineers Inc. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85135413911&doi=10.1109%2fEDUNINE53672.2022.9782386&partnerID=40&md5=226a59aa4cfe1bc4092356f2cf7df02b

Angraini, L. M., Yolanda, F., & Muhammad, I. (2023). Augmented Reality: The Improvement of Computational Thinking Based on Students’ Initial Mathematical Ability.

Int. J. Instr., *16*(3), 1033–1054. Gate Association for Teaching and Education.

Arksey, H., & O'Malley, L. (2005). Scoping studies: Towards a methodological framework. *International Journal of Social Research Methodology*, *8*(1), 19–32. Routledge.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101.

Chan, S.-W., Looi, C.-K., Ho, W. K., & Kim, M. S. (2023). Tools and Approaches for Integrating Computational Thinking and Mathematics: A Scoping Review of Current Empirical Studies. *Journal of Educational Computing Research*, *60*(8), 2036–2080.

Chiazzese, Arrigo, Chifari, Lonati, & Tosto. (2019). Educational Robotics in Primary School: Measuring the Development of Computational Thinking Skills with the Bebras Tasks. *Informatics*, *6*(4), 43.

Flanigan, A. E., Peteranetz, M. S., Shell, D. F., & Soh, L.-K. (2017). Implicit intelligence beliefs of computer science students: Exploring change across the semester. *Contemporary Educational Psychology*, *48*, 179–196. Elsevier.

Gerosa, A., Koleszar, V., Tejera, G., Gomez-Sena, L., & Carboni, A. (2021). Cognitive abilities and computational thinking at age 5: Evidence for associations to sequencing and symbolic number comparison. *COMPUTERS AND EDUCATION OPEN*, *2*.

Grover, S., & Pea, R. (2017). Computational Thinking: A Competency Whose Time Has Come.

Gubenko, A., Kirsch, C., Smilek, J. N., Lubart, T., & Houssemand, C. (2021). Educational robotics and robot creativity: An interdisciplinary dialogue. *Frontiers in Robotics and AI*, *8*, 662030. Frontiers.

Li, Q. (2021). *Computational thinking and teacher education: An expert interview study. Human Behavior and Emerging Technologies, 3 (2), 324-338*.

Luo, F., Yan, W., Liu, R., & Israel, M. (2022). Elementary Students' Understanding of Variables in Computational Thinking-Integrated Instruction: A Mixed Methods Study (Vol. 1, pp. 523–529). Presented at the SIGCSE - Proc. ACM Tech. Symp. Comput. Sci. Educ., Association for Computing Machinery, Inc. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85126097345&doi=10.1145%2f3478431.3499323&partnerID=40&md5=64dcae859a411827c1736bcfe0bbd410

Miller, J. (2019). STEM education in the primary years to support mathematical thinking: Using coding to identify mathematical structures and patterns. *ZDM*, *51*(6), 915–927.

Møller, A. K., & Kaup, C. F. (2023). Parents' and children's learning when collaborating on inquiry-based mathematics and computational thinking tasks. *J. Pedag. Res.*, *7*(2), 108–126. Duzce University, Faculty of Education.

Munn, Z., Peters, M. D. J., Stern, C., Tufanaru, C., McArthur, A., & Aromataris, E. (2018). Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach. *BMC Medical Research Methodology*, *18*(1), 143.

Nam, K., Kim, H., & Lee, S. (2019). Connecting Plans to Action: The Effects of a Card-Coded Robotics Curriculum and Activities on Korean Kindergartners. *ASIA-PACIFIC EDUCATION RESEARCHER*, *28*(5), 387–397.

Polat, E., Hopcan, S., Kucuk, S., & Sisman, B. (2021). A comprehensive assessment of secondary school students' computational thinking skills. *Br J Educ Technol*, *52*(5), 1965–1980. John Wiley and Sons Inc.

Sung, J., Lee, J. Y., & Chun, H. Y. (2023). Short-term effects of a classroom-based STEAM program using robotic kits on children in South Korea. *Int. J. STEM Educ.*, *10*(1). Springer Science and Business Media Deutschland GmbH. Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-85152714441&doi=10.1186%2fs40594-023-00417-8&partnerID=40&md5=b63a9856b98de029b207fb790c7596bd

Tonbuloğlu, B., & Tonbuloğlu, İ. (2019). The effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education*, *18*(2), 403–426. Vilnius University Institute of Data Science and Digital Technologies.

Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why? *The magazine of the Carnegie Mellon University School of Computer Science*.

Wing, J. M. (2014). Computational thinking benefits society. *40th anniversary blog of social issues in computing*, *2014*, 26. Academic Press New York, NY, USA.

Ye, H., Liang, B., Ng, O.-L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, *10*(1), 3.

Promoting Undergraduates Computational Thinking in a Knowledge Building Environment

Zuokun Li[1*], Shaoming Chai[2*], Zhenhai He[3]
[1]University of Macau, Macao
[2,3]South China Normal University, China
[1]yc27117@umac.mo, [2]charmingchainet@gmail.com, [3]zhenhai@m.scnu.edu.cn

ABSTRACT
Promoting undergraduates computational thinking (CT) remains a significant challenge. The purpose of this study was to foster students' CT in a knowledge building (KB) environment. The participants were 49 college students from a Chinese university. They were randomly assigned to ten groups. Each group used Knowledge Forum (a computer-supported collaborative knowledge-building environment), supported by customized scaffolding, to discuss and develop the computational artifacts of their choice. Data were collected from student groups' online discourse, pre-post CT scale, and group artifacts. We employed Friedman test, Wilcoxon signed rankings test, and Kruskal-Wallis test to analysis data. In general, we found KB activities facilitated students' CT. Engaging in KB activities can significantly promote students' algorithmic thinking, cooperativity, critical thinking, and problem-solving skills, except for their creativity. Moreover, students demonstrated differential engagement in various CT tasks, as reflected by their discourse and products, and needed more support in evaluation & generalization. This study can offer insights for educators to design KB environment to support students' CT.

KEYWORDS
Computational thinking, Knowledge building, Knowledge Forum

1. INTRODUCTION
Due to the rapid development of information and intelligent technologies and the educational goals of preparing all citizens for the 21st century literacy, computational thinking (CT) has been discussed and greatly emphasized in many countries' educational reform policies in the last decade (Denning & Tedre, 2019; Nouri et al., 2020; Yadav et al., 2016). CT draws on concepts which are fundamental to computer science (CS) (Wing, 2006), and is broader than CS as it 'includes a way of thinking about everyday activities and problems' (Shute et al., 2017, p. 146). The acquisition of CT is widely discussed in K-12 education, and numerous studies are exploring effective teaching methods at this level. However, at higher education level, research on promoting students CT is still lagging behind (Lu et al., 2022; Tivka & Tambouris, 2021). Most attention to computing may be focused on the undergraduates majoring in computing and engineering. Students in non-CS majors are exposed to basic computer courses in China. However, such courses emphasize computer operational skills, yet there is less attention on developing students' CT and considering it as a kind of universal thinking. Obviously, CT should be a thinking skill for all undergraduates in all professional fields, be embedded into the classes for students of all majors, and become a thinking ability to guide their daily lives and works (Sun et al., 2022). Hence, integrating CT into undergraduate curriculums, thereby benefiting students from diverse majors, presents a substantial challenge. It becomes imperative to study the theoretical and pedagogical foundations in this area to better support educators.

Knowledge Building (KB) provides a framework for teaching students how to work together creatively and effectively to advance group knowledge. It is regarded as a theoretical, pedagogical, and technological innovation that holds prospects for promoting students' CT (Bereiter,2020; Soliman, 2020; Zhu et al., 2023). Recent studies provide theoretical interpretations for understanding the potential of KB in fostering CT (Bereiter,2020; Soliman, 2020). Some studies also yield positive results, indicating that KB can advance CT (Soliman, 2020; Zhu et al., 2023). However, the nuances of KB environment design and the systematic evidence-based research have been insufficiently explored. This study aims to mitigate the gap, by introducing a KB environment designed to foster CT. We evaluate the effectiveness through its effects on both students' CT processes and outcomes. The following three research questions were investigated:

RQ1: How do undergraduates engage in CT tasks during their online inquiry in the KB environment?

RQ2: How does participation in KB activities influence undergraduates' CT skills in the KB environment?

RQ3: What is the overall quality of computational artifacts designed by students in the KB environment?

2. METHOD
The study adopted a case study approach to assess the effect of engaging a class of college students in online knowledge building activities on their CT. The participants of this study included 49 undergraduate students (37 males, 12 females) aged between 18 and 22 years old. They enrolled in the course "Computational Thinking and Problem Solving" over a period of 12 weeks at a key university in Guangzhou, China. The subjects were from several schools of the university, and they majored in artificial intelligence, cultural industry management, electronic information engineering, financial management, and software engineering. This course was designed to develop students' computational thinking through engaging in creative idea improvement activities. Participants were randomly divided into ten different groups (G1 to G10) through free teaming with each group averaging 4-6 participants. The instructor had a Ph.D. in education technology and six years of teaching experience using the

KB approach. One teaching assistant with a background in CS and education technology also participated in the students' online activities, providing them with feedback.

In this study, we collected data from three sources: (1) notes recorded in KF, (2) individual survey regarding their CT skills, and (3) the final group artifacts. We gathered student notes only from Phase 3 and Phase 4 of the course, as these phases highlight the application of CT. It enables us to understand student engagement in CT tasks. We employed Friedman test, Wilcoxon signed rankings test, and Kruskal-Wallis test to analysis data.

Throughout the creation process, students were introduced to the five CT tasks to help them design their products. Echoing our earlier statements, the five tasks follow the frameworks of Città et al. (2019) and Selby and Woollard (2013). Analyses focusing on these five tasks, as reflected in the students' discussions in KF, helped to answer the first research question. Two researchers first reviewed and discussed the notes, identifying CT task keywords/key phrases based on task definitions and student-used CT scaffolds. A coding scheme consisting of the CT keywords for the five tasks was developed. Using the coding scheme, two researchers coded all the notes of the nine groups together to ensure reliable coding results. We calculated the average agreement using the Kappa coefficient (Kappa = .920 > .750), which indicated a good consistency of coding results. Then, they discussed the disparities in their understanding and eventually agreed on each note. Finally, the results were represented in terms of frequency, and the Friedman test was then applied to ascertain if significant differences existed in student engagement across five CT tasks.

The Chinese version of the Computational Thinking Scales (CCTS), validated by Sun et al. (2022), was employed to assess students' CT skills, with consideration for the cultural context. It contained a total of 24 items divided into five subdimensions: creativity (7), algorithmic thinking (6), cooperativity (4), critical thinking (4), and problem-solving (3). The total questions were scored on a 5-point Likert scale ranging from "strongly disagree" (1) to "strongly agree" (5). The Cronbach's alpha value for the scale was .852 (greater than 0.800). The coefficients of each factor were: creativity (0.846), algorithmic thinking (0.821), cooperativity (0.892), critical thinking (0.725), and problem-solving (0.820), respectively. The values showed that the revised CTS also maintained high levels of reliability, which could be adopted to evaluate the CT skills of Chinese undergraduates effectively. This instrument was administered to the students as a pre-test prior to their engagement in the KB activities and subsequently as a post-test upon completion of the course. On the basis of delegate code, 45 questionnaires were matched for pre and post tests. The Shapiro–Wilk test, applied to the pre-test data, yielded a value of .848 ($p < .001$), indicating a deviation from normal distribution. The Wilcoxon signed-rank test was then applied to ascertain significant changes in students' CT skills after participating in KB activities. In addition, the effect size (r) was calculated. The guidelines (Cohen, 1988) for interpreting this value are: .10 = small effect, .30 = moderate effect, .50 = large effect.

Students' group artifacts were scored with the rubric, which was developed based on a previous relevant study (Atmatzidou & Demetriadis, 2016; Wang, 2023). It consists of the following three scoring categories: abstraction & decomposition, data practices, algorithmic thinking, automation, and evaluation & generalization. The criterion column explains the evaluation criterion for each dimensional CT skill in detail. The project will get a high score, up to 100 points, if it meets all the evaluation criteria. Two graduate students who were familiar with the graded criterion instrument on CT skills graded students' projects separately before reaching an agreement. Similarly, Kappa coefficient was employed and indicated a good agreement between the two coders (Kappa = 0.71). For those inconsistent results, two coders negotiated until reached an agreement, resulting in the final score for each group.

3. RESULTS

3.1. Engagement of undergraduates in CT tasks

The Friedman test indicates there was significant differences in student engagement across the five CT tasks ($\chi 2$ (4) = 21.057, $p < .001$). Dunn-Bonferroni post hoc tests were carried out and there were significant differences between AD-AMS ($p = .011$), AD-EG ($p = .011$), DP-AMS ($p = .019$), and DP-EG ($p = .019$) pairs after Bonferroni adjustments. No significant differences were observed in students' efforts between any other tasks.

3.2. Changes in undergraduates' CT skills

It was found that the scores of the students' CT scale on creativity, algorithmic thinking, cooperativity, and critical thinking dimensions and the total scores of CT showed statistically significant differences before and after participating KB [z (algorithmic thinking) = -4.251; z (cooperativity) = -4.424; z (critical thinking) = -3.833; z (overall scores) = -4.756]. When the average and totals of difference scores are taken into consideration, it is seen that these observed differences are in favor of positive rankings, i.e. posttest scores. The effect sizes of these determined differences were found to be high [r (algorithmic thinking) = .634; r (cooperativity) = .660; r (critical thinking) = .571; z (problem-solving) = -2.931; r = 0.437; r (overall score) = .709). However, students' levels of creativity did not exhibit significant differences pre- and post-activity engagement [z (creativity) = -1.939; r = 0.290].

3.3. Performance of groups' artifacts

We looked into the final computational artifacts students designed in the KB environment. These products were designed based on related products the students encountered in their daily life. For example, one group designed an online campus navigation map to help students or visitors plan the shortest route. We used a rubric to score each product design in terms of AD, DP, AT, AMS, and EG. Kruskal-Wallis test revealed significant difference among the five sub-skills (H (4) = 27.55, $p < .05$). We conducted post hoc tests using Dunn-Bonferroni. There were no significant differences between AD-DP, AD-AT, DP-AT, DP-AMS, AT-AMS, and AMS-EG pairs after Bonferroni adjustments. However, there was a statistically significant between AD-AMS (p = .021), AD-EG (p < .001), DP-EG (p = .001), and AT-EG (p = .007).

4. DISCUSSION & CONCLUSION

This study aimed to explore the impact of a KB environment on undergraduates' computational thinking (CT) skills. The results revealed a significant enhancement in algorithmic thinking, cooperativity, critical thinking, and problem-solving abilities among students. These findings underscore the effectiveness of the KB approach in fostering essential components of CT in an academic setting.

However, an unexpected outcome was the limited improvement in students' creativity. This suggests that while the KB environment effectively promotes algorithmic thinking, additional strategies might be needed to equally enhance creative thinking. Future iterations of the program could incorporate more open-ended tasks and encourage innovative approaches to problem-solving.

The study also noted varying levels of student engagement across different CT tasks. This variation provides valuable insights into how different aspects of CT are received by students and where more instructional support may be needed. Educators should consider these variations when designing and implementing KB activities, ensuring a balanced approach that addresses all facets of CT.

The integration of CT into undergraduate education is crucial in today's technologically driven world. The findings of this study affirm that KB environments can play a significant role in developing these skills. While the approach showed positive results in enhancing certain CT skills, it also highlighted areas for improvement, particularly in fostering creativity and ensuring uniform engagement across all CT tasks.

Future research should focus on refining KB strategies to address these areas, thereby offering a more comprehensive approach to CT education. Despite its limitations, this study provides a foundation for further exploration into the effective integration of CT in educational settings, ultimately contributing to the development of well-rounded, competent graduate.

5. REFERENCES

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. ACM Inroads, 2(1), 48–54.

Bereiter, C., & Scardamalia, M. (2014). Knowledge Building and knowledge creation: One concept, two hills to climb. In S. C. Tan, H. J. So, & J. Yeo (Eds.), Knowledge Creation in Education (pp. 97-118). Springer.

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, 1–25.

Chai, S., & Zhu, G. (2021). The relationship between group adoption of Knowledge Building Principles and performance in creating artifacts. Educational Technology Research and Development, 69(2), 787–808.

Chan, C. K. K., & Van Aalst, J. (2004). Learning, assessment and collaboration in computer-supported environments. In J.-W. Strijbos, P. A. Kirschner, & R. L. Martens (Eds.), What We Know About CSCL (pp. 87–112). Springer Netherlands. https://doi.org/10.1007/1-4020-7921-4_4

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. International Journal of STEM Education, 10(1), 47.

Città, G., Gentile, M., Allegra, M., Arrigo, M., Conti, D., Ottaviano, S., Reale, F., & Sciortino, M. (2019). The effects of mental rotation on computational thinking. Computers & Education, 141, 103613.

Denning, P. J., & Tedre, M. (2019). Computational thinking. The MIT Press.

Ezeamuzie, N. O., Leung, J. S. C., & Ting, F. S. T. (2022). Unleashing the Potential of Abstraction From Cloud of Computational Thinking: A Systematic Review of Literature. Journal of Educational Computing Research, 60(4), 877–905.

Hong, H.-Y., Chen, F.-C., Chai, C. S., & Chan, W.-C. (2011). Teacher-education students' views about knowledge building theory and practice. Instructional Science, 39(4), 467–482.

Hong, H.-Y., & Lin, P.-Y. (2019). Elementary students enhancing their understanding of energy-saving through idea-centered collaborative knowledge-building scaffolds and activities. Educational Technology Research and Development, 67(1), 63–83.

Hong, H.-Y., Lin, P.-Y., Chai, C. S., Hung, G.-T., & Zhang, Y. (2019). Fostering design-oriented collective reflection among preservice teachers through principle-based knowledge building activities. Computers & Education, 130, 105–120.

Hong, H.-Y., Lin, P.-Y., Chen, B., & Chen, N. (2019). Integrated STEM Learning in an Idea-centered Knowledge-building Environment. The Asia-Pacific Education Researcher, 28(1), 63–76.

A Systematic Literature Review of Computational Thinking Evaluation Research in China

Mengyao Wang [1],Litao ZHANG [2]*,Xinning Wang[3]
[1,2,3]chool of Education, Inner Mongolia Normal University, Hohhot
wmy2762894835@163.com,zh-litao@163.com,1243474812@qq.com

ABSTRACT
As Computational Thinking (CT) has received increasing attention in the field of education, the demand and interest in how to evaluate computational thinking skills has increased. This study adopts a systematic literature review method, following the PRISMA procedure to screen 27 domestic computational thinking evaluation related literature, systematically reviewing the way of evaluating CT in the literature in China using the content analysis method to integrate the data, and summarizing the current status of research in the field of evaluation of computational thinking in the country, and the focus of the research. "Evaluation method: learning outcome-oriented, reconstructing the multiple evaluation index system", in order to provide reference to the research in the field of computational thinking evaluation and enrich the relevant theoretical research results.

KEYWORDS
computational thinking, evaluation, computational thinking evaluation, systematic Literature Review

1. INTRODUCTION
Since Zhou Yizhen proposed computational thinking in 2006, computational thinking education has gradually received wide support and attention from the international education field. Many countries in the Americas, Europe and Asia have issued policy documents on computational thinking in order to develop computational thinking education in K-12. 2011, the United States incorporated computational thinking into the CSTA K-12 Standards (2011 Revision); 2013, the United Kingdom incorporated computational thinking into the New Curriculum Program; 2015, Australia included computational thinking as an important part of the New Curriculum Program; and 2012, Australia included computational thinking as an important part of the New Curriculum Program. "In 2013, the United Kingdom included computational thinking in the New Curriculum Plan; in 2015, Australia included computational thinking as an important part of the New Curriculum Program; and the Ministry of Education released the Information Technology Curriculum Standards for General Senior Secondary Schools (2017 Edition) and the Compulsory Education Information and Technology Curriculum Standards (2022 Edition), both of which explicitly mention computational thinking as the core of the information technology (science and technology) curriculum. one of the core literacies of the information technology (science and technology) curriculum. Computational thinking is a kind of thinking, which is a thinking process of problem solving, system construction and human behavior understanding by using the basic concepts and methods of computational science, combining engineering thinking, mathematical thinking and other thinking styles and characteristics. Evaluation, as an important part of computational thinking education, enables students to monitor and evaluate the learning outcomes of computational thinking in a timely manner, thus promoting the effective occurrence of computational thinking teaching. Evaluation of computational thinking education is the weak link in the current stage of research, and it is the focus of future research. In the process of promoting the development of computational thinking education, the role of computational thinking evaluation is very important, which is both an important part of the development of computational thinking and an important basis for verifying the effectiveness of the training (Yu et al., 2020).

Based on this, this paper will attempt to conduct a systematic literature review of computational thinking evaluation in China in order to explore the current research status and research focus in this field. This study can help relevant researchers to understand the research content of computational thinking evaluation more comprehensively and gain comprehensive insights into the topic; to position their research in the overall research picture, to discover the connection between the field of computational thinking evaluation and other research fields, and to provide valuable research topics for future related research.

2. STUDY DESIGN
In this paper, a systematic literature review approach was used to conduct the review following the routine procedures suggested by Jesson et al. "Planning" "Literature search" "Literature assessment" "data extraction," "data integration," and "writing the review" to conduct the review. Specifically, this paper utilizes a systematic literature review approach to conduct the study.

A systematic literature review is a necessary tool for accurately and reliably summarizing research evidence. This study adopts the method of systematic literature review to sort and analyze the relevant research results of computational thinking evaluation in China from 2019 to 2024, and the main processes are as follows: (1) according to the research objectives, clarify the keywords of literature search; (2) based on the keywords, launch literature search in the database to obtain the literature; (3) based on the research questions, set the criteria for the inclusion of the literature, carry out the assessment of the validity of the literature, and carry out literature primary screening; (4) read the full text of the literature after primary screening, and on this basis, carry out the secondary screening of the literature. (3) Based on the research question, set the literature inclusion criteria, carry out the assessment of the validity of the literature, carry out the initial screening of the literature; (4) Read the full text of the literature after the initial screening, and on the basis of this, carry out the secondary screening of the literature, and carry out a comprehensive analysis of the final sample of the literature.

2.1. Formulation of research questions Sub-sections Guidelines

Referring to the experience of systematic literature review in academia, this study will analyze the results from descriptive statistics used to answer the questions "What is the current status of computational thinking evaluation research?" and "What are the future research priorities for computational thinking evaluation research?" Two research questions.

2.2. Formulation of research questions Sub-sections Guidelines

This study draws on the method of literature acquisition in systematic reviews, using the CNKI database as the main data source, with the search time limited to 2019-2024 (as of February 1, 2024), and the language of literature in Chinese. The main search method was based on the subject word ("computational thinking" AND "evaluation"), the subject word ("computational thinking" AND "educational evaluation ") and subject line ("computational thinking" AND "educational evaluation") as the search formula, a total of 2,065 literatures were obtained.

2.3. Literature screening and quality assessment Sub-sections Guidelines

This study followed the Preferred Reporting Items for Systematic Reviews and Meta Analyses (PRISMA) methodology to screen the relevant literature on computational thinking evaluation research through four stages: search, screening, qualification and inclusion, and the specific PRISMA screening process is shown in Figure 1. Among them, literature inclusion and exclusion criteria were formulated according to the research questions, and the criteria for literature exclusion were: (1) non-Chinese papers; (2) non-journal papers; (3) non-CSSCI and Beida core journals; (4) full-text unavailability; (5) the studies did not have a clear research question, methodology, and conclusions; and (6) the topic did not focus on the evaluation of computational thinking. Finally, the valid research literature was determined to be 27 articles. Combining the 27 studies obtained from the screening, this study analyzed them sequentially in terms of the dimensions of literature publication, literature citation, and research subjects.

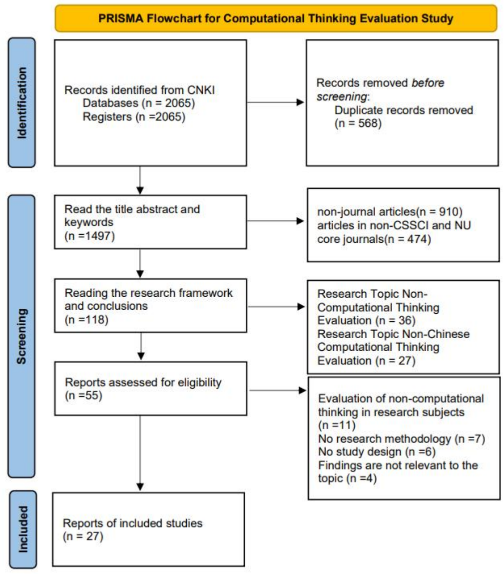


Figure 1. PRISMA Flowchart for computational thinking evaluation study.

3. STATUS OF RESEARCH ON THE EVALUATION OF COMPUTATIONAL THINKING

3.1. Literature published

The distribution of the sample literature by year provides some insight into the general development of the research. As shown in Figure 2, among the sample literature, in terms of publication time, the literature shows that the research on computational thinking evaluation in China published the largest number of papers in 2022 (n=8), followed by 2020 (n=6), indicating that this is related to the relevant national policies put forward.In October 2020, the Central Committee of the Communist Party of China (CPC) and the State Council issued the Overall Program for Deepening the Reform of Educational Evaluation in the New Era (hereinafter referred to as the Overall Program), which made an overall deployment and requirements for education evaluation and opened the curtain of education evaluation reform in China, so the research in the field of computational thinking evaluation in 2020 showed a trend of rapid growth in general. Since then until 2022, the Ministry of Education released the Compulsory Education Information Technology Curriculum Standards (2022 Edition), which explicitly pointed out computational thinking as one of the four core literacies in the subject of information technology. In general, the field of computational thinking assessment has been hot in recent years and is expected to remain a key research direction in the field of education in the future.

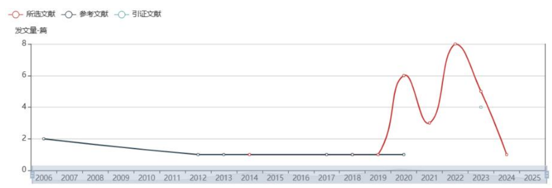


Figure 2. Distribution of Computational Thinking Evaluation Publications in China.

3.2. *Literature citations*

The citation rate of a paper can reflect the academic influence of the paper to a certain extent. Among the 27 sample literatures, the topics of the published highly cited papers are mainly related to the cultivation path, connotation, current situation, evaluation tools, and evaluation system of computational thinking evaluation. The citations of some of the highly cited papers, as shown in Table 1, show that the academic focus has gradually shifted from the connotation at the early stage of the study to the cultivation path and evaluation tools from the year of publication of the highly cited papers.

Table 1. Citation of highly cited papers (partial).

Citation Rate	Article Title	Subject	Year of publication	Journal
229	Construction and application of a computational thinking assessment tool for students at K12 level	Evaluation tools	2019	China Electronic Education (CEE)
125	How should computational thinking be evaluated? -- Based on a comparative analysis of 14 domestic and international assessment tools	Evaluation tools	2020	Journal of Distance Education
99	Construction and Exploration of Localized Computational Thinking Evaluation Indicator System - Sample Analysis and Validation Based on 1410 High School Students	rating system	2020	Journal of Distance Education
69	How far is the road to computational thinking development? --A Computational Thinking Assessment Perspective	Cultivation Pathways, Con -notation, Current Status	2020	Open Education Research
34	Content Construction and Methodological Design of Academic Assessment of Computational Thinking - A Comparative Literature Study Perspective	Cultivation Pathways, Connotation, Current Status	2022	China Distance Education
26	A new approach to evaluating computational thinking: micro-credentialing	Cultivation path	2022	Open Education Research
19	Design of a Computational Thinking Descriptive Framework to Support the Construction of an Evaluation System for China's Information Technology Curriculum	rating system	2022	Research on Electrochemical Education

3.3. *Analysis of Research Subjects*

The research subjects of the 27 sample literatures are characterized by plurality, as can be seen in Figure 3, the students who are the subjects of the research cover secondary education, primary education, and higher education students. The research disciplines include computer education, science education, educational theory and education management, and STEM education. Computational thinking scale mainly accomplishes the measurement of computational thinking through students' subjective evaluation of self-perception, and many teams in China have already conducted research related to scale ontologization, compiling computational thinking scales for higher education teacher training students, high school students, K12 students, and primary school students.

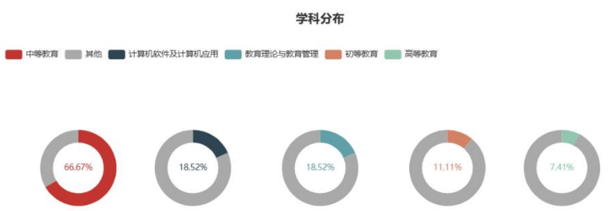


Figure 3. Distribution of research subjects of computational thinking evaluation in China.

4. RESEARCH FOCUS ON COMPUTATIONAL THINKING EVALUATION

In education and teaching, the cultivation of computational thinking is very important, and the change of students' thinking is diverse and dynamic, which requires teachers to be able to pay attention to the dynamic process of students' computational thinking development in real time in the process of cultivation. Evaluation can determine the current level of students' computational thinking, which is of great significance for teachers to target the cultivation of students' computational thinking (Liu Jiao and Li Jiansheng, 2019). By studying and analyzing the content of the sample literature, the research themes of computational thinking evaluation in China can be summarized into three categories: evaluation method research, evaluation tool research, and evaluation index system research, which are analyzed as follows:

4.1. *A Study of Computational Thinking Evaluation Tools*

As shown in Table 1, Yu et al. have sorted out and summarized the currently available evaluation methods (Yu, Xiao Min and Wang Meiling, 2018). Different evaluation methods are suitable for different grades, for example, for students in the lower grades, using text-discourse analysis may not be able to fully grasp the students' thinking process, so it is more suitable for behavioral analysis. For students in higher grades, they have already mastered certain basic subject knowledge, and the use of topic tests, work analysis and graphical analysis can all achieve the goal.

Table 2. Computational Thinking Evaluation Approach Form.

Evaluation Methods	Support Tools	Typical Cases
textual discourse	Analyzing Interviews, Audible Thoughts	Brennan and Resnick used work-based interviews;
Title Test	Test Questions	Román-González et al. proposed CT test questions; Aggrwal et al. proposed pre- and post-test test based on kodu program; Chen proposed aptitude transfer test questions;
Artwork Analysis	Scrap、CTP	Brennan and Resnick use Scrape to analyze information about the modules used in Scratch projects; Koh presents CTP charts to detect the development of computational thinking skills;
graphical analysis	Flowcharts, pseudo-code	Chen asked students to form code sequences for problem solving;
behavioral analysis	Field observation, screen recording	The Esteves et al. used an observational method to record important information during conversations.

4.2. *A Study of Evaluation Tools for Computational Thinking*

Huigongjian (2020) and others screened out 14 typical domestic and international assessment tools for computational thinking, among which those applicable to secondary school students and primary school students are shown in Table 2. From the table, it can be seen that the evaluation tools of various countries focus on three levels: conceptual level, skill level and ability level, the conceptual level is mainly to understand the concept of computational thinking; the skill level: reasoning, abstraction, decomposition, generalization, programming and so on; and the ability level: problem solving ability, transferability, creativity, cooperation ability, practical ability and so on. From the view of foreign evaluation tools, they tend to reflect the development level of computational thinking through the development of students' ability in the process of problem solving, which is of great significance for this study, especially the decomposition of thinking ability involved in the process of problem solving in Turkey and the U.S.A. is more inspiring for this study, and the programming ability in Japan and the algorithmic ability in the U.S.A., which are closely combined with the information technology curriculum, are of great significance for this study as well.

Due to the specificity of the research object, the dynamics of the learning process and the complexity of computational thinking, it is impossible for foreign evaluation tools to be fully applicable to Chinese students. The development of various technologies has facilitated the design and development of tools, platforms, and systems related to teaching and learning evaluation, thus helping to carry out teaching and learning evaluation in offline, online, and online-offline hybrid teaching. According to Qin Yuyou, the construction of evaluation standards must take into account their practicality in order to be implemented on the ground, and the key lies in "exploring a systematic and effective evaluation index system" (Qin Yuyou, 2021). Therefore, combining the psychological characteristics of our students with the evaluation criteria of computational thinking is more in line with the actual situation in China.

Table 3. Computational Thinking Evaluation Tool Sheet.

Target audience	Name of tool	Developer/user	Country	Evaluation content	Type of evaluation
primary and secondary school students	CTt	Román-González	Spanish	Computational Thinking Concepts, Problem Solving	diagnostic evaluation
	Bebras Tasks	Computational Thinking challenge	Lithuania	Ability to transfer computational thinking skills	Summative evaluation
	Dr Scratch	Moreno-León	Spanish	Computational Thinking Concepts, Programming skills	Formative evaluation
	CTSiM	Basu	US	Computational Thinking Concepts and Practices	Formative and Summative Evaluation
middle-school student	CTSES	Kukul	Turkish	Reasoning, abstraction, Decomposition, generalisation	diagnostic evaluation
	The Fairy Assessment	Werner	US	Algorithmic thinking, abstraction, Problem Solving	Summative evaluation
	FACT's systems of assessments	Grover	US	Computational Thinking Concepts, Problem solving, creativity, co-operation, transferability, etc.	Formative and Summative Evaluation
	NCV	Ota	Japan	Computational thinking, programming skills	Formative evaluation

4.3. Research on Computational Thinking Evaluation Index System

The evaluation indicator system is an important part of the evaluation implementation, and the establishment of the evaluation indicator system is based on the overall guiding direction of the evaluation objectives, and the decomposition and refinement of the evaluation objectives, in order to establish operable and specific indicators. Therefore, the indicator system can comprehensively reflect the characteristics of a certain thing, phenomenon or process, and the indicators are interconnected with each other to form a unified structural system (Zhou Guorong, 1993).

The evaluation index system of computational thinking is a series of indicators reflecting the characteristics of computational thinking, which cannot be separated from the basic guiding role of the evaluation framework, and one of the ideas of constructing the evaluation index system of computational thinking is to decompose the evaluation framework of computational thinking layer by layer. The construction of evaluation index system from the perspective of evaluation framework's guidance to the index system is mainly divided into three major categories and some minor categories, as shown in Table 4, one category is based on the direction of computational thinking three-dimensional framework, one category is based on the direction of the ISTE five-dimensional framework, and one category is based on the computational thinking evaluation indexes or index system constructed according to Selby's and Woollard's five elements combined with other frameworks.

Table 4. Computational Thinking Evaluation Indicators and Indicator System.

Grounded Theory	Proponent	Indicator or system of indicators
A three-dimensional framework for computational thinking	Brennan 和 Resnick	Computing concepts, practices and perceptions are primary indicators, and sequential, cyclic, parallel, event, conditional, test and iteration are secondary indicators.
	Yu Xiaohua et al.	Tier 1 indicators at cognitive/operational and non-cognitive levels, with problem identification and problem decomposition, abstract modelling skills, algorithmic design skills, automation skills, problem migration skills, articulation, connectivity and questioning as tier 2 indicators
ISTE Five Dimensional Skills	Bai Xuemei and Gu Xiaoqing	Creativity, algorithmic thinking, critical thinking, problem solving, and collaboration skills are Level 1 indicators.
Selby and Woollard's Five Elements	Chen Xingye	Attitude towards computational thinking and computational thinking skills are the primary indicators, and emotional attitude, quality of thinking, co-operative learning, decomposition, abstraction, algorithm, assessment and generalisation are the secondary indicators, comprising an evaluation system of 30 key indicators.
American Association of Computer Science Teachers	American Association of Computer Science Teachers	9 skills in data collection, data analysis, data presentation, problem decomposition, abstraction, algorithms and procedures, automation, simulation and parallelism.
Stanford, USA College of Education	Zhou Chun	The 5 sub-competencies of computational thinking and assessment objectives.

Most of the scholars in China form a localised evaluation framework based on the introduction of foreign evaluation scales and evaluation index systems. Yu et al. proposed a competency-oriented micro-certification to evaluate and certify the components of computational thinking, decomposing computational thinking into six sub-competencies, namely, problem identification and decomposition, abstract modelling, algorithm design, automation, problem transfer ability, and computational concepts from the cognitive and operational levels as well as from the non-cognitive level; discussing the developmental level of the sub-competencies in the K-12 stage and the appropriate assessment methods; demonstrating the implementation process of computational thinking micro-certification, and exploring the differences between the implementation in formal and informal learning contexts (Yu et al. the implementation process of computational thinking micro-certification, and explore the differences in implementation in formal and informal learning contexts (Yu, Xiaohua, Wang, Meiling, Cheng, Jiamin, & Qiu, Zhenhua, 2022).

Based on the analysis of ISTE's theoretical framework of computational thinking, Korkmaz et al. developed a scale with a six-factor structure for measuring students' computational thinking by combining the relevant research results of previous researchers. Bai Xuemei and Gu Xiaoqing used the CT scale with 22 measurement indicators designed and developed by Korkmaz et al. as a research tool, and combined it with the real educational context in China, localised the scale, and finally formed a scale suitable for students at K12 level (Bai Xuemei and Gu Xiaoqing, 2019). Finally, a computational thinking framework suitable for Chinese K12 students was formed (Bai Xuemei and Gu Xiaoqing, 2019). Chen Xingzhi et al. developed an indicator system containing two primary indicators, eight secondary indicators and 30 key indicators, used the Delphi method to score the indicators at all levels in the indicator system, collected the scores and comprehensively analysed them before revising the indicator system, and then formed the computational thinking evaluation indicator system after three rounds of iteration (Chen Xingye and Ma Yingying, 2020).

5. REFLECTIONS AND INSIGHTS CONCLUSION

Evaluation of computational thinking is both an important link in the cultivation of computational thinking and an important basis for verifying the effectiveness of cultivation. Combined with the current research status and research focus, this study mainly puts forward corresponding thoughts and insights from three aspects: evaluation objectives, evaluation contents and evaluation methods.

5.1. Evaluation Objective: Literacy development oriented, focusing on human development and tailored to the needs of the students.

In the era of digital intelligence, the use of big data analysis technology to evaluate the comprehensive quality of students and assess their high-level cognitive and non-cognitive abilities is the focus of change in the field of educational evaluation. Zhu Jing et al. point out that when entering the new stage of smart education, the use of AI technology to take care of the endogenous and innovative needs of personalisation has become a historical necessity again (Zhu Jing and Cai Jiandong, 2020). Based on the personalised and precise learning characteristics of smart education, Zhang Yaoyuan et al. point out that the purpose of assessment should be at least as detailed as the ability to 'identify' learners' individual strengths and talents, so as to 'tailor teaching to individual students' (Zhang Yaoyuan and Gong Xianghe, 2023). At the same time, intelligent technology-enabled education evaluation is conducive to the concept of "teaching according to ability". Therefore, the mutual empowerment of AI and education, as well as the deep expectation of innovative talents in the new era, has pushed the evaluation concept from focusing on "selection" to "teaching according to ability", and put human development in the focus of evaluation. "It puts human development at the centre of education evaluation reform. In the future, we can continue to deepen our theoretical understanding, combine with the actual situation of students and schools, and actively explore the practical research on the evaluation of computational thinking education.

5.2. Evaluation content: a systematic framework of evaluation content oriented towards computational thinking

The current study, which constructs a framework of evaluation content from three dimensions of computational concepts, computational practices and computational ideas, provides a consistent reference system for the evaluation of computational thinking. There is no systematic assessment of data collection, data analysis, system and problem identification. System is to assess students' overall knowledge of system architecture after integrating basic computing concepts, and it is the basis for developing students' problem solving and hands-on design and development in computing practice.

Currently, research focuses on the assessment of computational thinking at the cognitive level, such as computational concepts and computational practice, while less research has been conducted on non-cognitive assessment, such as computational concepts. Although computational thinking is primarily a cognitive-psychological construct of problem-solving ability, it also has complementary non-cognitive factors. The main reason for the above problems is the incomplete understanding of the concept of computational thinking, the omission of the cultivation of some important dimensions and the neglect of the linkage within the cultivation indicators, which only focuses on partial information and cannot be systematised. Therefore, the follow-up research should first focus on the comprehensive interpretation of the theoretical framework, on the basis of which it should supplement the cultivation and assessment of the blank content indicators to achieve the all-round development of students' computational thinking and to promote the systematisation of the computational thinking evaluation content system.

5.3. Evaluation: Learning outcome oriented, reconstructing a multi-dimensional evaluation index system

The evaluation of computational thinking should be constructed in terms of both principle understanding and project practice (Ren Youqun et al., 2016), reflecting both an individual's mastery of the core concepts and methods of the computing discipline and the ability to apply them to solving generalised problems. The Overall Programme anchors the fundamental purposes of accelerating the modernisation of education, building a strong education country and providing education to the satisfaction of the people, and in the context of the reform of teaching evaluation, the content of teaching evaluation has shifted to a comprehensive evaluation that places equal emphasis on knowledge, competence and literacy, and the subject of evaluation has also shifted from a single-subject evaluation to the diversification of the evaluation subject, and the mode of evaluation has shifted from a summative evaluation to a focus on formative evaluation, and the means of evaluation have also been progressively Evaluation methods are also gradually enriched from traditional questions and test papers to the use of various new-generation information technology to carry out evaluation. Bloom divided learning evaluation into three categories: summative evaluation, diagnostic evaluation and formative evaluation, and the research of Bloom and his team showed that: continuous formative testing, feedback and correction in the learning process is an effective strategy to promote students' knowledge mastery (B.S. Bloom, & JF Maddox, 1987).

Aiming at the complex composition of computational thinking, theoretical research proposes a variety of evaluation methods, and points out that it is necessary to comprehensively understand the development of students' computational thinking in the form of multiple combinations of these methods. At present, China's research on the evaluation of computational thinking is mostly biased towards macro-theoretical exploration, and the reform of educational evaluation promotes scholars to reflect on the status quo of teaching evaluation. In traditional teaching evaluation, there are problems such as lopsided content of teaching evaluation, a single subject of evaluation, and overly uniform standards. The follow-up research should: (1) optimise the evaluation methods, refine the implementation steps of each method and improve the feasibility of operation; (2) combine with the training content, combine appropriate evaluation methods, and practice the multiple evaluation model; (3) pay attention to the process and the results at the same time, let the evaluation go through the whole process of teaching and reduce the burden of the students while making the evaluation results more comprehensive and reliable.

6. CONCLUSION

This paper adopts a systematic literature review method, follows the PRISMA procedure to screen 27 domestic computational thinking evaluation related literatures, uses content analysis to integrate the data, and by summarising the current status of research in the field of computational thinking evaluation in China, the focus of research. And on this basis, three reflections and insights are drawn. In general, the current domestic computational thinking evaluation field has stepped into a stable development period, accumulated richer research results, and initially formed its own research characteristics, but at the same time, there are still some problems, with great development potential. This paper calls for more researchers to join the field and continue to promote the paradigm shift of the field in the future, including expanding to a wider range of research contexts and more balanced research questions at the ontological and epistemological levels, expanding to a more diverse range of research methods at the methodological level, and expanding to a more humanistic perspective at the value level, so as to better promote the growth of the field. The limitation of this paper is that it fails to compare the current development of the field of computational thinking assessment in foreign countries. In the future, we willfurther conduct a systematic review of the newer achievements in foreign countries based on the developed coding catalogue, in order to contrast and synthesise with the conclusions of this paper.

7. REFERENCES

Yu, Xiaohua and Wang, Meiling (2020). *How far is the road to computational thinking development? --Based on Computational Thinking Measurement Perspective.* Open Education Research, 26(01):60-71.

MOHER D, LIBERAT A, TETZLAFF , et al (2009). *Preferred reporting items for systematic reviews and meta-analyses-the PRISMA statement.* Jlournal of clinical epidemiology, 62(10):1006-1012.

Liu, Jiao and Li, Jiansheng (2019). *Assessment methods and application case studies of computational thinking.* Modern Educational Technology (10), 94-99.

Yu, Xiaohua, Xiao Min and Wang Meiling (2018). *Computational thinking cultivation in progress:Practical methods and evaluation in K-12.* Journal of Distance Education (02), 18-28.

Tian, Aili (2020). *Comprehensive quality assessment: the change and implementation of learning assessment in the age of intelligence.* China Electrochemical Education (01), 109-113+121.

Qin, Yuyou (2021). *How to promote education evaluation with joint efforts.* Chinese electrochemical education (7), 6-8.

Zhou, Guorong (1993). *Dictionary of applied statistical methods.* Beijing:China Statistical Publishing.

Yu, X., Wang, M., Cheng, J. M. and Qiu, Z. (2022). *A new way to evaluate computational thinking: micro-certification.* Open Education Research,28(01):107-120.

Bai, Xuemei and Gu, Xiaoqing (2019).*Construction and application of computational thinking evaluation tools for students in K12.* China Electrochemical Education(10),83-90.

Chen, Xingye and Ma, Yingying (2020). *Construction and exploration of localised computational thinking evaluation index system - sample analysis and validation based on 1410 high school students.* Journal of Distance Education (05), 70-80.

Zhou, Chun (2023). *Promoting the development of junior high school students' computational thinking with unit expressive assessment.* China Information Technology Education, (01):34-38.

Ren, Youqun, Sui, Fengwei and Li, Feng (2016). *What makes digital natives possible? --Also on the necessity and possibility of computational thinking into primary and secondary IT education.* China Electrochemical Education,(1):2-8

B. S. Blum, & JF Maddox (1987). *Educational evaluation.* Shanghai: East China Normal University Press, 228-267.

Which Scaffolding is More Effective? A Study of the Effect of Scaffolding on the Learning Achievement and Computational Thinking of Students with Different Levels of Metacognition

Yu-Die Zhu[1], Wei Li[2*], Cheng-Ye Liu[3], Ji-Yi Huang[4], Wen Li[5], Jian-Kang Rong[6]
[1,2,5,6]Wenzhou University, China
[3]The Chinese University of Hong Kong, Hong Kong
[4]Quxi NO.1 Primary School in Ouhai District Wenzhou, China
1584085180@qq.com, liweiwzu@wzu.edu.cn, chengyeliu@cuhk.edu.hk ,
hxiaoji_111@163.com, 1527666769@qq.com, 2502646889@qq.com

ABSTRACT

The purpose of this study was to investigate the effects of scaffolding types and metacognitive level on students' learning achievement and computational thinking. Eighty-three fourth-grade students in an elementary school in Wenzhou City participated in a six-week experiment and were assigned to Experimental 1 (n = 41) and Experimental 2 (n = 42) classes. Students in Experiment 1 class were intervened with the reflective scaffolding, while students in Experiment 2 class were intervened with the supportive scaffolding. The results of the experiment showed that the type of scaffolding interacted with the level of metacognition in terms of computational thinking and metacognitive tendencies. For students with low metacognitive levels, supportive scaffolding were more beneficial than reflective scaffolding in enhancing their computational thinking tendencies and metacognitive tendencies. However, the interaction between type of scaffolding and individual metacognitive level did not have a significant effect on learning achievement.

KEYWORDS

computational thinking, scaffolding, metacognition

1. INTRODUCTION

With the advent of the digital age, computational thinking is considered an important thinking skill for 21st century adolescents (Bocconi & K., 2022). Computational thinking refers to a set of thinking activities that utilize the fundamental concepts of computer science to solve problems, design systems, and understand human behavior (Wing, 2006). Computational thinking emphasizes the gradual automation of complex tasks by abstracting and decomposing them. However, when actually performing a task, students often struggle to break down large and complex problems into smaller, more manageable problems, a difficulty that affects their ability to successfully solve problems (Perry et al, 2019). Research has shown that without appropriate instruction, students face significant challenges in developing computational thinking skills (Denner et al., 2012). Therefore, researchers believe that learners can be given scaffolding to help them break down difficult tasks (Zhou et al., 2023).

Research has proved that different types of scaffolding have different effects on students (Atman Uslu et al., 2022), and more recent studies have compared the advantages of different types of scaffolding (Kim et al., 2021). Supportive scaffolding can guide learners on what to consider and how to connect ideas, and reflective scaffolding help learners clarify their reflective process through metacognitive questions. (Kim & Lim, 2019). Many studies have reported that scaffolding can improve learners' problem-solving skills and positively affect academic achievement (Joo-Yeun, 2015). However, they did not demonstrate which type of scaffolding was most effective for different students.,

In addition, research has shown that students' problem-solving abilities vary due to their different levels of metacognition, which can affect learning outcomes (Huang & Zheng, 2021). Students with high levels of metacognition will have higher problem-solving abilities, while students with low levels of metacognition may need more help in problem-solving activities. Therefore, different scaffolding should be provided for students with different metacognitive levels (Kim & Lim, 2019). In summary, this study aimed to investigate the differences in the effects of the types of scaffolding on students with different levels of metacognition. The research questions are as follows:

(1) Does the type of scaffolding affect the academic performance of students with different metacognitive levels?

(2) Does the type of scaffolding affect the computational thinking tendency of students with different metacognitive levels?

(3) Do the types of scaffolding affect the metacognitive tendencies of students with different metacognitive levels?

2. METHODOLOGY

2.1. Participants

Eighty-three fourth-grade students in an elementary school in Wenzhou City participated in a six-week experiment. Their average age was 11-12 years old and they were assigned to two classes. One class was Experiment 1 class (n = 41) and the other was Experiment 2 class (n = 42). Student information was anonymized.

Within the experimental classes, the experimental subjects were divided into two groups each based on high and low levels of metacognitive awareness by collecting data from the experimental subjects' metacognitive awareness tendency questionnaire. Both classes were taught by the same teacher and had the same study time and course content.

2.2. *Research Context*

In this experiment, four lessons from the fourth-grade information technology textbook of Zhejiang Province's compulsory education textbook were selected as the main learning content. During the experiment, the supportive scaffolding was presented in the form of hints and visual materials to provide classroom knowledge content and instructions that learners needed to receive. The reflective scaffolding consisted of exploratory questions, hints, and summaries that provided learners with metacognitive questions to help them review the content and reflect on the learning process. The instructor provided no additional support to control the experiment during the four-week learning process. Each time the design of the scaffolding was completed, it was submitted to a pilot school teacher with four years of teaching experience for review and revision.

2.3. *Measurement Instruments*

The measurement tools used in this study include the pre-test and post-test of learning achievement, the Computational Thinking Tendency Questionnaire and the Metacognitive Tendency Questionnaire. The pre-test and post-test of learning achievement were developed by the authors in conjunction with what the students had learned, and the papers were reviewed and approved by experts in the field and experienced information technology teachers, with a level of difficulty appropriate for fourth-grade students. Both the pre-test and post-test contain 10 multiple-choice questions. Each set of questions contained two reverse thinking questions of ten points each, totaling 100 points. The pre-test and post-test questions have roughly the same compositional structure and are of similar difficulty.

The instruments of the questionnaires of computational thinking was adapted from a scale developed by Hwang and Li (2020). It consists of 6 questions using a 5-point Likert scale scoring scheme (i.e., strongly agree, agree, generally, disagree, and strongly disagree). The Cronbach's alpha of the questionnaire was 0.763.

The Metacognitive Dispositions Questionnaire was adapted from the scale developed by Lai and Hwang (2014) and contained 5 entries with a Cronbach's alpha value of 0.757.The questionnaire was scored on a 5-point Likert scale with scores ranging from 1 to 5, representing in descending order, "Completely Disagree", "Disagree", "Neutral", "Agree" and "Completely agree".

2.4. *Experimental Procedure*

The experiment lasted for six weeks, with one class period per week and 45 minutes per session. In the first week, students need to complete the pre-test and pre-questionnaire. During the second through fifth weeks, the teachers taught the students, with the intervention using reflective scaffolding in Experiment 1 class and supportive scaffolding in Experiment 2 class. During the sixth week, all the students completed the post-test and the post-questionnaire. (shown in Figure 1).

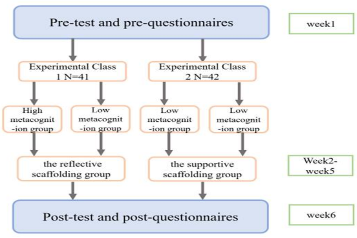


Figure 1. Experimental Procedure

2.5. *Data Collection and Analysis*

In this experimental study, the following types of data were collected: pre-test and post-test learning achievement, pre-test and post-test computational thinking tendency, and pre-test and post-test metacognitive awareness tendency.

In order to differentiate between students with different levels of metacognition, students were categorized into high and low metacognitive groups based on their metacognitive awareness tendency scores on the pre-test questionnaire. Experimental class 1 scored above the median (3.8) as high metacognitive level(HML) and below as low metacognitive level(LML); experimental class 2 scored above the median (4.0) as high metacognitive level and below as low metacognitive level.

After removing the data from the anomalies, a sample size of 16 for the high metacognition level group and 14 for the low metacognition group in Experimental 1 was collected, for a total sample size of 30 for Group 1; and a sample size of 15 for the high metacognition level group and 15 for the low metacognition group in Experimental 2 was collected, for a total sample size of 30 for Group 2. The total sample size of the two groups was 60. The experimental subjects are shown in Table 1.

Table 1. Subject status questionnaire

Group	Metacognition grouping	N
Class1	Low	14
	High	16
	Total	30
Class2	Low	15
	High	15
	Total	30

3. RESULTS

3.1. *Learning Achievement*

In order to understand the effect of the type of scaffolding and the level of metacognition on students' learning achievement, a two-factor covariate approach to data analysis was used. Students' pre-test scores were used as covariates to eliminate differences in learners' prior knowledge levels before the learning activity. The independent variables were the type of scaffolding and the level of metacognition, and the dependent variable was the students' learning achievement.

The results of the two-factor analysis of covariance are shown in Table 2, which shows that there is no significant interaction between type of scaffolding and metacognitive level in terms of learning achievement ($F=.211$, $p>0.05$), indicating that type of scaffolding and metacognitive level do not jointly have a significant effect on students' learning achievement. In addition, scaffolding type ($F=1.261$, $p>0.05$) and metacognitive level

(F=0.061, p>0.05) did not have a significant effect on students' learning achievement.

Table 2. Results of two-factor covariance analysis of students' learning achievement

Variable	SS	df	MS	*F*	*P*	η^2
Pre-test	1023.897	1	1023.897	2.911	.094	.050
Metacognition grouping	21.311	1	21.311	.061	.806	.001
Class	443.659	1	443.659	1.261	.266	.022
Metacognition grouping * Class	314.317	1	314.317	.894	.349	.016
Error	19345.627	55	351.739			

3.2. Computational Thinking Tendency

In order to understand the effect of scaffolding type and metacognitive level on the development of students' propensity to think computationally, a two-factor covariance analysis was used in this study. The two-factor covariance results were analyzed as shown in Table 3, and there was a significant interaction between scaffolding type and metacognitive level in terms of computational thinking tendency (F=5.973, p<0.05), with an interaction effect size of 0.098 between the two.

Table3. Results of a two-factor covariance analysis of students' computational thinking tendency

variable	SS	df	MS	*F*	*P*	η^2
Class	3.179	1	3.179	14.720**	.000	.211
Metacognition grouping	.012	1	.012	.056	.814	.001
Class* Metacognition grouping	1.290	1	1.290	5.973*	.018	.098
Error	11.878	55	.216			

The simple main effect of metacognitive level on students' propensity to think computationally was further analyzed, as shown in Table 4. The results showed that there was no significant difference between the metacognitive levels of the students in both Experimental 1 (F=3.658, p>0.05) and Experimental 2 (F=2.783, p>0.05) in terms of their propensity for computational thinking.

Table4 .Simple main effect analysis of metacognitive level on students' computational thinking tendency

variable		SS	df	MS	F	P	η²
Class1	Between groups	.819	1	.819	3.658	.064	.061
	Within groups	6.042	27	.224			
	Total	437.417	30				
Class2	Between groups	.588	1	.588	2.783	.129	.041
	Within groups	5.709	27	.211			
	Total	561.083	30				

The results of the simple main effect analysis of the type of scaffolding on students' propensity for computational thinking are shown in Table 5. There was a significant difference in the development of computational thinking tendency among students with low metacognitive levels using different scaffoldings in the learning process (F=18.527, p<0.001, η²=0.259). There was no significant difference in the development of computational thinking tendency for students with high metacognitive level using different scaffoldings during learning (F=0.744, p>0.05). The results suggest that scaffoldings are more favorable to students with low metacognitive levels compared to students with high metacognitive levels.

Table5. Simple main effect analysis of scaffolding type on students' computational thinking tendency

Variable		SS	df	MS	*F*	*P*	η^2
LML	Between groups	4.247	1	4.247	18.527***	.000	.259
	Within groups	5.960	26	.229			
	Total	474.083	29				
HML	Between groups	.151	1	.151	.744	.314	.018
	Within groups	5.687	28	.203			
	Total	524.417	31				

****p < 0.001*

Figure 2 shows the interaction plot between the type of scaffolding and the effect of metacognitive level on students' propensity to think computationally. The results show that students using supportive scaffolding have a higher propensity for computational thinking than students using reflective scaffolding. In addition, when using supportive scaffolding, students with low metacognitive levels developed significantly higher computational thinking tendencies than students with high metacognitive levels.

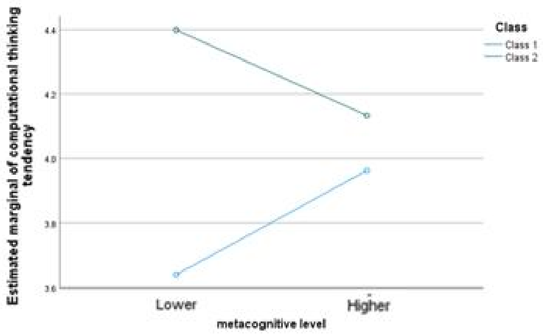


Figure 2. The Interaction of Scaffolding and Metacognitive Levels in the computational thinking tendency

3.3. Metacognition Tendency

In order to understand the effect of type of scaffolding and level of metacognition on the development of students' metacognitive tendencies, a two-factor covariance analysis was used in this study. The two-factor covariance results were analyzed as shown in Table 6, the type of scaffolding had a significant effect on students' metacognitive tendencies ($F=6.782$, $p<0.05$), with an effect size of 0.11. At the same time, the type of scaffolding and the level of metacognition had a significant interaction effect on the students' metacognitive tendencies ($F=8.545$, $p<0.01$), with an interaction effect size of 0.134 between the two.

Table6 .Results of two-factor covariance analysis of students' metacognition tendency

variable	SS	df	MS	F	P	η²
Class	2.020	1	2.020	6.782**	.012	.110
Metacognitive grouping	.335	1	.335	1.125	.293	.020
Class* Metacognitive grouping	2.545	1	2.545	8.545**	.005	.134
Error	16.379	55	.298			

***p < 0.01*

The simple main effect of metacognitive level on students' metacognitive tendencies was further analyzed, as shown in Table 7. There was a significant difference in metacognitive tendencies among students with different metacognitive levels in Experimental 1 class ($F=5.778$, $p<0.05$, $\eta^2=0.108$). There was no significant difference in metacognitive tendencies among students with different metacognitive levels in Experimental Class 2 ($F=0.57$, $p>0.05$). The results indicate that the metacognitive tendencies of students with high metacognitive levels are significantly higher than those of students with low metacognitive levels when the reflective scaffolding is used.

Table7. Simple main effect analysis of metacognitive level on metacognition tendency

variable		SS	df	MS	F	P	η²
Class1	Between groups	1.624	1	1.624	5.778*	.013	.108
	Within groups	7.588	27	.281			
	Total	467.400	30				
Class2	Between groups	.185	1	.185	.570	.449	.010
	Within groups	8.770	27	.325			
	Total	550.680	30				

**p< 0.05*

As shown in Table 8, there was no significant difference in the metacognitive tendencies of students with high metacognitive levels when using different scaffoldings ($F=0.11$, $p>0.05$). However, there was a significant difference in metacognitive tendencies for students with low metacognitive levels when using different scaffoldings ($F=12.548$, $p<0.05$, $\eta^2=0.213$). The results suggest that using supportive scaffolding is more effective than using reflective scaffolding in improving metacognitive tendencies for students with low metacognitive levels.

Table8. Simple main effect analysis of scaffolding type on students' metacognition tendency

variable		SS	df	MS	F	P	η²
LML	Between groups	4.409	1	4.409	12.548**	.000	.213
	Within groups	9.135	26	.351			
	Total	435.200	29				
HML	Between groups	.003	1	.003	.011	.822	.001
	Within groups	7.067	28	.252			
	Total	582.880	31				

***p < 0.01*

Figure 3 shows the interaction between the type of scaffolding and the effect of metacognitive level on students' metacognitive tendencies. The results show that students who used supportive scaffolding had higher metacognitive tendencies than those who used reflective scaffolding. In addition, supportive scaffolding are more beneficial for students with low metacognitive levels to improve their metacognitive tendencies.

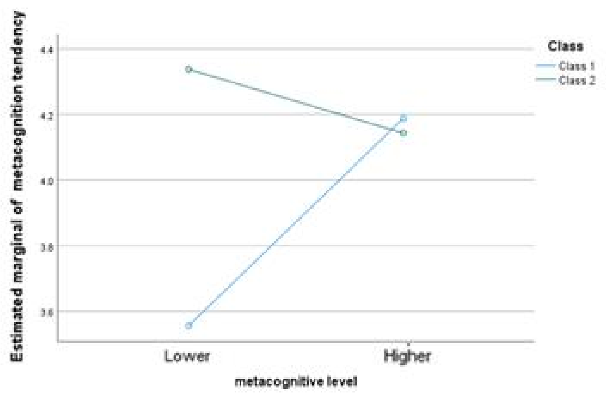


Figure 3. The Interaction of Scaffolding and Metacognitive Levels in the metacognition tendency

4. DISCUSSION AND CONCLUSIONS

This study examined the effects of scaffolding type on learning achievement, computational thinking tendency and metacognitive tendency of students with different metacognitive levels. The results showed that the type of scaffolding interacted with metacognitive level in terms of computational thinking tendency and metacognitive tendency. For students with low metacognitive levels, supportive scaffolding were more beneficial than reflective scaffolding in enhancing their computational thinking tendencies and metacognitive tendencies. However, the interaction between the type of scaffolding and individual metacognitive level did not have a significant effect on learning achievement.

In terms of learning achievement, according to the results of the study, learning achievement was not affected by the type of scaffolding, the level of metacognition, and the interaction between these two. This phenomenon may stem from the short duration of the experiment and the lack of instructional sessions, which limited the effect of scaffolding on students' learning interventions.

In terms of the development of computational thinking tendencies, the analyses revealed a significant interaction between scaffolding type and metacognitive level. Specifically, students who received supportive scaffolding demonstrated stronger computational thinking tendencies than those who used reflective scaffolding. Previous research has found that supportive scaffolding are more effective than reflective scaffolding in terms of spatial ability self-efficacy, which echoes the findings of the present study (Atman Uslu et al., 2022). Therefore, it can be inferred that supportive scaffolding play a more critical role in promoting the development of students' computational thinking tendencies.

In terms of metacognitive tendencies, the findings reveal that scaffolding type has a significant interaction with metacognitive level. This suggests that students' metacognitive thinking can be effectively promoted through the use of scaffolding. Therefore, scaffolding can be an effective strategy for students to promote metacognitive activities by providing explanatory questions that induce planning, monitoring, and assessment during the learning process to help students learn about specific domains (Lee, 2017). In addition, further comparative analyses found that supportive scaffolding promote students' metacognitive dispositions more than reflective scaffolding, especially in the group of students with low metacognitive levels.Jeon (2007) found in his study that scaffolding promote metacognitive thinking by enabling students to focus on important information, and that students in the low metacognitive group who performed better when receiving scaffolding about supportive types, which is consistent with the findings of this study.

In addition, there are some limitations of this study. First, the total sample size of this study is small, which may lead to weak representativeness of the experimental results, and it is recommended that the sample size can be enlarged in subsequent studies. Second, the experimental period of this experiment was short, and the intervention effect of the scaffolding was not obvious for students who were used to the traditional mode of teaching, making some aspects of the experimental results not significant. It is recommended that future researchers consider extending the experimental period to enhance the intervention effect of scaffolding on students' learning process.

5. REFERENCES

Atman Uslu, N., Yildiz Durak, H., & Ay, G. M. (2022). Comparing reflective and supportive scaffolding in 3D computer-aided design course: Engineering students' metacognitive strategies, spatial ability self-efficacy, and spatial anxiety. Computer Applications in Engineering Education, 30(5), 1454-1469.

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, , & K., E., J., Horvath, M.A., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V. and Stupurienė, G. (2022). Reviewing Computational Thinking in Compulsory Education.

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.

Huang, Y., & Zheng, X. (2021). Effects of Scaffolding Types and Individual Metacognition Levels on Learning Achievement in Online Collaborative Argumentation.

Hwang, G.-J., & Li, K.-C. (2020). Trends and strategies for conducting effective STEM reseGwo-Jen Hwang arch and applications: a mobile and ubiquitous learning perspective.

Jeon, H. J. (2007). The effects of learner's metacognition and scaffolding types on problem solving process in web-based PBL.

Joo-Yeun, K. L., Kyu Yon. (2015). The effects of scaffolding types on problem solving ability and achievement in problem solving learning with creative thinking method. Journal of Korean Association for Educational Information and Media.

Kim, J. Y., & Lim, K. Y. (2019). Promoting learning in online, ill-structured problem solving: The effects of scaffolding type and metacognition level. Computers & Education, 138, 116-129.

Kim, N. J., Vicentini, C. R., & Belland, B. R. (2021). Influence of Scaffolding on Information Literacy and Argumentation Skills in Virtual Field Trips and Problem-Based Learning for Scientific Problem Solving. International Journal of Science and Mathematics Education, 20(2), 215-236.

Lai, C.-L., & Hwang, G.-J. (2014). Effects of mobile learning time on students’ conception of collaboration, communication, complex problem-solving, meta-cognitive awareness and creativity.

Lee, S. (2017). Effects of Scaffolding and Metacognition and Collaborative Preference on Problem-solving in Blended Collaborative Learning. Korean Association for Educational Information and Media, 23(1), 83-108.

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

Zhou, Y., Chai, C. S., Li, X., Ma, C., Li, B., Yu, D., & Liang, J.-C. (2023). Application of Metacognitive Planning Scaffolding for the Cultivation of Computational Thinking. Journal of Educational Computing Research, 61(6), 1123-1142.

Assessing “Event” in Computational Thinking of Primary School Students: Design Principles and Test Validation

Shuhan ZHANG[1]
[1]Macao Polytechnic University
shuhan@mpu.edu.mo

ABSTRACT

Computational Thinking (CT) is a problem-solving skill that is essential to everyone. With the momentum of integrating CT into K–12 education, there is a growing trend of introducing CT at early educational stages, which necessitates more learning and assessment resources to support CT instruction. This study presents a validated assessment tool for measuring a CT construct, “event”, of primary school students. A principled approach, evidence-centered design, was adopted for test development, and six items were developed, covering different focal knowledge, skills, and abilities. To support students’ comprehension of the test, detailed function descriptions and an anchor item were provided. The test was then distributed to 639 primary school students (Grades 2–4) for validation. Item Response Theory was leveraged for psychometric analyses, and results revealed proper parameters, indicating the appropriateness of the test for the target group. Student performance was further analyzed, where a linear growth was identified along grade levels, and no gender difference was examined within each grade. Implications for CT primary education were discussed.

KEYWORDS

computational thinking, assessment, event, primary school, evidence-centered design~~, item-response theory~~

1. INTRODUCTION

The development of technology has brought computing skills into daily life (Bundy, 2007). Computational thinking (CT) has been regarded as one of the 21st-century skills that should be equipped by citizens (Voogt et al., 2015). The origin of CT can be traced back to 1980, when Seymour Papert introduced the idea of “algorithmic thinking”, depicting it as “the art of deliberately thinking like a computer, according, for example, to the stereotype of a computer program that proceeds in a step-by-step, literal, mechanical fashion” (Papert, 1980, p.27). The concept became popularized after Jeanette Wing reconceptualized it as “computational thinking” in 2006, noting that it is a problem-solving skill based on the basic concepts in computer science, which is essential for every individual, not just computer professionals (Wing, 2006; Wing, 2008).

CT is considered strongly associated with programming. Programming is a cognitive activity that requires understanding a problem as a computational task and developing corresponding solutions through writing a program (Pea & Kurland, 1984). Its association with CT depicts that CT underpins the cognitive process of solving problems whereas programming enlivens the process by generating the solutions (Lye & Koh, 2014). As a result, programming has become the main teaching tool for CT instruction (Grover & Pea, 2013), and CT and programming education has been incorporated into K–12 curriculum across the world (Bocconi et al., 2016).

With the growing momentum of CT education, students’ CT acquisition has gained intensive attention, which necessitates proper resources for teaching and learning practices. Assessment plays a central role in indicating students’ knowledge acquisition and providing feedback for further improvement. Thereby, there is a greater need for well-designed CT assessment tools appropriate for the target learners (Basu et al., 2021).

2. LITERATURE REVIEW

CT can be assessed with different approaches. Constructed response tests were most widely used in the literature (Tang et al., 2020). Based on well-defined items (e.g., multiple-choice questions), the approach allows large-scale distribution and time-efficient collection of quantitative data. Another measurement approach is performance-based assessment, which indicates students’ performance by analyzing their programming projects (e.g., codes) (McMillan, 2014). To scaffold the evaluation of the projects, a grading rubric is needed, where a checklist of the measured skills is provided (Tang et al., 2020). This approach is less applicable for novice learners, since students’ experience in using the programming platform is required (Chen et al., 2017).

Extensive attempts have been made in developing CT assessment tools targeting different age groups across K–12 education levels, encompassing secondary school students (e.g., Bubica & Boljat, 2021; Román-González, 2015; Yağcı, 2018), upper primary school students (e.g., Basu et al., 2021; Kong & Wang, 2021; Rowe et al., 2021), lower primary school students (e.g., de Ruiter & Bers, 2021; Relkin et al., 2020; Zhang & Wong, 2023), and preschoolers (e.g., Bers et al., 2014; Clarke-Midura et al., 2021). These tools cover a broad range of CT concepts (e.g., sequences, loops, conditionals) and CT practices (e.g., abstraction, debugging, algorithmic thinking), which could gauge students’ CT competencies in different dimensions.

Despite the growing literature on CT assessment, the line of research can be extended in three directions. First, as there has been an emerging trend in integrating CT into early childhood education (Bers, 2018), developing appropriate instruments for this age group is needed. Second, among the existing CT measurements, the construct “event” was hardly investigated. This may be due to the less application of the concept to daily problem-solving contexts. However, “event” is a critical component in CT conceptual frameworks, and a commonly used construct in programming platforms (Brennan & Resnick, 2012). Meanwhile, according to the K–12 Computer

Science Standards (CSTA, 2017), "event" is recognized as an essential concept that needs to be grasped by K–12 students. Therefore, recourses in assessing "event" would be valuable for CT research and teaching practices. Third, psychometric rigor of CT instruments should be improved (Grover & Pea, 2013). A tool with robust psychometric qualities should involve a systematic design model to clarify the rules and assumptions underlying the design, and a field test to generate the quantitative parameters for validation (Kane, 2010). Both steps, involving a principled design approach and a large-scale test distribution, are necessary to ensure the psychometric rigor of the design.

To address these gaps, this study developed an instrument for assessing "event" of primary school students. A principled approach was leveraged for test design, and a systematic development process was illustrated. To validate the test, a field test was administered, where psychometric qualities of the test were examined and student performance was analyzed. The study is guided by the following research questions:

RQ1: How can we design a test for assessing "event" of primary school students?

RQ2: What are the psychometric qualities of the test?

RQ3: How is students' performance on the test?

3. METHOD

3.1. Test Development

A principled approach, Evidence-Centered Design (ECD), was leveraged for the test design. ECD is a framework that connects student capacities with tasks through evidentiary arguments (Mislevy, 2007). The framework is composed of three models to guide the test design, namely, student model, evidence model, and task model (Mislevy et al., 2003). This section will elaborate on how the test was designed based on the three models.

3.1.1. Student Model

Student model defines what to measure (Mislevy et al., 2003). As the test aims to assess "event", the definition of the construct should be identified first. The definition proposed by Brennan & Resnick (2012) was referred to, noting that an event is an activity in which one thing triggers another thing to happen. Then, the scope of the construct was specified, where a set of focal knowledge, skills, and abilities (FKSAs) were proposed. According to the literature, multiple dimensions for evaluating programming performance can be identified, namely, the ability to identify the output of the given instructions, the ability to identify the instructions of the given output, and the ability to identify errors in the instructions to reach the given output (Zhang & Wong, 2023), which was incorporated into item development for the present study. The FKSAs for the test include: (1) Ability to identify the output of a set of instructions with simple events; (2) Ability to identify instructions with simple events to represent a given description; and (3) Ability to identify error(s) in a set of instructions that includes simple events.

3.1.2. Evidence Model

Evidence model aims to define how the constructs should be measured (Mislevy et al., 2003). Requirements for test development should be considered, in terms of the alignment with the measurement goal and the appropriateness for the target group. Table 1 displays the requirements drafted for the test, which covered test format, measurement goal (i.e., FKSAs), and presentation of the tasks. For example, as the items are designed for primary-aged children, the test should be independent of test-takers' prior programming knowledge and should minimize potential challenges irrelevant to CT (e.g., reading skills, writing skills).

Table 1. Requirements for the Test.

	Requirement
Test format	• The test should be independent of test-takers' prior programming knowledge. • The test should allow large-scale administration. • The tool should enable objective evaluation of students' performance.
FKSAs	• The test items should differ in degrees of difficulty to differentiate levels of competencies related to FKSAs. • Each FKSA should be able to be tested separately, without nesting with others.
Task presentation	• Text descriptions and icons/symbols should be understandable to the test takers. • The test should minimize possible challenges irrelevant to the measured CT constructs (e.g., reading skills, writing skills).

3.1.3. Task Model

Task model depicts the presentation materials needed for accumulating the evidence defined in the evidence model (Mislevy et al., 2003). Based on the requirements from the evidence model, six items were initially designed. The task features are as follows.

- **Item scenario**: A drawing scenario was adopted, which is commonly used in CT assessment tools (e.g., Román-González, 2015; Zhang & Wong, 2023). "Event" was introduced as a function in the scenario (see Figure 1). In the function, when the character, Hungry Snake, eats fruits, it triggers the Pen to draw letters, where different fruits correspond to different letters.

- **Representation for instructions**: Visual alternatives (e.g., arrows) were used to represent the instructions, as it is easier to understand compared to programming blocks, which are deemed suitable for young students (Zhang et al., 2023).

- **Task requirement**: An anchor item was provided to elaborate on how the instructions are executed by the character (see Figure 2). The items are designed based on sequencing the given instructions (inferring the output by stating the instructions in an orderly manner) to solve the problem.

Figure 3-5 presents some example items. Figure 3 demonstrates a task designed for FKSA1 (Ability to identify the output of a set of instructions with simple events), where students were asked to identify what the Pen would draw when Hungary Snake executed the given instructions. Figure 4 is an example for assessing FKSA2 (Ability to identify instructions with simple events to represent a given description), where students need to identify the map where Hungary Snake carried out the instructions, based on what the Pen had drawn. The item displayed in Figure 5 aims to measure FKSA3 (Ability to identify error(s) in a set of instructions that includes simple events). In this task, students were required to detect the error in the map that caused the malfunction of the Pen when drawing the designated letters.

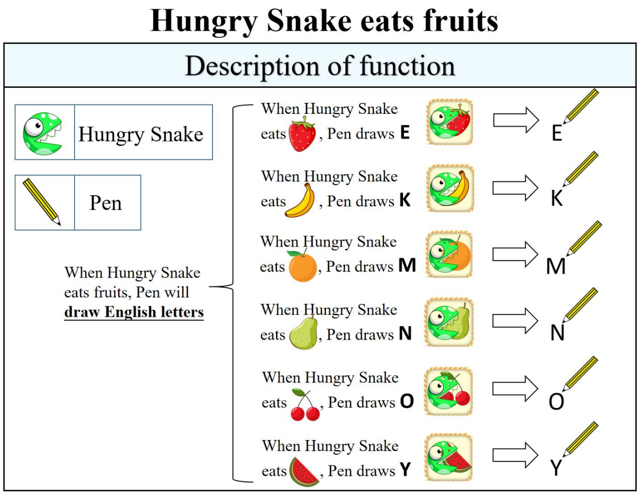


Figure 1. Description of *Event* Function.

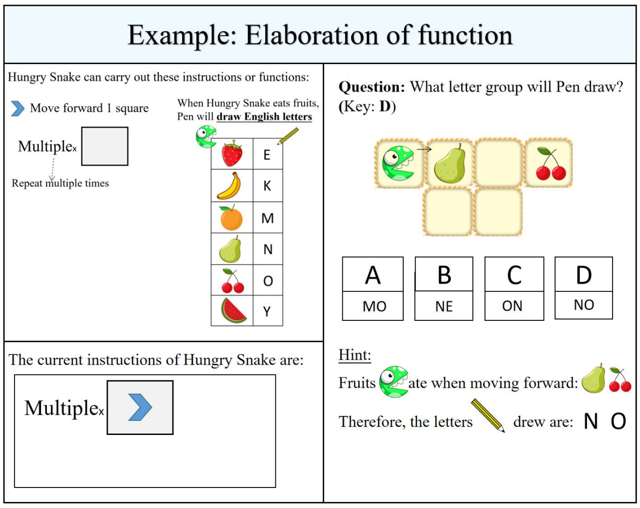


Figure 2. Anchor Item

3.2. Test Validation

To validate the test, a field test was administered. This section will introduce the details of the field test.

3.2.1. Sample

Participants were recruited from a public primary school in China. Students from Grades 2–4 were invited, which aligns with the target group of the test. A total of 639 students agreed to participate, and the characteristics of the sample are presented in Table 2.

Table 2. Characteristics of the Sample.

	Category	N
Grade	2	210
	3	212
	4	217
Gender	Male	323
	Female	289
	Not reported	28

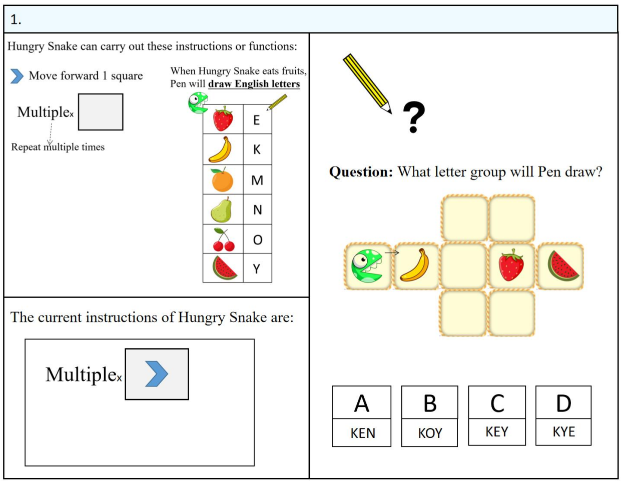


Figure 3. Example Item-FKSA1.

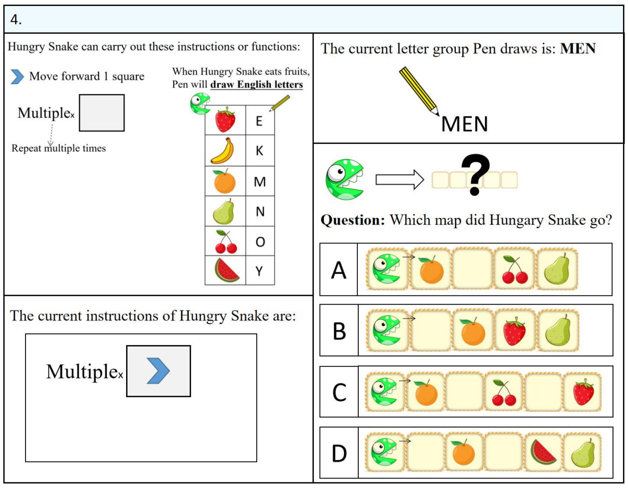


Figure 4. Example Item-FKSA2

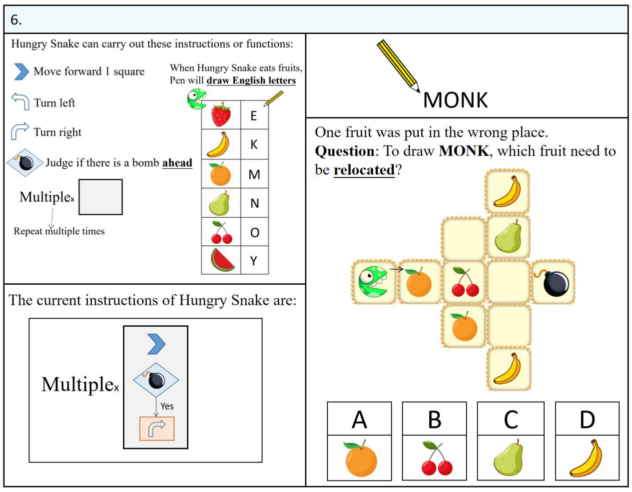


Figure 5. Example Item-FKSA3.

3.2.2. *Data Analysis*

To investigate the psychometric qualities of the test (RQ2), statistical analyses were carried out. Data analysis was performed regarding descriptive statistics, psychometric analyses based on Classical Test Theory (CTT), and psychometric analyses based on Item Response Theory (IRT). After confirming the psychometric qualities of the test, student performance across demographics was investigated (RQ3).

4. RESULT

This section will present the results of the test validation. Descriptive data on the score distribution and student performance will be demonstrated, and the psychometric evidence will be reported based on both CTT and IRT.

4.1. *Descriptive Statistics*

Descriptive data is presented in Table 3. The mean was 2.93 (out of 6), indicating that students performed moderately well. The skewness and kurtosis were within ±2.0, suggesting a proper value for score distribution (George & Mallery, 2019).

Table 3. Descriptive Statistics.

	Mean	SD	Median	Skew	Kurt
Event	2.93	3.76	3.00	-1.17	0.02

4.2. *Psychometric Evidence Based on CTT*

Internal consistency of the test was calculated. Cronbach's α value for the six items was 0.735, above the cut-off point of 0.7 (Nunnally & Bernstein, 1994), suggesting that the test could provide consistent results.

4.3. *Psychometric Evidence Based on IRT*

4.3.1. *Calibration*

Prior to parameter analyses, calibration of the data was performed regarding dimensionality, local independence, and model fit. First, for dimensionality, nonlinear confirmatory factor analysis was conducted in Mplus, and the root mean square error of approximation (RMSEA), comparative fit index (CFI), and Tucker-Lewis index (TLI) were examined. As all the items were designed to measure the same construct, a single-factor model was built. The results reveal RMSEA = 0.052, CFI = 0.976, and TLI = 0.960, which shows a proper fit (Hu & Bentler, 1999), confirming the unidimensionality of the data. Then, model fit for the three basic IRT models, one-parameter logistic (1PL) model, two-parameter logistic (2PL) model, and three-parameter logistic (3PL) model, was checked. Item fit indices for each model were generated with mirt package in R. Results indicate that 2PL model had the best fit, with no misfitting items according to the value of χ^2/df ratio (Aesaert et al., 2014). Therefore, 2PL model was applied for further analysis. Next, local independence of the items was viewed via Yen's Q3 statistics. The results illustrate that three item pairs were slightly higher than 0.2, indicating that local independence was not violated in general (Yen, 1993).

4.3.2. *Item Parameters*

The 2PL model generates two parameters for each item, namely, item discrimination and item difficulty. The acceptable value for the difficulty parameter ranges from -4 to 4 (Baker, 2001), and the desirable value for the discrimination parameter should be higher than 0.5 (Reeve & Fayers, 2005). Results from the field test show that the average difficulty index was 0.074 (range = -0.289, 0.533), indicating that the items had proper difficulty levels. The mean for item discrimination was 1.757 (range = 1.316, 2.444), suggesting that the items could effectively differentiate different levels of performers. The item characteristic curve (ICC) is presented in Figure 6. For each item, the graph displayed an S-shape curve, illustrating that it can effectively discriminate students of different levels. The x value was at a middle range when y equals 0.5, indicating that the difficulty level is moderate. In summary, each item generated proper parameters, suggesting that the test is suitable for the target students.

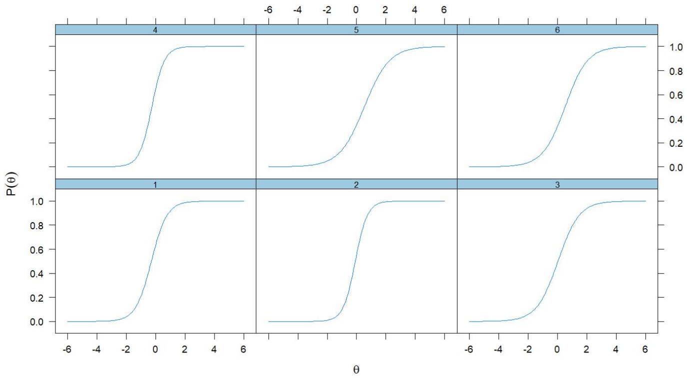


Figure 6. Item Characteristic Curve (ICC).

4.3.3. *Test Information*

Test information in IRT reveals the reliability of the test (Aesaert, 2014). Figure 7 presents the test information curve of the study. The information provided by the test was 4.82 when the ability of a test-taker was -0.1, demonstrating that the test provided the most information about students with average levels. Overall, the test undertakes a broad coverage of ability levels.

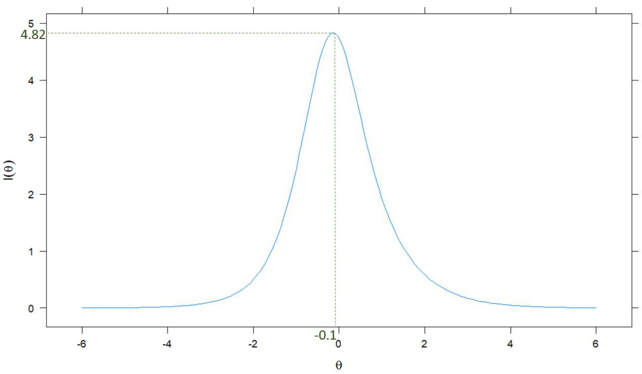


Figure 7. Test Information Curve

4.4. *Analysis of Student Performance*

Student performance was analyzed across grade levels and gender groups. Table 4 presents the performance across the three grades. An ANOVA test was conducted, and a significant difference was examined. The mean increased along with the grade level, ranging from 2.17 to 3.93, indicating that the test was marginally challenging for Grade 2, balanced for Grade 3, and slightly easy for Grade 4. Overall, the test was appropriate for the target age group.

Table 4. Student Performance by Grade.

	N	Mean	SD	F	*p*

Grade 2	210	2.17	1.70	54.263	<0.001
Grade 3	212	2.69	1.88		
Grade 4	217	3.93	1.80		

Further, student performance by gender was explored. T-tests were conducted within each grade, and the results are displayed in Table 5. It is suggested that there was no significant statistical difference between male and female students in each grade, indicating that boys and girls performed equally in the test.

Table 5. Student Performance by Gender.

	Gender	N	Mean	SD	t	*p*
Grade 2	Male	105	2.17	1.78	-0.128	0.898
	Female	99	2.20	1.62		
Grade 3	Male	108	2.75	1.98	0.184	0.854
	Female	87	2.70	1.72		
Grade 4	Male	110	4.05	1.77	0.966	0.335
	Female	103	3.82	1.85		

5. DISCUSSION AND CONCLUSION

This study aims to develop and validate an instrument for measuring the "event" construct in CT for primary school students. A principle approach, ECD, was leveraged for test development, and six items were originally designed. The test was then distributed to 639 2nd–4th graders for validation, and the results indicate proper psychometric indicators based on both CTT and IRT. Student performance was analyzed, where a linear growth by grades was found and no gender gap was detected.

The study contributes to the field by providing a validated assessment tool for the "event" construct, which enriches the pool of CT measurements with a concept that is hardly covered by other tools. The test is independent of any programming context, making it appropriate for novices and learners with different programming language backgrounds and applicable as a diagnostic test for pre–post conditions in educational programs. The instrument has been rigorously validated, and robust psychometric evidence was provided, which can be applied to future research and teaching practices. In addition, the study demonstrates a systematic design process of the measurement, which is replicable for future practitioners to develop their own CT instruments.

The test was easy to administer in primary school classrooms, as reflected by the test invigilators. The format of constructed response allows large-scale data collection in a short period. Yet there are trade-offs to consider for CT test design (Basu et al., 2021). While constructed-response tests tend to be administration-friendly, they can hardly gauge students' practical skills in programming platforms. Thereby, a combination of assessment tools is suggested, where diagnostic tests can be used as summative assessment to capture students' conceptual understanding and project-based measurements can be leveraged as formative assessment to suggest further improvement in programming practices.

Several limitations of the study need to be mentioned. First, as participants were from similar cultural backgrounds, caution is needed when generalizing the results to other cultural contexts. Second, since the Chinese version of the test was used for validation, the English version may need to be piloted before adopting it in English contexts. We welcome future studies to validate the test in different contexts and languages.

6. REFERENCES

Aesaert, K., Van Nijlen, D., Vanderlinde, R., & van Braak, J. (2014). Direct measures of digital information processing and communication skills in primary education: Using item response theory for the development and validation of an ICT competence scale. *Computers & Education*, *76*, 168-181. http://doi.org/10.1016/j.compedu.2014.03.013

Baker, F. B. (2001). *The basics of item response theory*. ERIC.

Basu, S., Rutstein, D. W., Xu, Y., Wang, H., & Shear, L. (2021). A principled approach to designing computational thinking concepts and practices assessments for upper elementary grades. *Computer Science Education*, 1-30. http://doi.org/10.1080/08993408.2020.1866939

Bers, M. U. (2018). Coding and computational thinking in early childhood: The impact of ScratchJr in Europe. *European Journal of STEM Education*, 3(3), 8. http://doi.org/10.20897/ejsteme/3868

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. http://doi.org/10.1016/j.compedu.2013.10.020

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., & Punie, Y. (2016). Developing computational thinking in compulsory education. *European Commission, JRC Science for Policy Report*, *68*.

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.

Bubica, N., & Boljat, I. (2021). Assessment of Computational Thinking–A Croatian Evidence-Centered Design model. *Informatics in education*. http://doi.org/10.15388/infedu.2022.17

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2), 67-69.

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

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. S. (2021). Developing a kindergarten computational thinking assessment using evidence-centered design: the case of algorithmic thinking. *Computer Science Education*, 31(2), 117-140. https://doi.org/10.1080/08993408.2021.1877988

CSTA. (2017). K-12 Computer Science Standards, Revised 2017. https://www.csteachers.org/Page/standards

de Ruiter, L. E., & Bers, M. U. (2021). The Coding Stages Assessment: Development and validation of an instrument for assessing young children's proficiency in the ScratchJr programming language. *Computer Science Education*, 1-30. http://doi.org/10.1080/08993408.2021.1956216

George, D., & Mallery, P. (2019). *IBM SPSS statistics 26 step by step: A simple guide and reference*. Routledge. http://doi.org/10.4324/9780429056765

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. http://doi.org/10.3102/0013189X12463051

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. http://doi.org/10.1080/10705519909540118

Kane, M. (2010). Validity and fairness. *Language testing*, 27(2), 177-182. https://doi.org/10.1177/0265532209349467

Kong, S. C., & Wang, Y. Q. (2021). Item response analysis of computational thinking practices: Test characteristics and students' learning abilities in visual programming contexts. *Computers in Human Behavior*, 122, 106836. https://doi.org/10.1016/j.chb.2021.106836

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. https://doi.org/ 10.1016/j.chb.2014.09.012

McMillan, J. H. (2014). Classroom assessment: Principles and practice for effective standards-based instruction. Pearson.

Mislevy, R. J. (2007). Validity by design. *Educational Researcher*, *36*(8), 463-469. http://doi.org/10.3102/0013189X07311660

Mislevy, R. J., Almond, R. G., & Lukas, J. F. (2003). A brief Introducation to Evidence-Centered Design. *ETS Research Report Series*, *2003*(1), i-29. https://doi.org/10.1002/j.2333-8504.2003.tb01908.x

Nunnally, J. C., & Bernstein, I. H. (1994). The Assessment of Reliability. *Psychometric theory*, *3*, 248-292.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Harvester Press.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New ideas in psychology*, 2(2), 137-168. https://doi.org/10.1016/0732-118X(84)90018-7

Reeve, B. B., & Fayers, P. (2005). Applying item response theory modeling for evaluating questionnaire item and scale properties. *Assessing Quality of Life in Clinical Trials: Methods of Practice*, *2*, 55-73.

Relkin, E., de Ruiter, L. E., & Bers, M. U. (2020). TechCheck: Development and validation of an unplugged assessment of computational thinking in early childhood education. *Journal of Science Education and Technology*, *29*, 482-498. http://doi.org/10.1007/s10956-020-09831-x

Román-González, M. (2015). Computational thinking test: Design guidelines and content validation. *Proceedings of EDULEARN15 conference*.

Rowe, E., Asbell-Clarke, J., Almeda, M. V., Gasca, S., Edwards, T., Bardar, E., Shute, V., & Ventura, M. (2021). Interactive Assessments of CT (IACT): Digital interactive logic puzzles to assess computational thinking in Grades 3–8. *International Journal of Computer Science Education in Schools*, 5(2), 28-73. http://doi.org/10.21585/ijcses.v5i1.149

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 103798. http:// doi.org/10.1016/j.compedu.2019.103798

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, 20(4), 715-728. https://doi.org/10.1007/s10639-015-9412-6

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

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. http://doi.org/10.1098/rsta.2008.0118

Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education & Information Technologies*, 24(1), 929-951. https://doi.org/10.1007/s10639-018-9801-8

Yen, W. M. (1993). Scaling performance assessments: Strategies for managing local item dependence. *Journal of Educational Measurement*, *30*(3), 187-213. http://doi.org/10.1111/j.1745-3984.1993.tb00423.x

Zhang, S., & Wong, G. K. W. (2023). Development and validation of a computational thinking test for lower primary school students. *Educational Technology Research and Development*, 71(4), 1595-1630. https://doi.org/10.1007/s11423-023-10231-2

Zhang, S., Wong, G. K. W., & Chan, P. C. F. (2023). Playing coding games to learn computational thinking: What motivates students to use this tool at home? *Education and Information Technologies*, 28(1), 193-216. https://doi.org/10.1007/s10639-022-11181-7

The Construction and Application of a Pedagogical Framework for Primary English Curriculum Based on Computational Thinking

Xinlei LI[1*], Guoyuan SANG[2], Martin VALCKE[3], Jinbao ZHANG[4], Johan VAN BRAAK[5]
[1,3,5] Department of Educational Studies, Ghent University, Belgium
[2,4] Faculty of Education, Beijing Normal University, China
xinlei.li@ugent.be, guoyuan.sang@bnu.edu.cn,
martin.valcke@ugent.be, zhangjb@bnu.edu.cn, johan.vanbraak@ugent.be

ABSTRACT
The primary English curriculum attaches importance to learning and teaching mode changes, the quality of thinking, and the development of problem-solving skills in a modern information technology context in China. Therefore, paying attention to the integration of computational thinking (CT) with the primary English curriculum in the design and implementation is an effective way of creatively solving subject problems and promoting curriculum development. This paper constructs a pedagogical framework for the integration of CT with the primary English curriculum from a disciplinary perspective, exploring the principles, elements, and processes of curriculum integration. In this study, CT is integrated into the primary English curriculum through a task-driven teaching and learning design in terms of knowledge and skills, processes and strategies, attitudes, and values. It is hoped that this paper will inspire discussion among teachers about CT in the teaching of subject curricula, particularly needed in language subjects. Moreover, students are enabled to develop a systematic and scientific mode of thinking and inquiry skills, with structured knowledge and competence in English language disciplines and interdisciplinary study in a digital society.

KEYWORDS
computational thinking, primary English curriculum, pedagogical framework

1. INTRODUCTION
Since the 21st century, CT as a thinking skill has been highly valued in the education reforms of many countries to meet the current needs of society and individuals in the information society, and they have also affected the development of discipline curriculum design and implementation. The integration of CT in lessons can be effective in helping students and teachers improve their knowledge, skills, and attitudes (Bocconi et al., 2016). There are several initiatives to actively integrate CT into the curriculum across different subject areas at international, national, and school levels, especially in primary grades (Israel et al., 2022; Voogt et al., 2015). Research on CT in primary schools indicated that the best pathway to provide CT to students at this age level may be through its integration into core subjects (Duncan et al., 2017; Yadav et al., 2016). In K-12 classrooms, researchers argued that not only STEM classrooms but also language arts can be integrated with CT skills through activities (Barr & Stephenson, 2011). It is necessary to enhance more opportunities for the CT integrated into the curriculums of the language arts, social sciences, and art disciplines (Yeni et al., 2024).

Digital education and curriculum reform in line with national conditions are being actively explored in China, as evidenced by the focus on CT in subjects. English as a foreign language subject in compulsory education in China aims to improve students' language competence, cultural awareness, quality of thinking, and learning competence (Ministry of Education, 2022). The newly revised curriculum standard focuses on the essence of education, takes the core competences as the overarching principle, and integrates all aspects including curriculum objectives, curriculum content, teaching implementation, and teaching evaluation to construct a panoramic blueprint for educating people in the foreign language discipline in the new era (Ministry of Education, 2022). To meet the needs for social and personal development, English lessons emphasize the changes in learning styles and teaching modes in the context of modern information technology. It emphasizes the enhancement of students' thinking skills in identifying, analyzing, and solving problems through the teaching of English lessons. Additionally, educators can improve English subject lessons with CT, even in primary schools without a related computational science subject, to achieve relative equity in education (Jacob et al., 2022). Therefore, it is significant to establish an understanding of what CT is and how to integrate it into the classroom in different educational contexts.

Based on the above analysis, the main problem addressed in this study focuses on the effective construction of a pedagogical framework for the integration of CT with the primary English curriculum. To explore the application of a pedagogical framework for CT integrated into primary school English lessons in an attempt to improve English language curriculum development. It aims to develop knowledge, skills, and attitudes of students' subject learning through CT, as well as their ability to use computer science and technology to solve practical problems.

2. BACKGROUND
The application and popularity of CT in curriculum development provide researchers with opportunities to define it at various stages and perspectives. Humans use CT as a method to solve problems (Wing, 2006). Previously, CT has been described as applying computational methods to tackle daily and interdisciplinary problems (CSTA & ISTE, 2011; Hsu et al., 2018). Wing (2006) first emphasized that CT involves "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to

computer science" (p. 33). CT involves a cohesive combination of skills and methodologies aimed at solving complex problems, serving as a way to learn topics across various disciplines, and is imperative for complete engagement in a computational world (Mills et al., 2021). Besides, CT can be seen as the cognitive processes entailed in the formulation of problems, enabling the representation of their solutions through a structured sequence of computational steps and algorithms (Grover & Pea, 2013). CT is also considered from computational concepts (sequences, loops, events, parallelism, conditionals, operators, data), computational practices (problem-solving practice through experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing), and computational perspectives (expressing, connecting, questioning) (Brennan & Resnick, 2012). Some researchers defined CT through its core components, such as decomposition, abstraction, algorithm design, evaluation, and generalization (Selby & Woollard, 2013).

Supporters of an integrated approach assert that integrating CT into a core subject is essential for reaching all students in instruction (Weintrop et al., 2016). Additionally, it provides teachers with the opportunity to construct their existing content and pedagogical knowledge when introducing CT in their classrooms (Rich et al., 2019). In addition to STEM subjects, CT can also be used to teach students in fields such as social studies, art, music, English (Garvin et al., 2019) and English as a second language (Jacob et al., 2018). Teachers attempt to explore potential teaching opportunities for integrating CT with the English language curriculum. Researchers have initiated the design and development of pedagogical frameworks aimed at facilitating the implementation of CT within the English language curriculum, such as the PRADA (Pattern Recognition, Abstraction, Decomposition, Algorithms) model (Dong et al., 2019) and CDIO (Conceive-Design-Implement-Operate) framework (Hladik et al., 2017). However, there is a notable lack of frameworks addressing the connection of CT, language, and literacy learning to guide instructional practices specifically tailored for culturally and linguistically diverse learners (Jacob et al., 2022). The previous pedagogical framework design primarily focuses on integrating core components of CT with the English language curriculum, lacking attention to the integration of attitudes, culture, and value knowledge. The research requires enhancing the systematic and progressive nature of the curriculum design framework. This study attempts to fill the gap in the design of primary school English curriculum pedagogical frameworks based on CT within the Chinese educational context. It aims to deepen the understanding of the integration relationship between CT and English language curriculum, assisting teachers in effectively guiding students' learning through instruction.

3. THE CONSTRUCTION OF A PEDAGOGICAL FRAMEWORK

The construction of the pedagogical framework requires an in-depth understanding of the principles and components of the integration of English curriculum standards and CT. In this study, CT is defined as a thought process for solving problems with or without technology and can be represented through knowledge and skills, process and methods, attitudes and values. We also focus on the corresponding relationship between CT and the primary English curriculum standard (2022 version).

3.1. The Construction Principles of Pedagogical Framework

The construction principles of the pedagogical framework combine different subjects and their needs. The primary English curriculum standards emphasize the goal-orientation, problem-orientation, and innovation-orientation of the curriculum design principles (Ministry of Education, 2022). This study developed the following principles in order to guide educators constructing pedagogical frameworks that are effective, adaptable, and responsive to the evolving needs of learners in diverse educational settings.

1) Ensure that the pedagogical framework is consistent with the overall educational goals.

2) Clearly define the purpose and expected outcomes of the framework and provide a roadmap for educators and learners.

3) Encourage an interdisciplinary content and approach that strengthens the links between different disciplines and promotes problem solving.

4) Utilize a variety of teaching and learning environments (e.g. plugged and unplugged settings, etc.) to enhance and support a framework that prepares students for the innovative demands of life.

5) Ensure that the framework is culturally relevant, recognizing and respecting the diversity of students' backgrounds and experiences.

6) Incorporate regular reflection and evaluation processes to assess the effectiveness of the framework, get feedback from educators and students, and make continuous improvements.

3.2. The Components of Pedagogical Framework

The TPACK framework could be a helpful model for integrating CT within the subject matter and pedagogical techniques that teachers would teach in their future classrooms (Yadav et al., 2017). The concept of TPACK can be found over time through the a series of publications (Mishra & Koehler, 2006). The teacher's knowledge is composed of three main parts: content, pedagogy, and technology in this model. The interaction between these components represents TCK (technological content knowledge), TPK (technological pedagogical knowledge), PCK (pedagogical content knowledge), and TPACK (technological, pedagogical, and content knowledge) in this model. Moreover, the term "TPACK" refers to the context-based integration of technology, pedagogy, and content knowledge for CT concepts, practices, and perspectives (Kong & Lai, 2021). This research is based on the TPACK model as an outline to identify that CT can be integrated into the English language curriculum teaching and learning in a digital environment. Therefore, the component of the pedagogical framework in this study focuses on CT, English language, pedagogy, and technology resources.

3.3. The Overview of Pedagogical Framework

The construction of the pedagogical framework focuses on the relationships among CT (knowledge and skills, process and methods, attitudes and values), English language curriculum (English language content and interdisciplinary content), pedagogy (curriculum implementation design and curriculum instructional strategies), technology and resources (plugged resources, unplugged resources, internal resources, and external resources) (see Figure 1). Within the integrated context of diversity, the integration of these four elements facilitates the attainment of curriculum objectives and expected outcomes to assist teachers and students in further reflecting upon and evaluating the effectiveness of curriculum design and implementation. Furthermore, the pedagogical framework considers students' learning stages and proficiency, emphasizing the continuity, sequence, and progression from low to high levels.

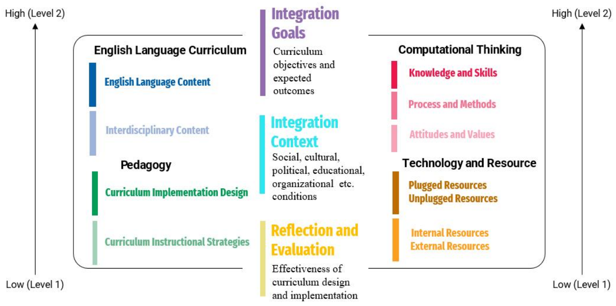


Figure 1. The Overview of Pedagogical Framework

4. THE APPLICATION EXAMPLES OF A PEDAGOGICAL FRAMEWORK

4.1. Objectives and Content Design of Primary English Curriculum Based on CT

In this study, CT is integrated into the primary English curriculum through a task-driven teaching and learning design in terms of knowledge and skills, processes and strategies, attitudes and values. The integration teaching objectives focus on English language competence, computational thinking, culture awareness, and learning competence. This example described and developed how CT integrated into primary English lessons with interdisciplinary content in grade 5 in Beijing, China. The course teaching content was designed based on two units (Unit 2 What do Flowers do? & Unit 3 How do seeds travel?) of the Beijing Publishing House's Primary English Grade 5 textbook in which natural science is an interdisciplinary topic. Among them, the interdisciplinary theme content guides students to understand individuals, society, and nature.

Firstly, teachers guided students to understand and apply the basic knowledge and concepts of CT such as data collection, data representation, decomposition, pattern recognition, pattern generalization, abstraction, and algorithm design. Through debugging, cooperation, and innovative computational thinking-related methods, students can effectively carry out learning activities. Students can browse, search, filter, evaluate, and manage data and digital content to solve problems in tasks. Through the completion of task assignments, teachers guide students to have confidence and persistence in solving complex problems. They develop their ability to communicate and cooperate with others to achieve common goals or solutions in games and group work.

Secondly, teachers organize activities to help students learn the English language and interdisciplinary content. Students are able to recognize sentence patterns related to the names of plant parts and vocabulary for plant functions and be able to apply them in relevant situations. Students can understand, comprehend, and read discourse. In the activity, students saw the pictures of different plants and objects appearing in the PPT, the teacher led the students to observe and discover the similarities and differences among objects. Based on the above observations and findings, teachers let students learn new sentence patterns to describe the similarities and differences between different plants and objects: "What's the English for the different parts of a …? They are … ." "What do…do? They… ". Teachers guided students to identify problems in plant science and attempt to solve them through observation, hands-on practice, and independent inquiry.

Finally, students can understand the function of the role of plants, growth processes, and other scientific knowledge through English sentence patterns, algorithm design, and story reading about plants in traditional Chinese culture. For example, the teacher shows an animation of the plant growth process and growing conditions and asks students "What do the leaves do to make food?" And students try to answer with "They need (sunlight, water, nutrients, etc.) ….". Based on the information in the animation, the teacher summarizes the sequential process of plant growth and allows students to sequence and design algorithms. Students completed the following tasks to put the correct order below and design the algorithm by themselves through group work (see Figure 2). Students are able to work through debugging, collaboration, and creative approaches related to CT. Additionally, students can carry out learning activities effectively through teachers' instruction.

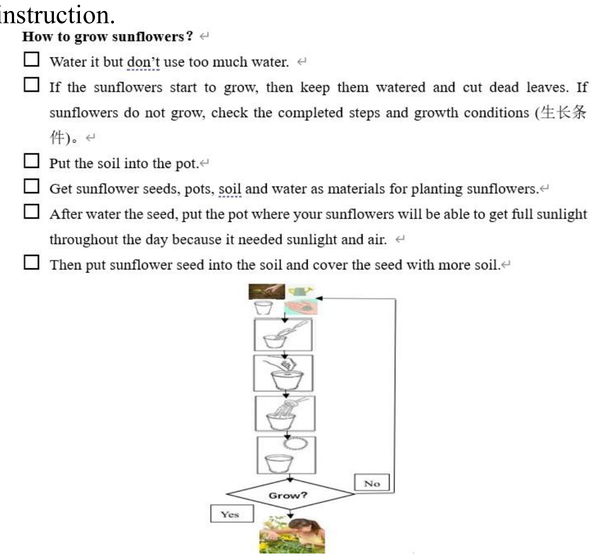


Figure 2. The Task of Activity

4.2. Implementation and Evaluation of Primary English Curriculum Based on CT

The implementation strategies were unplugged and plugged activities to enhance students' language and interdisciplinary learning performance in terms of knowledge and skills, processes and methods, and attitudes and values. Students test their performance after lessons' implementation by completing lessons' exercises. A semi-structured interview method was used for six of their teachers in Beijing, China. This research carried out in-depth interviews and text transcriptions, coding from cognition and implementation status. Teachers' interviews mainly include conceptual understanding of CT, implementation design, and strategies of primary school English curriculum based on CT.

From a student learning perspective, there is a learning challenge and an adaptation process when students do activities integrated with CT that are different from those previously conducted in school. In this lesson, students used video, PowerPoint, electronic whiteboard, A4 paper, pen, vocabulary card, and textbooks to learn. There is still a lack of support from the technological environment for students to use plugged CT activities in their everyday classroom learning. Students with high levels of English language learning are used to learning English in an unplugged way and have a strong motivation and interest in unplugged CT learning. Flowcharts, relational diagrams and pictures help students to break down complex problems and understand interdisciplinary and English subjects very effectively and logically through results of completion of classroom activity tasks. Students with high scores are accustomed to using tools such as the Internet, multimedia, and books to find information and solve complex interdisciplinary and open-ended problems. Students enjoy learning and engaging in fun and innovative knowledge and activities.

After the classroom implementation, teachers found that students have improved their understanding of CT and English learning to a certain extent, and the teacher's teaching effect has achieved the expected outcomes. Teachers' awareness of CT concepts has deepened following the curriculum implementation, and it is necessary to integrate CT into primary English lessons. The integration of CT into the primary English curriculum needs to be supported by national policy, local government, school leaders, experts for school hardware and facilities, and computer science professionals. The primary school English teachers who are interviewed believe that the integration of CT and primary school English curriculum needs to follow the principles of involvement and effectiveness. Primary English teachers need to focus on students' interest in learning, interdisciplinary knowledge, and English language content in the process of curriculum integration. Teachers' awareness of CT and teaching skills still need to be strengthened and training is a current challenge. Teachers urgently need the support of practical cases, guide books, and resource packs, so that teachers can operate more conveniently and effectively in the process of teaching practice.

5. DISCUSSION AND CONCLUSIONS

This study inspires discussion among teachers about CT in the teaching of subject curricula, particularly needed in language subjects. The integration process should be concerned not to increase the teachers' workload. It is only necessary for teachers to make further improvements on the original basis. There are challenges in terms of the feasibility of implementing lessons in different schools, for example, designing problem solve tasks and inquiry-based classrooms, which often take longer time for collaborative inquiry discussions. The pedagogical framework can help teachers to design activities to achieve their objectives more effectively and to solve real-life problems through lesson activities. Existing research also shows that pattern recognition is the most acceptable concept for students to integrate into English language learning. The activities that students found most difficult were those related to abstraction and creativity, and students felt they faced difficulties with both CT and the English language in this study. The unplugged and plugged activities can both support students in English and CT learning. The effective connection between curriculum objectives and CT concepts makes the implementation of learning activities effective and feasible. Students have positive attitudes towards CT for learning in the primary English curriculum and are motivated by learning interest, peer communication, learning styles, inquiry-based tasks, and interdisciplinary knowledge. Teachers' CT awareness and teaching skills need to be strengthened, trained, and practical support. Due to time and conditions constraints, research needs to further understand the construction of the pedagogical framework. Therefore, future studies are encouraged to develop creative ways for students and teachers to integrate CT into primary English lessons. The limitation of this study was that the participants were in fifth grade in elementary schools, it is suggested that empirical studies involving students of different ages be conducted in the future.

6. REFERENCES

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education – Implications for policy and practice*. Publications Office of the European Union. https://doi.org/10.2791/792158

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, BC, Canada*.

CSTA & ISTE. (2011). *Operational definition of computational thinking for K-12 education.* Retrieved November 16, 2023, from https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Dong, Y., Cateté, V., Jocius, R., Lytle, N., Barnes, T., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2019).

Prada: A practical model for integrating computational thinking in K-12 education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 906–912). https://doi.org/10.1145/3287324.3287431

Duncan, C., Bell, T., & Atlas, J. (2017). What do the teachers think? Introducing computational thinking in the primary school curriculum. In *Proceedings of the Nineteenth Australasian Computing Education Conference* (pp. 65–74). New York: ACM. https://doi.org/10.1145/3013499.3013506

Garvin, M., Killen, H., Plane, J., & Weintrop, D. (2019). Primary school teachers' conceptions of computational thinking. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 899–905). ACM. https://doi.org/10.1145/3287324.3287376

Grover, S., & Pea, R. (2013). Computational thinking in K – 12 : A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Hladik, S., Behjat, L., & Nygren, A. (2017). Modified CDIO framework for elementary teacher training in computational thinking. In *Proceedings of the 13th International CDIO Conference*. Calgary, Canada.

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. https://doi.org/10.1016/j.compedu.2018.07.004

Israel, M., Liu, R., Yan, W., Sherwood, H., Martin, W., Fancseli, C., Rivera-Cash, E., & Adair, A. (2022). Understanding barriers to school-wide computational thinking integration at the elementary grades: Lessons from three schools. In *Computational thinking in PreK-5: Empirical evidence for integration and future directions* (pp. 64–71). ACM. https://doi.org/10.1145/3507951.3519289

Jacob, S., Nguyen, H., Tofel-Grehl, C., Richardson, D., & Warschauer, M. (2018). Teaching computational thinking to English learners. *NYS TESOL Journal*, *5*(2), 12–24.

Jacob, S. R., Parker, M. C., & Warschauer, M. (2022). Integration of computational thinking into English language arts. In *Computational Thinking in PreK-5: Empirical Evidence for Integration and Future Directions* (pp. 55-63). https://doi.org/10.1145/3507951.3519288

Kong, S. C., & Lai, M. (2021). A proposed computational thinking teacher development framework for K-12 guided by the TPACK model. *Journal of Computers in Education*, *9*(3), 379–402. https://doi.org/10.1007/s40692-021-00207-7

Mills, K., Coenraad, M., Ruiz, P., Burke, Q., & Weisgrau, J. (2021). Computational Thinking for an Inclusive World: A Resource for Educators to Learn and Lead. In *Digital Promise*. https://doi.org/10.51388/20.500.12265/138

Ministry of Education of the People's Republic of China. (2022). *English curriculum standards for compulsory education (2022 version)*. Beijing: Beijing Normal University Publishing Group.

Mishra, P., & Koehler, M. J. (2006). Technological Pedagogical Content Knowledge: A Framework for Teacher Knowledge. *Teachers College Record*, *108*(6), 1017–1054. https://doi.org/https://doi.org/10.1111/j.1467-9620.2006.00684.x

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019). Computational thinking, mathematics, and science: Elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education*, *27*(2), 165–205.

Selby, C. C., & Woollard, J. (2013). Computational thinking : The developing definition. . In *Paper presented at the 18th annual conference on innovation and technology in computer science education*, Canterbury.

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.

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*, 127–147. https://doi.org/10.1007/s10956-015-9581-5

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. https://doi.org/10.1007/s11528-016-0087-7

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591

Yeni, S., Grgurina, N., Saeli, M., Hermans, F., Tolboom, J., & Barendsen, E. (2024). Interdisciplinary Integration of Computational Thinking in K-12 Education : A Systematic Review. *Informatics in Education*, *23*(1), 223–278. https://doi.org/10.15388/infedu.2024.08

Developing Computational Thinking through Sport Games in Primary Grades

Bingqi Yao[1]
[1]Wufu Road Primary School, China
cokyao@126.com

ABSTRACT
Computational thinking (CT) is the most essential mind-set for people to live in intelligent society which almost exists in every fields, therefore, fostering the primary grades' CT in schools is the most urgent matter. The literature on the subject shows that programming is the key to develop children's CT which essentially demonstrates how computers process problems and is known as algorithms. However, many schools in China even can not offer programming education because of teaching facilities limitations. In the background of above issues, the paper took selection sort and binary search algorithms as examples to design the learning subject named Looking for Lucky Boy or Girl, which was aimed to explore not only the effectiveness of interdisciplinary training low-grades' CT in sports games but also low-cost programming education for the primary grades in school. Besides, the research was conducted by case study method whose steps consisted of designing, implementation, assessment and optimization. The study applied the pre-test and post-test in the same group whose data was involved in paired samples t-tests by SPSS. The statistics indicated that there was extremely marked difference between pre test and post test score, the student even behaved better in post-test. On the basis of results, it found that integrating algorithms learinng into sports games can effectively promote the primary grades's CT in schools. In a word, it is feasible to interdisciplinary develop students' CT by sports games in early primary schools.

KEYWORDS
algorithms, computational thinking, early primary schools, interdisciplinary, sports games

1. INTRODUCTION
Computational thinking is used to refer to not only a series of tasks which computers executes to solve the problems, but also several steps which people integrating information technology into problem-solving process. Nowadays, computational thinking is as important as reading and writing skills for people which has outstanding interdisciplinary characteristics. Excluding computer science, computational thinking is embedded in lots of subject fields such as mathematics, physics, Chemistry, etc. However, there were few researches indicated how to foster students' computational thinking across disciplines. Besides, many schools in China can not offer computer education. Therefore, it's more and more urgent to explore the cheap, simple, general ways to improve students' computational thinking level.

Based on above issues, the study took selection sort and binary search algorithms as examples to design learning activities and applied case study method to explore whether integrating sorting algorithms principles to sports games in lower primary school can effectively promote students' computational thinking level.

2. LITERATURE REVIEW

2.1. Computational Thinking
Computational thinking emerged with the birth of computers. At that time, people knew little about what computational thinking was. It was not until 2006 that Jeannette M. Wing defined computational thinking as how computers address problems, design systems and understand human with fundamental concepts about computer science. In addition, Professor Chou pointed out that computational thinking was extremely related to mathematical thinking and engineering thinking. With the development of information technology, almost all subjects are challenged by computerization which means there are distinctive computational models in different subject fields and they rely on computers to simulate and solve research problems. In summary, computational thinking refers to the problem-solving strategic mechanism which are formed as follow: decomposition, abstraction, modelization and automation.

2.2. Learners' Features
According to Piaget's cognitive-developmental theory, students in primary grades are in the key period which turn pre-operation to practical operation stage. In other words, students' image thinking is evolving to conceptual thinking which refers to students should understand and experience computational thinking in terms of concrete entities. GUI programming, such as scratch, mBlock, Arduino and so on, is becoming more and more popular which make computational thinking learning tangible, visual and easy to mastery. Moreover, lower grades in primary schools is the crucial period for students to develop sports skills and interests, so making full use of the period to train the physical abilities is an urgent matter.

2.3. Programming Education
Programming skills is the most indispensable part of information literacy and artificial intelligence literacy, which manifests computer coding is the crucial carrier for computational thinking learning. From 19th century on, many countries around the world put the programming education on the front burner, which included coding in curricula for primary schools and even kindergarten. However, computer screen is harmful for students' eyesight. As a result, screen-free programming education was proposed which indicates students learning how to code with devices, toys and games. There are two kinds of screen-free programming education devices. One kind is programming robots such as Bee-Bot, Turtles, Cubetto, etc. The other is mechanical devices namely Code Monkey Island, Mindware Code Hopper, etc. Generally, the

physical devices are expensive, thus many researches turn to design computational thinking learning games with stickers, cards and blocks and so on instead of expensive devices.

The computers execute a sequence of instructions to solve problems which are known as algorithms. Therefore, understand and experience algorithms is the primary issue comparing with programming. The world is totally unordered and everything could be digitalized and symbolized, thus, computers should put data in specific order before solving problem which is also represented as sorting algorithms. In addition, query is also a critical part of computational thinking which is represented as query algorithms. Based on learners' feature of lower grades in primary school, the binary search algorithm would be incorporated in the research.

2.4. Interdisciplinary Learning

Chinese Information Technology Curriculum Standards for Compulsory Education claimed that interdisciplinary learning should be incorporated into the course. In sports games, students imitate computers to interact with the world in their body language, which is a low even zero cost way to develop computational thinking. It's really an excited news for the school which can not offer programming education.

In sports games, students move their body to simulate algorithms which is as same as entities such as stickers, cards, and wooden blocks in physical programming devices. For example, there is a sport game combining sports skills such as walking, running, jumping, etc with cycle structure in programming. The game asks students to follow the teacher's instructions and take corresponding accordingly. If the instruction is given as Jumping Until Reaching Red Flag or Stopping Walking When Meeting Yellow Flag, they correspond with the repeat-until loop and do-while loop in programming education respectively.

3. METHODOLOGY

3.1. Participants

Students of first grade is in bad class discipline and even could not understand algorithms, therefore, the study took students in first class, second grade of F Primary School in Shantou city as research subjects, 30 participants in total. F Primary School could not offer any computational thinking learning activities for the lower grades. In the view of present issues, sports games are simple, economical and practical carriers for programming education.

3.2. Research Methods

The research adopted case study method to explore whether developing low-grade students' computational thinking through sports games is feasible. The method consists of four parts that are as follows by designing, implementing, accessing and optimizing classroom activities.

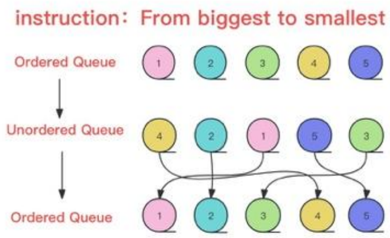
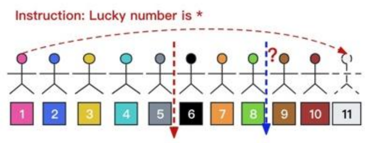
3.3. Teaching Activities Design

The learning theme is named as Looking for the Lucky Boy or Girl, which students will experience and ponder over how selection sort algorithms work through queue formation sport game and how binary search algorithms work through the sport game named Looking for the Lucky Boy or Girl.

It will take three classes to complete the learning subjects, one class is for pre-tests and sports games preparations, then the second class is for queue formation sport game, finally, the last class is for Looking for the Lucky Boy or Girl and post-tests.

The specific teaching steps are showed in Table 1 which contains four parts, namely, beginning, preparing, main and ending parts.

Table 1. Teaching Activities Design

Teaching Steps	Teaching Activities
Beginning part	Greeting Roll-Call
Preparing part	Warm-up Exercises
Main part	(Case 1)Firstly, the students (30 in total) should be divided into six groups, each group are numbered off one by one from the left to the right. Secondly, each group will be reshuffled and make students each group in a different order. Thirdly, the teacher give the instruction such as From Biggest to Smallest, each group finally shape the new ordered queue.  (Case 2)Firstly, the students will be divided into three groups, students of each group are numbered. Secondly, after teacher offers the lucky number, students will be in continuous divided into two equal group until the lucky boy or girl is found. Last but not least, if odd number is existed, add a student to the group. 
Ending part	Relaxation Exercises Summary

3.4. Evaluation Design

The research set answer at three stages which contains answers, reasons and confidence according to findings of misconceptions and certainty of response index.

Besides, it’s also shown in table 2 that the scores are classified into into four categories, namely, having solid knowledge, self-doubt, having weak knowledge, having poor knowledge.

Table 2. Evaluation Criterion

	reasons	confidence	scores
solid knowledge	√	√	3
self-doubt	√	×	2
weak knowledge	×	×	1
	√	×	1
	×	√	1
	√	√	1
poor knowledge	×	√	0
	×	×	0

The assessment instruments was made on the base of evaluation criterion above and would be used in pre-test and post-test. As table 3 shown, question 1 is used to test whether students have limited knowledge about ordinal and sorting, 2 and 3 are used to assessed students’ selection sort algorithms level, 4 and 5 are for binary search algorithms measurement .

Table 3. Assessment Instruments

Assessment Items

Ordinal and Sorting

1.1 The students lined up in a column according to their height. There were 5 students in front of Max. Where did Max rank?

A. Fourth B.Fifth C.Sixth D.Seventh

1.2 Why did you choose the answer?

A. Five minus one is four.

B. Five and zero makes five.

C. Five and one makes six.

D. Five and two makes seven.

E. The other reason.

1.3 Is you answer true? A.true B.False

Selection Sort Algorithms

2.1 Please rank the numbers from biggest to smallest.

3	1	4	2

2.2 Why made you do that?

2.3 Is you answer true? A.true B.False

3.1 How many times did you make the selection?

A. 1 B. 2 C. 3 D. 4

3.2 Why did you choose the answer?

A. I make once selection.

B. I make twice selections.

C. I make three times selections.

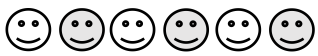
D. I make four times selections.

E. The other reason.

3.3 Is you answer true? A.true B.False

Binary Search Algorithms

4.1 Please divide the row below into two parts evenly.



4.2 Why made you do that?

A. Dividing 3 into 6 equals 2.

B. Dividing 2 into 6 equals 3.

C. Dividing 1 into 6 equals 6.

D. Dividing 6 into 6 equals 1.

E. The other reason.

4.3 Is you answer true? A.true B.False

5.1 Cut the numbers in half each time, how many times did you cut the numbers from 1 to 100 can exactly meet the number of 25.

A. 1 B. 2 C. 3 D. 4

5.2 Why did you choose the answer?

A. Dividing the numbers in half once can meet the number of 25.

B. Dividing the numbers in half twice can meet the number of 25.

C. Dividing the numbers in half three times can meet the number of 25.

D. Dividing the numbers in half four times can meet the number of 25.

E. The other reason.

5.3 Is you answer true? A.true B.False

4. RESULTS AND CONCLUSIONS

4.1. Mean

The study distributed 30 tests and received 30 both of pre-test and post-test, response rate of our test is 100%.

Table 4. Mean Scores

	Pre-Test	Post-test
Q1	1.70	2.83
(Q2+Q3)/2	0.35	2.57
(Q4+Q5)/2	0.52	2.03
Mean(Q1-Q5)	0.69	2.41

According to table 2 and table 4, the students behaved better in post-test. On the dimension of Ordinal and Sorting, students get the mean of 2.83 after sports games learning which manifests that they almost have solid knowledge about ordinal and sorting. The children also make obvious progress in comprehending algorithms of selection sort and binary search. They get mean of 2.57 on the dimension of selection sort algorithms, and get mean of 2.03 on the dimension of binary search algorithms which is far greater than 1(Having weak knowledge) .

4.2. Normality Test

Normality test should be taken before adopting paired samples test, the results is shown below.

Table 5. Normality Test

	Shapiro–Wilk Test		
	Statistic	df	Sig.
Pre-Test	0.981	30	0.132
Post-Test	0.953	30	0.198

As table 5 shown, the research do normality test both pre test and post test, whose results were Sig.>0.05, it indicated the data corresponded to normal distribution.

4.3. Paired Samples Test

The data from pre-test and post-test was involved in paired samples t-tests by SPSS whose statistics is shown as table 6.

Table 6. Paired Samples Test

Paired Differences			
Mean	t	df	Sig. (2-tailed)
1.72	-15.592	30	0.000

According to table 6, there was extremely significant difference between the pre-test and post-test data, moreover, students got greater scores in post-test.

4.4. Conclusions

Nowadays, computational thinking is the most critical mind-set for people which almost exists in every subject fields. It is also well-known as algorithms which essentially demonstrates how computers process problems. The literature on the subject shows that programming is crucial carrier for cultivating children's computational thinking. However, many schools in China even can not offer programming education because of economic constraints. In the background of above issues, the paper took selection sort and binary search algorithms as examples to design learning subject, which was aimed to explore not only the effectiveness of interdisciplinary training low-grades' computational thinking in sports games but also low-cost programming education.

Statistics above indicated that there was extremely marked difference between pre test and post test, and students even behaved better after learning activities. On the basis of statistical analysis, it found that integrating algorithms principles into sports games can effectively promote the development of children's computational thinking in schools. Therefore, it is feasible to interdisciplinary foster students' computational thinking by sports games in early primary schools. Besides, sports games is a simple, economical and practical way to train computational thinking.

The follow-up studies will be conducted by designing more and more teaching cases corresponding to students' learning features in the hope of incorporating more kinds of algorithms into sports games.

5. REFERENCES

Bai, Mingli Xie, Qi. (2023). The value implication and implementation path of screenless programming teaching for young children - based on the perspective of computational thinking cultivation. Educational Science Forum (17), 19-23.

Chen Guo-Liang,Li Lian Dong Rong-Sheng. (2020). Toward Computational Thinking 2.0.Chinese University Teaching(04),24-30.

Duo Zhaojun,Liu Yansong Ren Yonggong. (2022). Research on the intrinsic mechanism and teaching practice of programming education to promote children's computational thinking development. Research on Electrochemical Education (08), 101-108. doi:10.13811/j.cnki.eer.2022.08.013.

Hasan, S., Bagayoko, D & Kelley, E. L. (1999). Misconceptions and the Certainty of Response lndex (CRI)UJ]Physics Education, 34(5) 294-299.

Kang, Jianzhao. (2021). The origin, practice path and characteristics of programming education in Finnish primary and secondary schools. Research on Electrochemical Education (08), 101-107+115. doi:10.13811/j.cnki.eer.2021.08.014.

W.W. Lo, H.R. Chan, T.E. Liu Ilene R. Berson.(2020). The Rise, Goals, and Practices of Screenless Programming Education - An Analysis of Twitter Big Data Based on Social Media Research Methodology (SMR). Journal of Distance Education (05), 101-112. doi:10.15881/j.cnki.cn33-1304/g4.2020.05.011.

W.W. Lo, H.R. Chan, T.E. Liu Ilene R. Berson.(2020). The Rise, Goals, and Practices of Screenless Programming Education - An Analysis of Twitter Big Data Based on Social Media Research Methodology (SMR). Journal of Distance Education (05), 101-112. doi:10.15881/j.cnki.cn33-1304/g4.2020.05.011.

Wan Kun. (2022). Connotative Characteristics and Design and Implementation of Interdisciplinary Learning--Taking Information Technology Curriculum as an Example. Journal of Tianjin Normal University (Basic Education Edition) (05), 59-64. doi:10.16826/j.cnki.1009-7228.2022.05.011.

Wang You-Mei, Wan Ping, Nan Xi-Xuan Liu Chen-Chen. (2021). Entity programming for computational thinking development: tools and strategies. China Electrochemical Education(08),92-98.

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

Analysis of Problem-Oriented Computational Thinking Evaluation and Development Status Among High School Students

Mingqi CHEN[1], Shuang HAN[2*], Jihong XU[3]
[1,2,3]Inner Mongolia Normal University, China
1776033676@qq.com, 1362117995@qq.com, 115202177@qq.com

ABSTRACT
This study measured the level of development of computational thinking in 130 students through a computational thinking assessment tool developed by the research team. The results show that the five dimensions of computational thinking -- modularity, formalization, modeling, automation and systematization -- are significantly positively correlated. Currently, high school students' modular thinking level is higher, and systematic thinking level is lower. Teachers should guide students to lay a good foundation of modular thinking, strengthen the training of systematic thinking, synthesize multi-dimensional thinking training and provide personalized guidance and incentives. This paper aims to grasp the development status of high school students' computational thinking, explore the reasons behind it and provide targeted and operational guidance for teaching practice.

KEYWORDS
computational thinking, evaluation of computational thinking, development status, problem solving

1. INTRODUCTION
Computational thinking refers to a series of thinking activities generated by an individual in the process of forming a solution to a problem by applying the thought methods in the field of computer science (Ministry of Education of the People's Republic of China, 2020). For high school students, the development of computational thinking can not only make their way of thinking more logical, enhance their ability to solve problems, but also prepare them for future college study and real life. Evaluation of computational thinking has always been an important issue in the training of computational thinking. Reasonable evaluation of computational thinking can help teachers understand the real level of students' development of computational thinking, so as to make targeted improvements in teaching practice and enhance students' computational thinking and problem-solving ability.

At present, scholars have carried out some researches on the ways, methods and strategies of computational thinking evaluation in high school students. Bai and Gu developed an evaluation tool suitable for measuring the computational thinking ability of Chinese middle school students in five dimensions, including creativity, algorithmic thinking, cooperative ability, critical thinking and problem solving (Bai & Gu, 2019). Hui, Lan and Qian found that evaluation tools of computational thinking can be divided into four types: based on test questions, based on scales, based on programming tasks and based on system environment (Hui, Lan, & Qian, 2020). Li and Xie built a computational thinking evaluation framework based on the SOLO theory, and divided students' computational thinking level into five levels (Li, & Xie, 2023). The above evaluation tools or frameworks have different orientations, or they are divided into dimensions from the perspective of conceptual interpretation of computational thinking, or they focus on evaluation through student performance. However, the research team of the author has developed a computational thinking evaluation tool for high school students based on curriculum standards, which focuses on practicability and problem solving. Through this tool, the paper analyzes the current situation of students' computational thinking and puts forward some suggestions. The study aims to construct a scientific evaluation framework, develop effective evaluation tools, enrich the theoretical system of computational thinking assessment and guide teaching practices.

2. RESEARCH METHODS AND TOOLS

2.1. Questionnaire Survey
In this study, the research team developed a computational thinking evaluation tool for high school students based on curriculum standards. First of all, based on the existing research and combining with the characteristics of information technology in high school, the research team constructed a computational thinking evaluation model suitable for high school students. Then, combined with the curriculum standards, the use of questionnaire survey to collect expert opinions to modify and improve the indicators, the use of content analysis method to allocate the weight of indicators, forming an evaluation index system. Finally, the cognitive characteristics and current situation of Chinese high school students are analyzed. The test questions are designed and trial tested, and the evaluation tools of computational thinking are modified and improved based on the test results. The design of the evaluation tool drew upon classic example problems from high school mathematics, classic cases from information technology courses, and computational thinking assessment tasks from Bebras Tasks. The questionnaire contains a total of 15 questions, which are divided into five dimensions according to the index system: modularization, formalization, modeling, automation and systematization. There are 3 test questions in each dimension, among which the difficulty of the questions is 1 question each of high difficulty, medium difficulty and low difficulty. The questions are objective, and each choice is assigned a score from 1 to 4 based on the student's computational thinking level.

2.2. Trial Test
The research team selected two classes of freshman of a senior high school in H City (52 students in total) to test the

Research on Computational Thinking Assessment Tools Oriented Toward Core Competencies (S20231081Z)

evaluation tool. The evaluation tool of the test consisted of 15 questions, which were set to be submitted within 40 minutes, and 52 questionnaires were finally collected. The test results show that most of the students completed and submitted the test questions within the prescribed time, indicating that the setting of the number of questions is relatively scientific. The research team identified the high and low groups by 27% of the total number of people, calculated the difficulty coefficient and discrimination of each question according to the average score of the high and low groups, and finally replaced the four questions with low discrimination. Due to the good teaching staff, environment, and the students' high level of receptiveness to learning at this high school, they attach a high degree of importance to the subject of information technology, and the development of computational thinking is closely related to the subject knowledge of information technology in grade 1 of high school, so the test results have a certain accuracy.

2.3. Reliability

The commonly used reliability test method is Cronbach's Alpha(α) coefficient. It is generally considered that α>0.90 is the best, 0.80 is very good, 0.70 is moderate, and 0.60 is the minimum acceptable range. The results are shown in Table 1. The α coefficient of the computational thinking evaluation tool in this study is 0.709, so the internal consistency among the items of the tool is acceptable.

Table 1. Reliability.

Cronbach's Alpha	Item
.709	15

2.4. Validity

Validity analysis is to test whether the questionnaire questions are consistent with the research purpose. As shown in Table 2, KMO=0.739>0.6 and P <0.05, indicating that this data is suitable for factor analysis.

Table 2. Validity.

KMO		.739
Bartlett Test of Sphericity	Approximate Chi-square	240.912
	df	105
	P	.000

3. RESULTS

3.1. Basic Respondent Information

The research team sent questionnaires to 5 high schools, including 4 high schools in H city and 1 high school in S city. Each school selected 20-30 students, a total of 130 people, and the survey objects were all senior grade one students. Among them, there were 73 boys, accounting for 56.2%, and 57 girls, accounting for 43.8%. The students are mainly distributed in 4 regions: 78 in region A, accounting for 60%; 44 in region B, accounting for 33.9%; 5 in region C, accounting for 3.8%; and 3 in region D, accounting for 2.3%.

3.2. Correlation Analysis

As shown in Table 3, the correlation analysis results among variables show that there are all significant positive correlations between each pair of the five dimensions of the evaluation tool – modularity, formalization, modeling, automation and systematization.

Table 3. Correlation Analysis among The Dimensions of Computational Thinking in High School Students

Dimension	1	2	3	4	5
Modularity					
Formalization	.236**				
Modeling	.395**	.298**			
Automation	.403**	.480**	.443**		
Systematization	.239**	.389**	.288**	.374**	

Note. ***p*<0.01; 1: modularization; 2: formalization; 3: modeling; 4: automation; 5: systematization.

3.3. Difference Analysis

As shown in Table 4, the F value between the groups was 8.970, and the significance was <0.01, indicating that high school students' computational thinking level had significant differences in different dimensions. The LSD (minimum significant difference) method was used to compare the difference between the mean values of different dimensions of computational thinking of high school students. It was found that the difference between the formalization, modeling and systematization levels of high school students was not significant. The modularization level was significantly higher than the three, while the systematization level was significantly lower than the three.

Table 4. Difference Analysis among the Dimensions of Computational Thinking in High School Students (M±SD).

Dimension	Score	F	LSD
Modularity	3.06±0.64		
Formalization	2.86±0.65		
Modeling	2.89±0.65	8.970**	5<2,3,4<1
Automation	3.01±0.66		
Systematization	2.89±0.70		

Note. ***p*<0.01; 1: modularization; 2: formalization; 3: modeling; 4: automation; 5: systematization.

4. CONCLUSION

4.1. Different Dimensions of Computational Thinking Mutually Support Each Other

The five dimensions of computational thinking are not completely separated, but are interrelated and mutually reinforcing, and belong to the entire problem-solving process. Among them, modularity reflects the learner's understanding of the problem, and understanding the problem is the starting point of problem solving. Modularity can be regarded as the premise and basis of solving problems using computational thinking. Formal description of problems is the basis of algorithms (Ministry of Education of the People's Republic of China, 2020). The formalization ability lays the foundation for establishing accurate calculation models, selecting algorithms and using computing tools to process calculation results, making subsequent modeling and automation possible. In addition, after learners establish models and select algorithms, they can apply intelligent tools such as computers to process them. In turn, computers can also optimize and iterate models in the process of automatic problem processing. Therefore, learners' modeling level is significantly positively correlated with their automation level. Finally, systematization reflects learners' ability to summarize and transfer after solving problems, the improvement of high school students' level of other dimensions of computational thinking can promote their final systematization ability. On the whole, the five dimensions of computational thinking are in a sequential relationship, reflecting the steps of solving problems with computational thinking. The former dimension is the basis of the latter dimension, and some of the latter dimension can also react on the former dimension. Therefore, each dimension is mutually complementary and interdependent.

4.2. High School Students Lack in Systematic Thinking

Compared with other dimensions of computational thinking such as modularity, the systematization level of high school students is low, which reflects the lack of summary and transfer ability of high school students. Huang et al. believe that systematic thinking is a thinking method that regards cognitive objects as a system and comprehensively thinks about and understands cognitive objects by comprehensively considering the relationship and interaction between internal elements of the system and the interaction between the system and the external environment (Huang, Yang, Wang, Huang & Yang, 2014). In teaching practice, especially in the case of exam-oriented education, students tend to focus on whether a specific topic is correct or not, but do not pay attention to the summary, reflection and transfer after solving the problem, and can not draw inferential examples. From the perspective of teachers, some teachers study too much the teaching methods of specific knowledge points, but neglect to guide students to build the knowledge system of the whole subject. At the end of the class, there is often a lack of summary and sorting, which leads to the low level of students' systemization.

5. SUGGESTIONS

5.1. Lay a Good Foundation of Modular Thinking

High school students perform better in modularity. Educators should continue to carry forward this advantage, and modularity can be used as the basis for cultivating other dimensions of thinking. In terms of teaching content, teachers can gradually guide students to expand their thinking scope through modular training, and extend this ability to decompose problems to a wider range of fields. Learning programming languages is an effective way to cultivate modular thinking, which involves decomposing problems into modular code segments and helping students understand how to decompose complex problems into manageable parts.

5.2. Strengthen the Training of Systematic Thinking

The level of systematic thinking in high school students' computational thinking is relatively low, so we should strengthen the training of systematic thinking in high school students. Cultivating systematic thinking can be achieved by solving intricate problems and exploring concepts with strong correlations, as well as connections between different disciplines. Teachers can introduce the teaching mode of project-based learning to drive students to cooperate, communicate and explore actively in real problems, and improve their comprehensive problem-solving ability. At the same time, it can guide students to conduct interdisciplinary learning, help students break knowledge barriers, better understand knowledge in different fields, understand the links between knowledge and the overall framework, and cultivate comprehensive thinking.

5.3. Synthesize Multi-dimensional Thinking Training

It is suggested that teachers should adopt a multi-dimensional combination method when designing tasks or projects, and cover the application of multiple dimensions of computational thinking at the same time, so that students can comprehensively use various modes of thinking to solve problems, and avoid the situation that the deficiency of one ability will cause the level of other dimensions to decline. In addition to the improvement of teaching, we should also pay attention to the training of teachers themselves and enhance the level of professional development of teachers. Primary and secondary school teachers can cooperate with college teachers to develop teaching activities of computational thinking, learn teaching methods of computational thinking, and improve teachers' computational thinking ability and teaching level.

6. REFERENCES

Bai, X. M. & Gu, X. Q. (2019). Construction and application of computational thinking evaluation tools for K12 students. *China Educational Technology, 2019*(10), 83-90.

Han, S. (2023). *Research on computational thinking evaluation of high school students based on information technology curriculum standards*. Master Dissertation, Inner Mongolia Normal University. doi: 10.27230/d.cnki.gnmsu.2023.000099.

Huang, L. L., Yang, K., Wang, Y., Huang, X. Y., & Yang, X. H. Analysis of computational thinking characteristics. *Computer Education, 2014*(17), 12-15. doi: 10.16512/j.cnki.jsjjy.2014.17.027.

Hui, G. J., Lan, X. F., & Qian, Y. Z. (2020). How to evaluate computational thinking? - a comparative analysis of 14 evaluation tools at Home and Abroad. *Journal of Distance Education*, *38*(04), 84-94. doi: 10.15881/j.cnki.cn33-1304/g4.2020.04.009.

Li, Y. & Xie, Z. X. (2023). Research on computational thinking evaluation based on SOLO classification theory - taking high school information technology as an example. *Basic Education Curriculum, 2023*(23), 60-66.

Ministry of Education of the People's Republic of China. (2020). *Curriculum Standard for Information Technology in Ordinary High Schools (2017 Version, Revised in 2020)*. Beijing: People's Education Press, 2020.

A Curriculum Design Paradigm for Computational Thinking: Informed by Big Ideas and KUD Learning Goals

Dongshuang ZHANG[1*], Yong Mao[2], Zhengrong CHEN[3], Yuanyuan ZHANG[4]
[1,2,3,4] ITCCC, China
zhangdongshuang@itccc.org.cn, maoyong@itccc.org.cn, chenzhengrong@itccc.org.cn

ABSTRACT
The newly revised Information Technology Curriculum Standard for Compulsory Education (2022 Edition) by the Ministry of Education emphasizes the significance of core literacy courses, positioning them as foundational components of the discipline. Computational thinking emerges as a key skill among the four proposed core literacies, crucial for tackling complex challenges in today's rapidly evolving digital landscape. This paper proposes a curriculum design paradigm centered on big ideas and KUD learning objectives, aiming to deepen students' understanding of IT concepts and enhance their problem-solving abilities. By integrating big ideas into the curriculum, teachers can plan more effectively, aligning with students' core literacy development. The design encourages real-world problem-solving, fostering analytical and algorithmic skills for practical application.

KEYWORDS
Big ideas, KUD learning goals, Computational thinking, core literacy

1. INTRODUCTION
The rapid pace of technological advancement has integrated transformative technologies such as Artificial Intelligence, the Internet of Things, and Big Data into our daily lives. The 2022 revision of the Information Technology Curriculum Standard for Compulsory Education by the Ministry of Education highlights the critical need to cultivate core competencies in students, encompassing information literacy, computational thinking, digital innovation, and social responsibility in the digital age. Computational thinking, recognized as a vital cognitive process for problem-solving and system design (Wing, 2006), is universally valued by educators but presents challenges in effective curriculum design and implementation (Yadav, 2014; Kalelioglu, 2016; Heintz, 2016). Traditional education models often focus on transferring knowledge and assessing student understanding through exams, neglecting the development of innovation and practical problem-solving skills (Smith, 2005). The misconception that programming education is synonymous with computational thinking limits the scope of the latter, which should transcend specific technologies and encompass broader problem abstraction, systematic problem-solving, and solution dissemination.

This study aims to propose a curriculum design paradigm for computational thinking, grounded in big ideas and KUD (Know, Understand, Do) learning objectives, offering a theoretical and practical framework to navigate these complexities. By engaging with real-world issues, students are expected to develop a deeper, foundational understanding of computational thinking and information technology, enhancing their skills and abilities.

2. CORE CONCEPTS OF BIG IDEAS AND KUD LEARNING GOALS

2.1. Big Ideas
Big ideas are the essential and unifying concepts that define a discipline, forming the basis for understanding its scope and depth. In the context of computational thinking education, these concepts integrate a range of ideas, providing students with a comprehensive view of the field and deepening their grasp of fundamental principles. For instance, the concept of "algorithm" is a big idea that bridges simple sorting processes to complex machine learning algorithms, allowing students to recognize the shared characteristics and broad applicability of various algorithms (Wiggins & McTighe, 2005).

2.2. KUD Learning Objectives
The KUD framework—standing for "Know, Understand, Do"—offers a structured approach to educational planning, recognized for its precision in guiding teachers to plan course content, instructional strategies, and assessment methods more effectively. This model categorizes learning objectives into three distinct types: 'Know' objectives involve acquiring essential facts and concepts; 'Understand' objectives develop higher-order thinking skills such as analysis, synthesis, and evaluation; and 'Do' objectives emphasize the application of knowledge in various contexts, fostering problem-solving and decision-making abilities. By integrating the KUD model into curriculum design, educators can create learning experiences that promote deep understanding and the ability to apply learning in meaningful ways (Wiggins & McTighe, 2005).

3. MODEL OF CURRICULUM DESIGN INFORMED BY BIG IDEAS AND KUD LEARNING GOALS
The following sections detail the essential components of a curriculum design model grounded in big ideas and KUD learning objectives. The discussion will cover the formulation of engaging questions, the construction of knowledge maps, the design of challenge tasks, the development of problem-solving strategies, and the multidimensional assessment and visualization of learning outcomes. These elements collectively form a comprehensive instructional strategy aimed at fostering deep understanding and application skills in students.

3.1. Identifying Core Questions
Course design initiates with defining open-ended "big questions" that integrate key concepts, prompting students to draw from diverse knowledge and engage in thorough inquiry. These questions are designed to spark curiosity and steer students towards an integrated learning journey with an emphasis on critical thinking.

3.2. Building Knowledge Maps

Big ideas form the core of instruction, connecting key concepts to KUD objectives through knowledge maps that structure teaching. These maps direct teaching plans, focusing learning segments on essential concepts.

3.3. Designing Challenge Tasks

Challenge tasks, aligned with KUD objectives, facilitate students' deeper understanding and application of key concepts through varied assessments, enhancing comprehension and skill mastery.

3.4. Developing Problem-Solving Maps

Problem-solving maps, based on KUD objectives, provide a structured approach to learning, helping students master problem-solving strategies and achieve the progression from knowledge to understanding to practice.

3.5. Designing Learning Activities and Resources

Instructional design revolves around KUD objectives, with all learning activities and resources aimed at supporting student achievement of these goals, thus promoting a deep understanding of key concepts.

3.6. Assessing and Visualizing Learning Outcomes

Multidimensional assessment and visualization of student learning outcomes, such as project presentations and professional challenges, not only motivate students but also provide feedback on teaching effectiveness for instructors.

3.7. Setting Assessment Criteria

Assessment criteria should measure the degree of achievement of KUD objectives and reflect how students understand and apply key concepts. The assessment process is closely integrated with learning, with practice and feedback enhancing learning effectiveness.

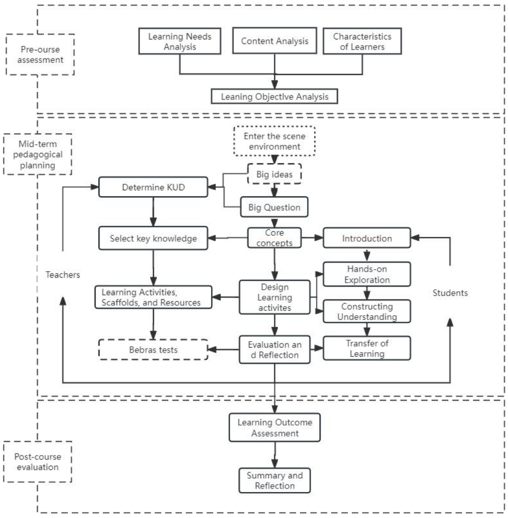


Figure 1. The Curriculum Design Model Based on Big Ideas and KUD Learning Goals

4. COMPUTATIONAL THINKING CURRICULUM DESIGN INFORMED BY BIG IDEAS AND KUD LEARNING GOALS

4.1. Integration of Big Ideas of Computational Thinking

Computational thinking is a series of complex thinking processes during problem-solving, such as problem decomposition, algorithm design, problem-solving, pattern recognition, abstract thinking, etc. In curriculum design, integrating big ideas of computational thinking, such as problem decomposition and algorithm design, into a tangible teaching case, facilitates students' comprehensive understanding of computational thinking.

For instance, in the "My Campus Map" project, students will explore key concepts of computational thinking through unplugged activities. For example, students can work in groups to create a paper map of the campus, marking multiple routes from the classroom to the playground. Through this activity, students learn how to decompose problems (problem decomposition), find the best route (algorithm design), and recognize which routes are more efficient under certain conditions (pattern recognition).

4.2. Establishing KUD Learning Goals

KUD learning objectives for the "My Campus Map" project should be crafted to facilitate a gradual learning process that starts with acquiring basic knowledge, moves on to deepening understanding, and culminates in applying this knowledge in a practical context. These objectives are tailored to the project's big idea of navigating and mapping a familiar environment, making them accessible and engaging for young learners.

Know: Students will acquire fundamental knowledge about map symbols, directions, and the layout of their school campus. They will learn to identify key locations and represent them on a map.

Understand: Students will gain an understanding of how different paths can be chosen based on various criteria, such as the shortest route, the most scenic route, or the safest path for different weather conditions. They will discuss the rationale behind choosing one path over another.

Do: Students will apply their knowledge by creating a detailed map of their school, selecting the best routes for different scenarios, and explaining their choices. They might also design a simple game or activity that involves navigating the campus using their maps.

This approach ensures that the KUD objectives are aligned with the project's educational goals and are age-appropriate, encouraging students to actively engage with the concepts of computational thinking through hands-on, unplugged activities.

5. CONCLUSION AND PROSPECTS

The study has comprehensively examined a computational thinking curriculum design paradigm grounded in big ideas and KUD learning objectives. The results suggest that this approach effectively integrates and deepens students' learning experiences, offering a practical framework for curriculum planning and assessment. This design pattern allows for the intentional construction and execution of curricula, with a focus on enhancing core literacies and stimulating innovative thinking and curiosity. By applying knowledge and skills to real-world challenges, students develop a nuanced understanding and problem-solving abilities, building confidence and autonomous learning capacity. This curriculum design is well-aligned with the evolving demands of the digital age, fostering students' preparedness for active and responsible participation in the information society. The research advocates for broader engagement with computational thinking curriculum design, aiming to validate and expand its application in diverse educational settings, and envisions a new educational era that enables each student to fully realize their potential and intellect.

6. REFERENCES

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

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2014). Introducing computational thinking in education courses. *Proceedings of SIGCSE'14*, Atlanta, GA, USA: ACM.

Kalelioglu, F., Gulvahar, Y., & Kukui, V. (2016) A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic J. Modern Computing,* 4(3), 583-596

Heintz, F., Mannila, L., & Farnqvist, T. (2016). A Review of Models for Introducing Computational Thinking Computer Science and Computing in K12 Education[C]. Frontiers in Education Conference.

Smith, P. L., & Ragan, T. J. (2005). A framework for instructional strategy design. In P. L. Smith & T. J. Ragan (Eds.), *Instructional design* , 3rd ed., 127-150. New York, NY: Wiley & Sons.

Wiggins, G., & McTighe, J. (2005). *Understanding by Design* (2nd ed.). Alexandria, VA: Association for Supervision and Curriculum Development.

Hubwieser, P., & Muhling, A. (2015). Investigating the Psychometric Structure of Bebras Contest: Towards Measuring Computational Thinking Skills. In 2015 International Conference on Learning and Teaching in Computing and Engineering.

How Preservice Teachers using Computational Thinking to Solve Problem:Based on a LSA Research

SU Qing[1,2*] ,YU Rong[3]
[1]School of Educational Technology, [2]Key Laboratory of Education Digitalization of Gansu Province, North West Normal University, Lanzhou, Gansu, China
[3]Mental Health Education and Counseling Center, Lanzhou City University, Lanzhou, Gansu, China
suqing8909@163.com, 48352811@qq.com

ABSTRACT
The research revealed significant behavioral sequences in the application of computational practices to problem solving through LSA of course high-performers.The research found that the high-performers' behavior sequence has the following three characteristics:1)Students' using computational practices to solve problems can be divided into three stages;2)Significant behavioral sequences exist between the three stages of computational practice;3)Various interactive behaviors aid computational practice for high-performers.Based on this,the research gave three recommendations:1)Designing curriculum content that accurately understands the meaning of computational thinking;2)Designing project-based learning activities based on the process of computational practice;3)Providing rich scaffolds for students' computational practice.

KEYWORDS
Computational Practice;LSA;High-Performers;Behavioral Sequences

1. INTRODUCTION
With the advent of the information age, computational thinking is regarded as an increasingly essential skill for learner in the twenty-first century(Grover & Pea, 2013; Sun et al., 2021) .Wing defined computational thinking as "taking an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing"(Wing,2008) .Then, she defined it as "the thought processes 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 2014).Computational thinking has become increasingly important for all generations as digital assets and computation are increasingly embedded into the processes and tools we use as a society, at work, and in everyday activities.

Recent years, it is indicative that a large number of studies focusing on computational thinking have been published(T.C. Hsu, Chang, & Hung, 2018). Computational thinking's large body of literature including(a)assessment methods and frameworks that encompass the complexity of computational thinking,(b)approaches that align learning strategies with computational thinking and (c)knowledge needed to teach computational thinking and methods which support to teachers(Christina, T & Efthimios,T,2021).

Project based learning engaging students into authentic project around real challenges and problems, encouraging students using computational thinking process to solves the challenges and problems, has been proved as one of the most useful strategies for computational thinking learning.

As an important research area, researchers in the examined studies develop and validate assessment methods, propose framework or measure students' computational thinking in order to achieve deep understanding of students' learning(Fronza et al., 2017) by different assessment methods. According to existing studies, assessment area can be divided into five sub-area:Self-report methods; Text or Scale; Artifact Analysis; Observations; and Frameworks. By using observations of students' actions, screen recording, learning analytic, camera recordings, researchers' notes, structure-based observations, studies gain a complete picture of students' understanding((Da Cruz Alves et al.,2019). It should be noted that, as a process of solving problem, few studies focus on how people using computational thinking to solve problem so called computational thinking behavior pattern in problem-solving.

In this paper, we use project based learning as the strategy and reveal students' behavior pattern in problem solving by using computational thinking.

2. LITERATURE REVIEW
Lag Sequential Analysis(LSA) which was introduced by American scholar Gene P.(Sackett,G.P.,1978) in 1978 for testing the probability of people engeging in a particular behaviour followed by another and their significance(Bakemen,R.,1997).Recent years, LSA has been increasingly noticed by educational researches and used in the study of learning behaviour analysis.

With the growing interest in programming education, LSA has been widely applied in relevant research.Starting from learning analytic, Wu Linjing et al. explored the difference in behaviour patterns during programming among different types of learners by analyzing the learners' behaviors in programming process by coding as well as LSA, and proposed targeted suggestions for improving the efficiency of programming learning(Wu Linjing et al.,2020). Akrolu N et al. used LSA to analyse the programming behaviors of 15 sixth-grade primary school students in a graphical programming environment and developed a pattern of focusing, gaming, and determining behaviors in programming problem solving(Akrolu,N. & Mumcu,S.,2020).Sevda K et al. Analyzed the sequence of interactive behaviours of 18 students aged 8 to 11 years and their teachers in one-to-one robotics sessions by using lagged behaviours, finding that students assembled blocks, shared ideas, and teachers provided guidance and asked questions were the most frequent behaviors, and discussed the teacher-student interactions in details(Sevda,K. & Burak,S.,2017).

The aim of this study is to use LSA to reveal the behaviour patterns of the course learning achievers applying computational thinking to problem solving.

3. METHOD

3.1. Context

This study was conducted in an educational technology course offered for junior students in an public, normal university in west China. The course which named Intelligent Innovation and Maker Education in Primary and Secondary Schools which under the guidance of project based learning theory by using open-source hardware. During the course, students are divided into groups randomly and each group should finish 9 projects.

3.2. Participants and Research Question

Twenty-five junior educational technology students, including 7 males and 18 females, aged between 20-22 years old participated in the course.During the assessment period of the course, students were randomly divided into five groups to carry out a three-week, six-hour project production based on the theme of "Smart Dormitory". Upon completion of the assessment project, the groups were required to present their project works and submit design proposals, programme codes and introduction videos. Five professional instructors assessed the projects in five dimensions, including innovation, technicality, artistic quality, specifications, as well as team presentation and collaboration, through the report and the review of the materials.

This research focuses on the sequence of behaviors of high-performing groups after project assessment.

RQ1： Does the group develop a significant behaviour sequence while completing the project?

RQ2：If so, what particular behaviors are included in the behaviour sequence?

3.3. Data Collection

Brennan K et al. define computational thinking as three dimensions: computational concepts, computational practices, and computational perspectives(Brennan K & Resnick M,2012). Computing practice includes the range of behaviours in problem solving, and project development.Pinkard N et al. proposed an operational definition of computational practice, defining it as three components: decomposition and abstraction, collaborative coding, and testing and debugging(Pinkard N et al.,2019).Research adopted Pinkard N et al.'s findings and incorporated the practicalities of open source hardware project production to develop the computational practice coding schemes for this research.Meanwhile, research incorporated the learning behaviours of discussing in groups, referring to materials and questioning teachers into the coding scheme as well, finally forming the coding scheme of this study.

Table 1. Coding Scheme

Domains	Sub-categories	Details
Computa tional Pr actice	Decomposition and Abstraction (DA)	Decompose the complex problem into solvable sub-problems based on the project theme and choose suitable algorithmic models, equipment originals for description.
	Collaborative C oding(CC)	Using the programming platform to collaboratively code problem-solving programmes.
	Testing and Deb ugging(TD)	After programming completed and clicked "Upload Program",debugging the program to find out the problems of hardware and software.Solving them by modifying the program module, adjusting the program parameters and replacing the equipment components.
Learning Behavio ur	Discussing in G roups（DG)	Group discussion centred on project completion.
	Referring to Ma terials(RM)	During the completion process, refer to related materials through the Internet or review information about programs that have been written in previous lessons.
	Questioning Te achers(QT)	Questioning teachers and seeking help when problems arise.

The study recorded the entire process of all 5 groups participating in the course final assessment, and recorded the screen on the computers they used to assist in the analysis of the relevant behaviors.Following the final assessment, video files of the top scoring performance group were chosen to be analyzed.The video files were over 6 hours, and the research took 3 seconds as the sampling time and coded the student behaviour , resulting in approximately 6,300 behaviour codes.The coding was carried out independently by the two coders and the resulting coding results were tested for consistency with the coefficient of 0.887, indicating the consistency between the two coders.The research used GSEQ software to process and analyse the behaviour data to form the frequency tables of the behaviour transitions and the adjusted residual table.Based on the LSA theory, if the residual table has a Z-value greater than 1.96 then it indicates that the behavioral path is significant(Bakeman R,1997).Research mapped the behavioral sequence transitions of high-performers according to the behavioral sequences with Z-values.

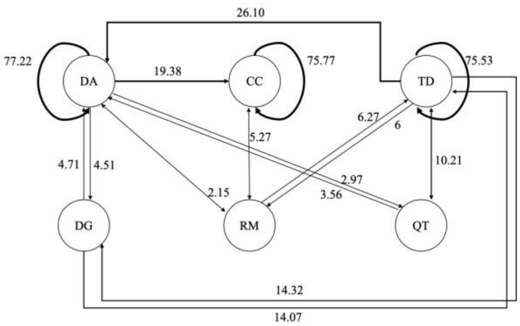


Fig.1. The behavioral transition diagram of high-performing groups

4. FINDS AND DICUSSION

4.1. Students' using computational practices to solve problems can be divided into three stages

The first stage named decomposition and abstraction(DA→DA).During this stage,students begin with the theme of the assessment project, focusing on the key issues of the project as well as decomposing and abstracting them into specific problems that they can solve.Based on their own experience, students focused on the project theme, identified the problems that needed to be solved, and proposed a variety of solutions to the problems. Subsequently, students identified and abstracted the best solutions for these problems in terms of problem-solving principles and the practicalities of open-source hardware and equipment, forming specific solutions that could be achieved by using open-source hardware as well as algorithms and programs.The second stage named collaborative coding(CC → CC).In this stage,students design algorithms and solve them one by one by coding programs mainly for the specific issues from the previous stage.The third stage named testing and debugging(TD→TD).Students debugged and tested the programme in terms of smoothness of operation, stability and suitability for the problem solutions. Based on the analysis of the computer recording files, it was found that students debugged the written programs from the dimensions of adding, subtracting and replacing program modules, adjusting the program modules' parameters, and testing and replacing the hardware devices.

4.2. Significant behavioral sequences exist between the three stages of computational practice

The behavioral sequences DA→CC shows that for high-performers, students' starting point for achieving the solution of the projects is to decompose and abstract the problem, based on which they enter the coding phase to achieve the solution of the problem.

For the designers of problem solving, testing and dealing with the problems which arise is a vital behaviour in computational practice.Research found that students significantly transferred to the decomposition and abstraction stage after the testing and debugging stage, as well as significantly transferred to the coding stage through the decomposition and abstraction stage.This significant behavioral sequence indicates that after testing the project problem, high-performers did not hastily adjust the program, but returned more to the decomposition and abstraction of the problem, to analyse the root cause of why the program errors occurred, and then to correct the errors by coding the program to achieve better solution for the problem.

4.3. Various interactive behavior aid computational practice for high-performers

Research found that students produced numerous significant interactive behavioral sequences besides behaviors associated with computational practice.During the decomposition and abstraction stage, the significant behavioral sequences were DA↔DG,DA↔QT,DA↔RM.

Research found that the students did not work alone in decomposing and abstracting the problems, and showed significant behavioral sequences of teamwork as well as discussion.In the decomposition and abstraction of the problem, students first brainstorm the set of proposed problems for the project based on the theme of the project.Having formed the set of problems, students first discussed around the reasonableness of the problems, and the team worked together to analyses whether the problems formed were appropriate to the project context and authentic experience.Next, through team discussion, students describe the problem solution in their own words, abstract the problem, select equipment and procedures, and set the foundation for coding programs to implement automatic problem solving.Besides group discussions, the behavioral sequences of DA interacting with QT were equally significant.In the decomposition and abstraction of the problem, students question their teachers to gain answers to the identified key issues of the project, the rationality and availability of the chosen hardware equipment.In terms of the significant interactive behavioral sequence of DA and RM, students search the internet and books to determine whether the selected hardware equipment and the programs can solve the problem.In summary, the three significant behavioral sequences of DA and GD, QT and RM can indicate that students' interactions with peers, teachers, and materials make the abstract decomposed project problem more clear, and lay a solid foundation for the stage of coding.

During the coding stage, students' significant learning behavioral sequences were the interaction between CC and RM.Students need to select appropriate sensors and actuators according to the problem that the project intends to solve, and implement the solution of the problem through connections and programming.During this process, students need to both apply relevant knowledge learnt in previous courses and choose new equipment or develop new features to meet new requirements.Thus, the interaction between students and the search for materials in coding is mainly expressed by viewing the information about the equipment and procedures that they have learnt and applying them to the project problem solving.Students also need to select new sensors and actuators to achieve new functions when facing new problems in new projects, which require students to keep searching for materials.Research found that students' search for information often came from two sources. The first is a "library" of previously done projects and tasks that students can use when they need to complete a similar problem. The other is through the Internet, especially forums, to find information on new problems and new equipment, and to do it by imitation.

During the testing and debugging stage,the significant behavioral sequences were TD↔DG,TD↔RM,TD↔QT.From the behavioral sequences, the research found that students continuously identified problems in the solution of the project and explored modifications to the solution through discussions with the teacher, peers, and the community formed by the materials.

5. RECOMMENDATION

5.1. Designing curriculum content that accurately understands the meaning of computational thinking

As we all know, computational thinking requires students can combine their own life, learning practice, using computer-related knowledge to solve problems, complete projects and draw conclusions.Therefore, in the cultivation of computational thinking, it is necessary to focus on its problem-solving thinking process as a major line, to drive the curriculum with problems, and to cultivate students' awareness of problems and their ability to solve problems using computational thinking.Focusing on the role of real-life problems around students in driving their learning, enabling students to try out different approaches to solving problems so as to gain relevant knowledge, enhance their computational thinking and improve their problem-solving skills.

5.2. Designing project-based learning activities based on the process of computational practice

As a practical process of problem solving using computer-related knowledge or methods, the research found that the behavioral sequences of computing practices are not random.We should orientate students' behavioral sequences through the design of learning activities and help them to learn along the pathway of computational practice in problem solving, so as to achieve the development of their computational thinking.

5.3. Providing rich scaffolds for students' computational practice

During the three stages of students' computational practice, the research found that all of them required teachers, peers, and resources as scaffolds to support students in the appropriate behaviors.Meanwhile, students often use the problem solutions they have already completed in previous courses as scaffolds for solving new problems.For this reason, the research recommends that in the practice of developing computational thinking, it is important to pay attention to the role of traditional resource-based scaffolds as well as the provision of intellectual scaffolds through the formation of teacher-student learning communities. Furthermore, a mechanism needs to be established to guide students to keep the generated problem solutions in a proper and categorized manner, so as to continuously supplement the scaffold content.

6. ACKNOWLEDGEMENTS

This work was supported by Gansu Provincial Department of Science and Technology 2022 Basic Research Programme - Soft Science Special Project "Research on Unplugged Programming Activities to Cultivate Computational Thinking of Primary School Students in the Early Age Groups - Taking Gansu Province as an Example" (Project No.: 22JR4ZA052)

7. REFERENCES

Grover,S., & Pea, R. (2013).Computational thinking in K-12: A review of the state of the field. Educational Research, 42(1), 38-43.https://doi.org/10.3102/0013189X12463051

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical & Engineering Sciences, 366(1881), 3717 – 3725. https://doi.org/10.1098/rsta.2008.0118.

Wing, J. M. (2014). Computational thinking benefits society. Retrieved December 24, 2018, from http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.

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. https://doi.org/10.1016/j.compedu.2018.07.004

Christina, T., Efthimios,T.(2021).Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review.Computers & Education,162,1-23.https://doi.org/10.1016/j.compedu.2020.104083

Fronza, I., El Ioini, N., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. ACM Transactions on Computing Education, 17. https://doi.org/10.1145/3055258

Da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. R. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. Informatics in Education, 18, 17 – 39. https://doi.org/10.15388/infedu.2019.02

Sackett,G.P.(1978).Observing Behavior: Theory and applications in mental retardation (Vol. 1)[M]. Baltimore: University Park Press.

Bakeman R.(1997). Observing interaction: An introduction to sequential analysis[M]. Cambridge: Cambridge university press.

Wu Linjing et al.(2020).A study of learners' programming process analysis under learning analytics perspective.Modern Distance Education,(02),68-75.

Akrolu N, Mumcu S.(2020). Focus-Fight-Finalize (3F): Problem-Solving Steps Extracted From Behavioral Patterns in Block Based Programming:[J]. Journal of Educational Computing Research, 58(7):1279-1310.

Sevda K, Burak S.(2017). Behavioral patterns of elementary students and teachers in one-to-one robotics instruction. Computers & Education,111(8):31-43.

Brennan K, Resnick M.(2012).New frameworks for studying and assessing the development of computational thinking [C] AERA:1-25.

Pinkard N, Martin CK, Erete S.(2019).Equitable approaches: opportunities for computational thinking with emphasis on creative production and connections to community[J]. Interactive Learning Environments,(6):1-15.

Theoretical value, dynamic mechanism and implementation path of the "new three-dimensional goal" for computational thinking

Jin XIN[1], Hongshui SHAN[2], Dilibaha SAFUTIN[3] , Zhenyue SUN[4]
[1,2,3]AuthorsXinjiang Uygur Autonomous Region Electrified Education Center, China
[4]Huanggang Normal University, China
1stAuthor527614240@qq.com, 2ndAuthor332782997@qq.com, 3rdAuthordilbahar528@126.com, 4thAuthor897354667@qq.com

ABSTRACT

In the era of intelligence, educational reform faces the challenge of world uncertainty, and the cultivation of core literacy is the most important issue. This article explores the theoretical value of the "new three-dimensional goal" of knowledge, problem-solving, and value, studies how to make computational thinking a key driving force for transitions between different levels, and attempts to provide direction for the practical implementation of core literacy cultivation through path description and case analysis.

KEYWORDS

1. INTRODUCTION

Driven by globalization and technological innovation, China has actively promoted educational innovation to cultivate talents in the new era. The "Three-dimensional Goals" cover three dimensions: knowledge and skills, process and method, emotional attitude and values. Although it promotes the transformation of basic education, it focuses too much on exam-oriented learning and ignores practical operation and innovative thinking. Therefore, China has entered the era of "core literacy" since 2014, emphasizing the cultivation of essential character and ability to adapt to lifelong development and social needs. Compared with the "Three-dimensional Goals", core literacy pays more attention to knowledge application, multi-faceted ability cultivation and social adaptability, and focuses on individual differences and innovation. From the "Three-dimensional Goals" to core literacy, it shows the exploration and progress of China's educational reform, and how to deepen the cultivation of core literacy is one of the most important issues at present.

2. THEORETICAL VALUE

The "new three-dimensional goal" is a brand-new solution to address the challenges of this topic. Rooted in the concept of educational democratization, it integrates humanism, understanding orientation, and practical approach, emphasizing problem-solving and moral cultivation in real-life situations. It encourages students to gain deep understanding and learning through practical experiences [1]. Drawing on modern theories such as multiple intelligences and constructivism, it demonstrates strong inclusiveness and adaptability. This goal system is a concise summary of the cultivation of students' core literacy, centrally reflecting the core pursuit of education and demonstrating profound insights into educational objectives. By providing a practical guidance framework for educators, it promotes student subjectivity, autonomous learning, and comprehensive development, enabling the true implementation of core literacy cultivation.

3. DYNAMIC MECHANISM

3.1. Explanation of ArchitectureSub-sections Guidelines

The "new three-dimensional goal" consists of the knowledge layer, problem layer, and value layer, which together support a comprehensive educational framework (as shown in Figure 1).

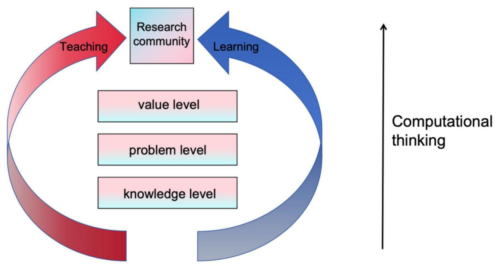


Figure 1: Diagram of the logical structure of the "New three-dimensional goal"

The knowledge layer serves as the foundation, focusing on the question of "what to learn" and emphasizing students' mastery and application of basic knowledge and skills. Through deep guidance by teachers, students engage in "progressive learning" and "immersive learning" of structured, logical, and systematic knowledge, building a solid knowledge structure.[2]

Core literacy can only be formed through continuous problem-solving processes [3], thus the problem layer serves as the core, focusing on the question of "how to learn". It aims to cultivate students' thinking and problem-solving abilities to adapt to changing social environments. Through the design of real-world problem scenarios, students are able to exercise their advanced and humanistic abilities while solving problems.[4]

The value layer, as the highest level, delves deeply into the question of "why to learn", guiding students to understand the meaning of learning and stimulating their intrinsic motivation. It emphasizes the role of core literacy in individuals' lifelong sustainable development, focusing on shaping students' moral character, values, and life perspectives. By implementing the goal of cultivating moral integrity, it becomes a highlight and symbol of educational work.

3.2.The dynamic mechanism of transitionxplanation of ArchitectureSub-sections Guidelines

Computational thinking, as a new way of thinking for problem-solving [5], serves as the critical driving force for the upward transition of each level within the "new three-dimensional goal". It achieves the transition from the knowledge layer to the problem layer by connecting cognitive materials, decomposing, and abstracting operations [6], deepening students' understanding of the essence of problems and laying the foundation for the cultivation of core literacy. It encourages students to challenge traditions, stimulate innovative consciousness and critical thinking, and achieve the transition from the problem layer to the value layer. At the same time, computational thinking emphasizes dynamically responding to challenges, helping students learn efficiently, adapting to the demands of the digital era, and preparing them for future life and careers.

4. IMPLEMENTATION PATH

4.1.Vertically - anchoring the educational value through the "focus effect"

The implementation of the educational goals ensures the coherence and staged achievement of literacy improvement [7], serving as the cornerstone of the teaching design for the "new three-dimensional goal". Given the limited cognitive resources of humans, teachers need to employ the "focus effect" strategy to select key teaching content, allowing students to access the most valuable learning resources within a limited time. The operational steps are as follows:

Enhancing Curriculum Thinking

Systematically construct and orderly advance teaching/learning objectives, including analyzing teaching materials, clarifying academic requirements, connecting with core literacy, and evaluating the evaluability of objectives based on SMART criteria [8].

4.1.2.Focus on the value of educating people

By meticulously decomposing and screening, teachers can focus the teaching on the content that best reflects the value of educating people, reducing students' cognitive pressure, promoting deep learning, and achieving vertical connectivity of the educational value of courses across different educational stages [9].

4.1.3.Clarify teaching objectives

Ensure that the objectives cover multiple dimensions including knowledge, methods, thinking, and literacy, addressing questions such as the purpose, content, and methods of teaching, while avoiding overly complicated objectives.

4.1.4.Drafting Learning Objectives

Integrate action methods, learning content, standards, and subject literacy from the perspective of thinking literacy, transforming teaching objectives into specific, measurable, and hierarchical learning objectives, and guiding students to clarify the meaning, content, and methods of learning.

When teaching the course "Password Security, I Will Guard It," the author reprocessed the fifth-grade information technology textbook published by Tsinghua University based on the "new three-dimensional goal," focusing on password security and aiming to cultivate logical thinking and innovative ability. Using Scratch as the carrier, the course integrated algorithms with students' daily lives, guiding them to understand the principles of decryption and encryption and achieving content upgrading. Through rich teaching activities, students gained knowledge and skills, core concepts, computational thinking, and other dimensions. The constructed "new three-dimensional goal" included: the knowledge layer - mastering enumeration algorithms and linked list applications; the problem layer - applying computational thinking to solve decryption and encryption problems; and the value layer - cultivating the habit of analyzing problems, experiencing the joy of learning, and establishing awareness of information security. This goal system provided clear navigation for students' deep learning, enabling them to master programming skills while deeply understanding the value of information security and achieving the overall goal of educating people.

4.2.Horizontally - Expanding Point-Chain-Network through "Cumulative Effect"

Within the framework of constructivism, the "cumulative effect" manifests in three dimensions: knowledge accumulation deepens understanding, the development of cognitive structure enhances adaptability, and social interaction promotes knowledge sharing and improves learning outcomes. Based on this, a clear expansion path for the point-chain-network can be formed.

4.2.1.The Chain of Questions Runs Through

Based on the "new three-dimensional goal" and in combination with students' experiences and confusions, teachers design a hierarchical and systematic chain of teaching questions [10]. These questions revolve around a central theme, form a sequence, and are interrelated, aiming to guide students to deeply understand the learning content and train their thinking in terms of rigor, divergence, criticism, and profundity.

In the lesson of "Cryptography," by designing core questions and tasks, a dual-line drive of problem chain and task chain is formed, guiding students to gradually explore the principles of decryption and encryption, deepen their understanding of "big concepts" such as algorithms, and enhance learning outcomes. (As shown in Figure 2)

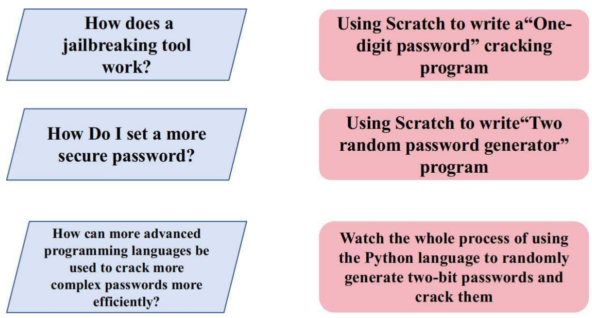


Figure 2: "Three questions, three tasks" schematic

4.2.2.Interdisciplinary Learning

Interdisciplinary learning is a deep learning approach that integrates knowledge synthesis and problem-solving, serving as an important implementation path for curriculum integration in the era of literacy development [11]. By combining concepts and methods from different disciplines, it can optimize problem-solving solutions and promote the integrity and rationality of students' high-level thinking. In the lesson of "Cryptography," to illustrate the enumeration algorithm, the problem of "chickens and rabbits in the same cage" from primary school mathematics was introduced, enabling students to understand that the characteristic of this algorithm is to exhaust all possibilities to solve the problem. Although it is not efficient, it is suitable for scenarios where there is no pattern to follow, such as password decryption. This cross-disciplinary transfer enhances the stickiness of knowledge and the activity of thinking.

To help students understand that the core of "programming" is algorithms rather than syntax, this lesson broke through the limitations of programming software. After experiencing decryption and encryption in Scratch, the observation and experimental methods from science class were introduced to demonstrate the process of breaking passwords with Python programs. This allowed students to experience the differences in time efficiency between different languages and focus on comprehending the computational thinking behind them.

4.2.3.Enhancing the Sense of Value

To enhance a sense of identity requires a comprehensive evaluation, such as the "new three-dimensional goal" framework, which recognizes knowledge, abilities, and moral character, enabling students to understand themselves more objectively. In the teaching of "Cryptography," immediate teacher evaluation is paralleled with student self-evaluation, making the evaluation more three-dimensional and objective. At the same time, to deepen the educational value, it is necessary to broaden the learning areas, such as increasing reading, participating in social practices, exploring interdisciplinary fields, cultivating critical thinking, and setting examples to form positive values. Activities like watching videos of academicians and making collective commitments to safeguarding password security not only enhance students' awareness of information security but also demonstrate the deep integration of the "new three-dimensional goal" with computational thinking, serving as a typical model for cultivating core literacy.

5. SUMMARY AND PROSPECT

Education in the age of intelligence is undergoing profound changes, demanding higher core literacy from individuals. Computational thinking, as a key mode of thinking, injects new vitality into the "new three-dimensional goal." It deepens students' understanding of knowledge, guides them to solve practical problems, and cultivates their innovative consciousness and sense of responsibility.

Looking ahead, we should delve deeper into the integration of computational thinking and educational innovation, integrating it into curriculum design, teaching methods, and evaluation systems. We should also expand its applications through AI, blended learning, and personalized learning. Only with the joint efforts, attention, and support of all parties for the application of computational thinking in education can we cultivate talents of the new era who possess high-level knowledge literacy, problem-solving abilities, and correct values, and jointly build a better future!

6. REFERENCES

Zhang Hua. On the core quality of curriculum: towards the “New three-dimensional goal” [J] . Education research, 2023,44(09) : 76-85.

so wah-ching. Practice and thinking of in-depth teaching based on progressive knowledge [J] . Tianjin education, 2020, (34) : 43-44.

Yu Ying, Wang Weiquan. Problem solving: The Logic mainline of the integrated implementation of information technology curriculum [J] . Information technology education in China, 2024, (02) : 4-11.

Li Panpan, Zhang Wei, Zeng Xinyao, Song Lingling, Liu Jianfang. A framework for evaluating the core elements of computational thinking based on Brumm's classification of educational goals [J] . Software Guide, 2023,22(02) : 160-165.

Li Ying, Xie Zhongxin. Research on evaluation of computational thinking based on SOLO classification theory —— taking information technology as an example [J] . Basic Education curriculum, 2023, (23) : 60-66.

Guo Kaiqi, Cao Dianbo, Deng Jie, etc. . The strategy to realize the educational goal of the compulsory education information technology curriculum under the background of the new curriculum standard [J] . Teaching and management, 2023, (21) : 94-99.

Cui Yunjun. Teaching-learning-evaluation consistency: the key to deepening the reform of curriculum teaching [J] . Chinese basic education, 2024, (01)

an Guiqing. On the comprehensiveness and practicality of compulsory education courses [J] . Global education outlook, 2022,51(05) : 14-26.

Queen Xiong. Types and teaching functions of “Problem chain” —— taking chemistry teaching as an example [J] . Educational Science Research, 2010(05) : 50-54

AIGC's Potential and Feasibility in Fostering K-12 Computational Thinking

Jiayi Yang[1]
[1]Beijing Normal University, China
yangjiayi1003@126.com

ABSTRACT

Under the development wave of Artificial Intelligence Generative Content (AIGC), China has entered the Education 4.0 era, which urgently requires digital transformation and the high-quality development of education. Furthermore, there is a growing demand for higher standards in talent cultivation, with computational thinking emerging as an essential foundational skill for individuals. This study adopts a systematic literature review method to explore the potential and feasibility of leveraging AIGC in fostering computational thinking at the K-12 level. Through a comprehensive review of existing research in this field, our aim is to uncover what is possible and achievable for AIGC to contribute to the development of computational thinking. The findings of this study indicate that applying AIGC in programming education, particularly in providing personalized instruction and conducting whole-process computational thinking level assessments, can significantly enhance students' problem-solving skills and foster their computational thinking abilities.

KEYWORDS

AIGC Computational thinking K-12

1. INTRODUCTION

With the rising wave of computational thinking, AI education is rapidly emerging. The development and popularization of AIGC tools such as ChatGPT, and Midjourney have contributed to the transformation of education in China by providing students with a more hands-on learning experience, helping them to better understand and master computational thinking. Schools mainly develop learners' CT by introducing programs such as programming, robotics, etc.

Our study investigates the impact of AIGC on the development of computational thinking in K-12 students, helping teachers and researchers to fully understand the importance of integrating generative AI into education. Instead, our study focuses on outcomes and analyzes the possibilities of AIGC's impact on the field of education and on the development of computational thinking in learners. Feasible recommendations are made for AIGC to promote computational thinking development at the K-12 level.

The following research questions are trying to be proposed:

RQ1:What are the trends in computational thinking development from 2015 to 2023?

RQ2:What are the possibilities and value of AIGC in developing computational thinking in students?

RQ3:What are some instructional strategies for developing computational thinking using the AIGC?

2. METHODOLOGY

This study adopts the systematic literature review method. The steps of literature search and evaluation are shown in Figure 1. CNKI Academic Journals Full Text Database was used as the data source, and "Generative Artificial Intelligence" & "Education" and "Computational Thinking Cultivation" as the search topics. "Education" and "Computational Thinking Cultivation", select the domestic academic literature published in CSSCI source journals, and set the time from January 2015 to December 2023. The time period is set from January 2015 to December 2023, because after 2015, the research is gradually expanding from university to K12 level. A total of 232 articles of domestic literature were screened; the keywords "Artificial Intelligence Generated Content" and "Training of computational thinking" were used as the research terms. The keywords "Artificial Intelligence Generated Content" and "Training of computational thinking" are used, and 66 articles of foreign literatures are obtained from the core database of Web of Science. The total number of foreign literature is 66.

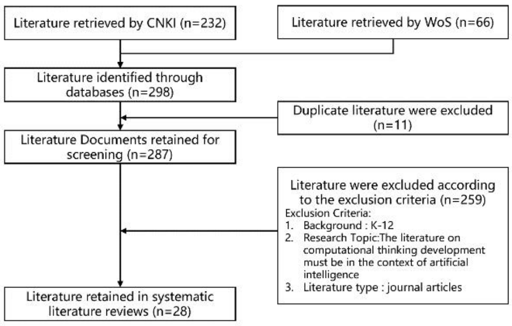


Figure1 Literature Screening Procedure

To ensure the reliability and consistency of the literature, the search results had to be verified individually, and the initially acquired literature was selected through three filtering conditions.

Reading the titles and abstracts according to the screening criteria and eliminating the literature that is not highly relevant to the research topic of this paper, we finally got 28 valid literature.

3. RESULTS OF THE REVIEW

Based on the research questions, our analysis yielded the following results:

3.1. What are the trends in computational thinking development from 2015 to 2023?

Exploring the content of K-12 Computational Thinking development is essentially a deeper reading of its concepts. China's general high school information technology curriculum standard states that students with computational thinking, construct solutions to problems by integrating resources and applying reasonable algorithms; summarize the process and method of computerized problem solving and transfer it to related problem solving (MOE,2018). In the era of artificial intelligence, human intelligence should be iterated forward with the help of intelligent tools that are fundamentally different from human beings, horizontally breaking through human knowledge barriers and downwardly compatible with human foundations and differences, with a view to realizing human intelligence (Yang, 2023).

3.2. Possibilities and value of AIGC in developing students' computational thinking?

Computational thinking, which is inseparable from programming teaching. On the one hand, programming teaching is an effective way to cultivate computational thinking; on the other hand, the cultivation of computational thinking is an important goal of programming teaching. Relevant literature suggests that programming is an effective framework for developing CT skills (Sun, Hu,& Zhou, 2022; Angeli & Giannakos, 2020). The AIGC tool can help students develop computational thinking by providing resources and guidance for programming learning and realize the enhancement from the skill manipulation level to the innovation and creativity level.

Computational thinking is a generic skill that is not only relevant to computing but crosses disciplinary boundaries (Teaching London Computing, 2017). However, when combined with other disciplines, the goals of developing computational thinking need to be coordinated with the curriculum goals and carried out in a way that is specific to the discipline. This is often a challenging task. AIGC can provide a variety of innovative teaching methods according to the course objectives and the characteristics of the discipline.

Educators need to take appropriate steps to verify that students are actually mastering what they are learning. The AIGC tool can not only assess according to the existing assessment methods, but also collect the process data of learning based on the digital mobile devices, realizing the transition from outcome-based assessment to process-based assessment, and from selective assessment to developmental assessment.

3.3. What are some instructional strategies for developing computational thinking using AIGC?

3.3.1. Innovative applications in programming education

AIGC can help students understand programming concepts, master programming languages and tools, etc., which are all important components of computational thinking, by providing programming-related knowledge, skills and examples. Meanwhile, the age and gender of students will affect the effect of programming. AIGC can carry out programming teaching according to the age and gender characteristics of students, provide personalized programming learning plans and recommend relevant learning resources according to students' learning needs and interests, provide real-time feedback to help students carry out individual exercises, and help students complete individual programming ability level testing to help them learn and practice programming better, thus Help learners' computational thinking development.

3.3.2. New Approaches to Personalized Instruction and Higher Order Thinking Development

AIGC tools such as ChatGPT can be used as easy-to-use chatbots to generate customized and personalized content to a certain extent for students of different age groups and learning levels by conversing with users. , which carries out personalized teaching and precise instruction. In terms of teaching, it can provide many auxiliary functions such as learning situation analysis, homework correction and feedback, learning data monitoring and evaluation, etc., changing the traditional education's excessive focus on knowledge transfer to the cultivation of higher-order thinking such as computational thinking. In addition, since the content generated by generative AI such as ChatGPT is uncertain, students first need to understand this uncertainty and think about and determine the authenticity, and this process is essentially the process of judgmental thinking formation.

3.3.3. A new engine for personalized assessment

In terms of evaluation, AIGC tools can evaluate learners' learning effectiveness by collecting, organizing and analyzing learners' whole process learning data. In addition, AIGC-supported learning evaluation will realize the evaluation of learners' higher-order abilities, such as critical thinking, problem solving ability, reasoning and reflection ability, etc., by visually presenting the learners' thinking trajectory, problem solving process, etc., and shifting from the traditional test evaluation to personalized evaluation.

4. CONCLUSIONS

AIGC has great potential and value in developing students' computational thinking. Applying AIGC to education allows students to experience AI technology, stimulate interest in exploration, and enhance their thinking skills. By providing programming learning resources, AIGC can help students advance from the skill operation level to the innovation and creation level. AIGC provides innovative teaching methods such as accurate assessment and resource pushing to promote the personalized development of learners' higher-order thinking. Its assessment methods better reflect students' computational thinking level and provide feedback for educators to optimize teaching strategies.

To further explore how the AIGC tool can be utilized to more effectively develop students' computational thinking, future research could develop targeted instructional programs and assessment methods. It is also necessary to focus on students of different ages and learning levels to develop personalized instruction and precise guidance to better promote the development of students' computational thinking.

5. REFERENCES

Angeli, C. , & Giannakos, M. N. . (2020). Computational thinking education: issues and challenges. pergamon.

Ashfaq, M., Yun, J., Yu, S., Loureiro, S.M.C., (2020). I, Chatbot: Modeling the determinants of users' satisfaction and continuance intention of AI-powered service agents. Telematics Inform . 54, 101473 https://doi.org/10.1016/j.tele.2020.101473.

Bers, M. U., Gonz´alez-gonz´alez, C., Bel´en, M., Torres, A., Study, C., Development, H., et al. (2019). Coding as a playground : Promoting positive learning experiences in childhood classrooms.Computers&Education,138(April),130145. https:// doi.org/10.1016/j.compedu.2019.04.01

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing Computational Thinking in Compulsory Education - Implications for policy and practice.

Bocconi, S., Chioccariello, A., Kampylis, P., Wastiau, P., Engelhardt, K., Earp, J., et al. (2022). In A. I. dos Santos, R. Cachia, N. Giannoutsou, & Y. Punie (Eds.), Reviewing computational thinking in compulsory education.

Cortese, A. (2022). Metacognitive resources for adaptive learning. Neuroscience Research, 178, 10-19.

Ministry of Education (2018). Information Technology Curriculum Standards for General High Schools (2017 Edition) [M]. Beijing: People's Education Press: 5-6.

Pea, G. R. . (2013). Computational thinking in k-12. Educational Researcher, 42(1), 38-43.

Shute, V. J., Sun, C., & Asbell-clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142-158. https://doi.org/10.1016/j. edurev.2017.09.003

Sun, L., Hu, L., & Zhou, D. (2022). Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. Computers and Education, 181(27), Article 104457. https://doi.org/10.1016/j. compedu.2022.104457

Tarnavsky Eitan, A., Smolyansky, E., & Knaan Harpaz, I. (2020). Connected papers. Connected papers. About. https://www.connectedpapers.com/.

Teaching London Computing (2017).Interdisciplinary Computational Thinking [EB/OL]. [2019-5-30]. https://teachinglondoncomputing.org/interdisciplinary-computational-thinking/.

Wing, J. M. . (2017). Computational thinking's influence on research and education for all. consiglio Nazionale delle Ricerche (CNR)(2).

Yadav, A., Mayfield, C., Zhou, N., & Hambrusch, S. (2014). Computational Thinking in Elementary and Secondary Teacher Education, 14(1).

Yang, X. (2023). The educational transformation landscape based on generative AI - What does ChatGPT mean for education? Chinese Journal of Educational Technology (5), 1-8. 1-8.

Yu, Y., & Chen, Y. (2018). Design and development of high school artificial intelligence textbook based on computational thinking. Open Access Library Journal, 5(9), 1 -15.

Luo, X., Tong, S., Fang, Z., Qu, Z., (2019). Frontiers: machines vs. humans: the impact of artificial intelligence chatbot disclosure on customer purchases. mark. sci. 38 (6), 937- 947.

Research on the Application of STEAM Teaching Concept based on Project-based Learning in English Teaching

Liqiao Nong[1], Jianyun Ma[2*]
[1]Guangxi Polytechnic of Construction, China,
[2]Henan University of Science and Technology, China
979620226@qq.com, 760959089@qq.com

ABSTRACT

STEAM education involves science, technology, engineering, arts and mathematics disciplines, emphasizing that students pose practical problems and apply interdisciplinary knowledge to solve problems, and develop students' problem-solving and innovation skills. This study attempts to explore the integration of STEAM education concepts into English courses in order to promote the development of English teaching. By means of literature research, this study conducted relevant research on the connotation and characteristics of STEAM education concept at home and abroad, the integration of STEAM and language subjects, and analyzed the appropriateness of STEAM education concept based on project-based learning in English teaching and the design of teaching mode. With the teaching form of four links: theme selection, task assignment, project exploration and achievement display, this study takes Computer English as the course carrier, and the instruction implementation is carried out for freshmen students, forming a typical case of the application of STEAM teaching concept based on project-based learning in English teaching.

KEYWORDS

project-based learning, STEAM education, English teaching

1. INTRODUCTION

STEAM education originated in the United States in the 1990s, the concept of STEM was originally proposed by the American Science Board in the report, STEM means the integration of Science, Technology, Engineering and Mathematics. It originally implemented in undergraduate education, and it aims to cultivate the practical ability of science and engineering students. In response to the lack of creativity in its implementation, Virginia University of Science and Technology scholar G.Yakman also proposed to add "A" (Art) to STEM, and proposed STEAM education, which means that the art and humanities and social sciences and other disciplines are integrated into STEM to provide creative sources for science and engineering practice. STEAM education is an educational concept and practice that integrates liberal arts, science and engineering. STEAM education supports students to understand the world in an integrated way and transform the world in the form of comprehensive innovation, so that students can learn integrated knowledge in the process of solving real problems, and at the same time, gain the inherent ability to design problems, solve complex problems, cooperate, make decisions and innovate and create, rather than only accept the knowledge content of a single discipline. STEAM education is a project-based learning (PBL) or task-based learning (TBL) way of learning. It emphasizes the study and application of interdisciplinary knowledge to solve problems in life. STEAM education focuses on knowledge learning and ability cultivation, a "learning" approach based on "doing", and focuses on the application of multidisciplinary knowledge and the relationship between students. As an instrumental language discipline, English is a good carrier to take the project-based STEAM teaching into implementation. However, how to apply the STEAM teaching concept based on project-based learning to train students' comprehensive ability in English teaching is the highlight of this study.

2. LITERATURE REVIEW

2.1. Research on Educational Applications of STEAM

In the research field of STEAM education, Chinese scholars mainly analyze the connotation, characteristics and implementation strategies of STEAM education. (Yang Mingquan, 2024) analyzed the concept of STEM, and he believed that the term "STEM education" has multiple connotations that are interrelated and have different directions: it is not only a research field, a concept of education, but also a curriculum form and a teaching method. He believes that the current STEM education is facing practical problems such as the solidification of teaching methods, unclear education goals, and insufficient teachers' ability. Only by integrating STEM education research and its localization transformation and innovation into the internal composition of China's education reform can STEM education develop sustainably. Based on the analysis of domestic STEAM education research hotspots, combined with China's national conditions and education status, (Dong Hongjian & Hu Xianyu, 2020) placed expectations on the development of "STEAM+" in STEAM education to improve the STEAM teacher training system and establish a diversified evaluation system. In a word, research on STEAM education are various in a variety of disciplines. In recent years, more scholars tend to study STEAM education from application mode, specific teaching, curriculum design and other dimensions, and try to innovate the development path of STEAM education based on the actual situation of the country and the actual needs of students.

2.2. Research on the Application of STEAM in English Teaching

The research on the application of STEAM education concept in English teaching mainly focuses on the application of the interdisciplinary concept of STEAM education and the project-based learning method, which contributes to the innovation and upgrading of English teaching. It summarizes and refines the STEAM competence of teachers, STEAM education system, STEAM curriculum resources development and other new research areas. Based on the project-based teaching and interdisciplinary concept under the STEM concept, (Xu Shengnan, 2020) designs English activities, handicraft making, scientific experiments and other learning projects in the English learning of science and engineering students. Focusing on the innovation and upgrading of English teaching for engineering students, (Li Yongmei, Zhang Dongxia & Tan Xin,2019) tried to introduce the interdisciplinary concept of STEAM education, project Language teaching model (PBLL) and English teaching methods based on information technology in English teaching, so as to promote the innovation of college English teaching model. All in all, under the guidance of STEAM, English teaching has gradually shifted from focusing on the mastery of English knowledge points to focusing on the input, output and application of English knowledge in real situations, which requires more scholars to study the application of STRAM education in English subjects.

3. RESEARCH ON STEAM ENGLISH TEACHING MODEL BASED ON PROJECT-BASED LEARNING

Project-based learning is a teaching process guided by constructivism, which promotes the development of students' comprehensive ability through high-investment inquiry learning in real situations. The selection of project themes, project division, project exploration, project output and presentation of diverse evaluation and reflection require the participation of students. Students are always the main body of learning in project-based learning, and teachers play the role of mentors, collaborators and helpers. The STEAM English teaching mode based on project-based learning is divided into thematic selection, task assignment, project exploration and group presentation. The realization of the project is an important support for English learning knowledge points and language skills learning, and it is also the guarantee of the successful completion of project-based learning. When selecting project topics, it is necessary to consider that the selected topics should not be closely related to students' personal life, professional and textbook content, reflecting the practical significance of project-based learning inquiry. In addition, the determination of the project theme should also consider the regularity of students' physical and mental development, and should fit in with their cognitive characteristics and interests. Project tasks should also be realized by students through group cooperation. Secondly, students should be guided to divide the work among group members and arrange the exact task progress for the task of the group cooperation project. After fully and deeply understanding students' learning characteristics and strengths, teachers guide students to voluntarily assign groups according to their own interests and needs, ensuring that each member of the group can fully participate in class activities and show their strengths in project-based learners. Project inquiry is the core link in the process of project-based learning. In the process of interactive inquiry, students participate in classroom learning by actively experiencing and exploring knowledge, and fully highlight their subject status. In the process of interactive inquiry, students participate in classroom learning by actively experiencing and exploring knowledge, and fully highlight their subject status.

4. A PRACTIAL CASE OF STEAM ENGLISH TEACHING BASED ON PROJECT-BASED LEARNING

Taking the computer English course as an example, the teaching object is computer major college students, and the project theme is Design Proposal. This project aims to go through the process of selecting an ISP (Internet Service Provider). The whole task is divided into three steps. Step One is about an overview of all ISPs. Step Two focuses on information collection. Step Three rests on making comparisons and presentations. The project follow three steps, which is Step One: Organize a small group with 4 – 6 people in the class; Share the work of researching online resources for information about ISP companies in China; Summarize what students have found and just select one of them to research in depth. Step Two: Find information about the selected ISP company as much as possible, including its cost, service types, fees, etc.; Interview some clients of the ISP company, and summarize the different clients' comments on the company; Illustrate what students have found about the company in the form of a table. Step Three: Compare what students have found with another group, and find out your ISP company' s strong points and weak ones; Make a presentation about your ISP company in front of the whole class; Select the best ISP in your mind. Finally, students present their research results in the form of PPT, and discuss in the class to make formative evaluation, as shown in Figure 1.

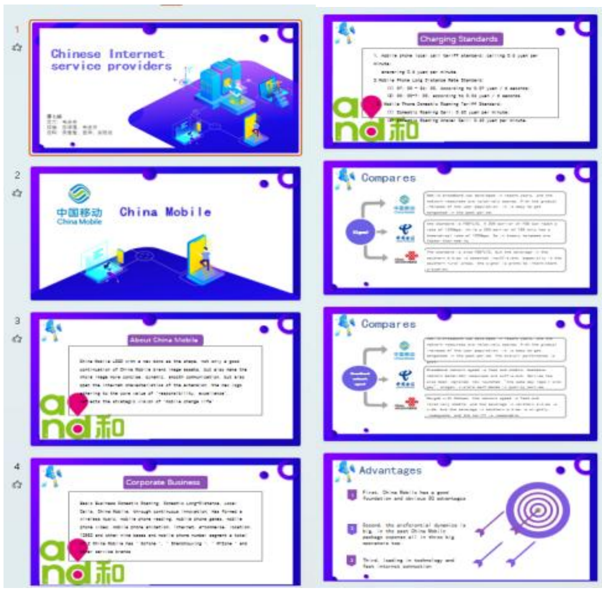


Figure 1. Presentation of the project

5. CONCLUSION

This research is based on constructivism theory, humanism theory and cognitivism theory through the literature study and research of STEAM education concept and STEAM integration with disciplines. An English themed activity teaching design scheme based on STEAM concept is designed to enhance class participation, stimulate students' learning interest and improve students' confidence in English learning, and the teaching practice has proved that the teaching design has achieved the expected results. In the future teaching process, it is necessary to continue to learn knowledge related to STEAM education and conduct in-depth research, to explore the interdisciplinary linkage mechanism and teaching incentives between STEAM and English teaching, and to continuously promote English theme-based project-based activity teaching based on STEAM education concept in teaching practice to improve students' classroom participation.

Funding: This research was funded by Guangxi Polytechnic of Construction School Project "Research on the Creation of Intelligent Teaching Environment and Innovation of Teaching Mode under the Background of Digital Transformation" (Project No.: 2023YB013).

6. REFERENCES

Dong Hongjian, & Hu Xianyu. (2020). Research and analysis of STEAM education in China and its future prospects. Modern Educational Technology, 27 (09), 114-120.

Li Yongmei, Zhang Dongxia, & Tan Xin. (2019). An Analysis of innovative foreign Language teaching Paths in new Engineering universities based on STEAM model. Future and Development, 43 (04), 104-107.

Xu Shengnan. (2020). Research on the Model of English "Learning by Doing" Innovation and Entrepreneurship Project for Science and Engineering College Students based on STEM education. Journal of Heilongjiang Teacher Development College, 39 (02), 132-134.

Yang Mingquan. (2024). On the localization construction of STEM Education: Connotation, Value and Practice exploration. Modern Distance Education Research, 36(01), 39-45+53.

A Few Suggestions to Algorithm Teaching in 5&6 Grade Computational Thinking Education

Xinxin Zhang[1]
[1]Shijia Primary School, P.R.China
64483049@qq.com

ABSTRACT

Computational thinking is solving problems by abstracting, decomposing and modeling the problem into forms that could be simulated and validated by computer. To do so, the key is the algorithm we use. When teaching Algorithm to students in primary school, or students of any other age, mathematical knowledge is an inevitable part of the course. However, due to the cognition of the students in primary school, they are easily confused by whether they are learning math or computer science. This should raise attention to algorithm teaching in primary school, how to put prominence to the features of algorithm teaching and help the students understand why we should study algorithm. This paper gives out suggestions by the writer's own teaching experience to algorithm teaching in primary school according to the matters that might occur in present teaching situations and the characteristics of the students, for which put prominence to the features of algorithm teaching by comparing to mathematical teaching.

KEYWORDS

algorithm teaching, primary school, teaching suggestions

1. INTRODUCTION

The objective in the computer classes which is now called "information science & technology" in primary school has raised from certain skills and the use of certain software to a more sophisticated level since the issue of the "compulsory education curriculum plan and standards" at march 25th, 2022. In which it lines out four major qualities students should acquire through the information science & technology course, information awareness, computational thinking, digital learning and innovation, information responsibility.

The description of "computational thinking" is the thinking activities involved in problem-solving, such as abstraction, decomposition, modeling and algorithm design, that individuals use in the field of computer science. Students with computational thinking are able to abstract, decompose, model problems and form solutions through designing algorithms. They are able to simulate process and verifying problem-solving solutions, reflect on and optimize these problem-solving solutions, and apply them to other situations. (China Compulsory Education Curriculum Plan and Standards for Information Science &Technology, 2022, p.5)

Obviously, we ought to focus on developing the students' thinking mind during classes. But how do we know what is going on in their mind? We used to rely the answers students brought up and the works they done to estimate the level of comprehension. When it comes to "computational thinking", there are no straight answers in problem-solving, only solutions that are formed by a series of steps which makes up an algorithm. How students use algorithms to solve certain problem shows the development of their "computational thinking".

2. LITERATURE REVIEW

This literature review will show the studies of algorithm teaching and the challenge within.

When teaching algorithm teachers trends to pass on the formula of an algorithm to the student as in math class, as a result, students memorize the formula instead of forming it with their own intellectual work, this type of teaching method is ineffective. (Mathematics, 2022, p. 3857). In middle school and high school teaching algorithm is usually combined with programming, which concretize the abstract theory and helps students understand. But is algorithm only used for programming? If we give students this impression when teaching algorithm, it might limit the use of algorithm outside of class. Research shows that postsecondary students are largely unaware of the impact of algorithms on their everyday lives. The teaching content for "information science & technology" in primary school is mostly presented in the form of text, charts, and images, which is highly theoretical. Algorithm as an even more abstract concept, will be harder for primary school students to understand. There are teaching tools and methods that are already in use in teaching math and programming which are also useful in teaching algorithm, but what is the particular feature in algorithm class that distinguish it from math and programming class is what I have been thinking and working on while teaching algorithm.

3. PRESENT SITUATION

I teach over 300 students in 8 classes of 18 in 5th grade, 1 class hour each week for every class, 18 weeks each semester. It has been nearly 3 semesters now, led the students into 6th grade, there are 3 major issues that students are confused with:

3.1. Why should we study algorithm?

The math foundation students built will help them understand algorithm, but sometimes it helps a little too much. Algorithm in Chinese translation is "Suan Fa", it is easily misinterpreted as "ways to calculate", students are familiar with this concept in math class, such as:

Table 1. Example of Ways to calculate in math class.

associative law	A x B x C = A x (B x C)

commutative law	A x B = B x A
distributive law	(A + B) x C = A x C + B x C.

These methods can easy the calculation which brings the result faster. But when it comes to a real-world problem rather than problems on paper, it takes logical steps which forms an algorithm rather than just calculation. If students can't to realize the value of algorithm in solving real-world problems, it could be the gap for them when it comes to bringing solutions to a more complexed problem.

3.2.We already know!

Based on the cognition of the students, we teach the classic algorithms such as, enumeration, recurrence, divide and conquer. However, some of the ideology has already been shown in their math class, such as "find the flawed(lighter) product using a scale".

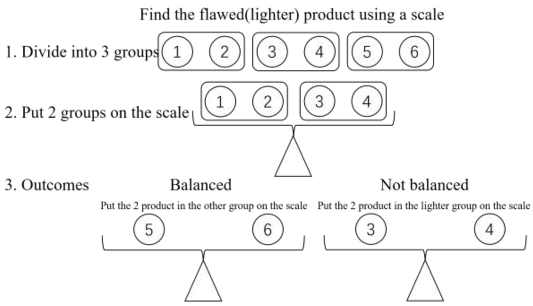


Figure 1. "Find the flawed(lighter) product using a scale"

Although algorithm is a new subject in primary school, if we can't acknowledge the fact that some of its ideology has already been taught in math class, there are chances that students have already solved the problem you brought up, this it will weaken the interest in learning algorithm.

3.3.Am I in a math class?

When we explain the principle of an algorithm it is very common to use math formula, equations, symbols etc., students are confused with the question, "am I in a math class?", even teachers have the same question, "what is the feature of algorithm class that differs it from math?".

When it comes to solving password problems, which is classic in teaching enumeration, you need to calculate the range of the enumeration to show the difficulty in breaking the password:

Table 2. Calculate the Range of the Enumeration.

Combination	Numbers	Numbers & Letters	Numbers & Letters with case sensitivity
Enumeration range	10 x 10 x 10	$(10+26)^3$	$(10+26 \text{ x } 2)^3$

When it comes to teaching recurrence, the classic come to the Fibonacci sequence also known as the rabbit sequence, we also need to summarize the formula:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89……

$$f(n) = f(n\text{-}1) + f(n\text{-}2)$$

The formula helps students understand the algorithm, but differ from math class the results of the calculation it is not the solution, it might be the answer to one of the steps in an algorithm. Thus, in algorithm class calculation and formula has its own use, but paying too much attention to it might end up failing to come up with the full solution.

4. SUGGESTIONS

To clear the confused minds for both teacher and students in teaching and studying algorithm we can try these strategy:

4.1.Answer "why should we study algorithm?"

It has come to peoples' senses that computer is a powerful tool in any field of work, the questions is can it help you solve particular problems. The key is to understand how its functioned, we can compare it to how the human brain solves an equation, such as "X + 3 = 7", it is so easy that students can give out the right answer in an instant, but computer solves it in a different way, by putting numbers into "X" until the equation stands.

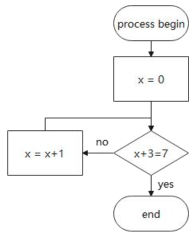


Figure 2. Process of Solving "X+3=7" with Computer

Describing the algorithm can help you understand how the problem is solved and shows how the computer is functioned. With algorithm you have the help of the computer, it brings advantages when facing complex problems. Once the students understand "why should we study algorithm?", it will bring up their interest and motivation in class.

4.2.Challenging tasks shows advantage of algorithm

According to "Zone of Proximal Development", the task in class should be in reach of students' abilities. With the use of algorithm, the tasks should be beyond the reach of just using math methods.

There is an old Chinese question from "Zhang Qiu Jian Suan Jing", "$5 for a rooster, $3 for a hen, 3 chicks for $1, can you buy 100 chicken with $100?", although students can come up with the equation group easily, but solving it is beyond primary school math:

$$a + b + c = 100$$
$$5a + 3b + c/3 = 100$$

The method to solving this equation group is in 7th grade math, but using algorithm students can solve it in 5th grade:

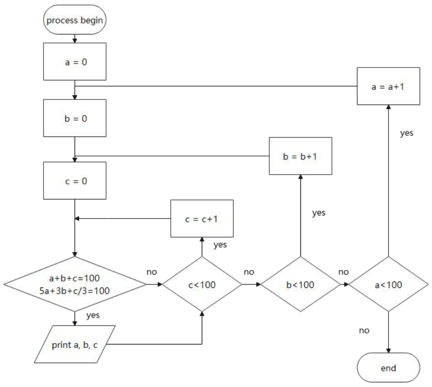


Figure 3. Process of Solving “buy 100 chicken with $100”

Trying every possible number to find values that fits the equation group seems like an impossible mission, but using algorithm and run it through the computer results will come out in minutes.

4.3.Focus on the process

When we got a wrong answer in math it’s a calculation error, but when we run an algorithm with the computer and failed, it’s the steps in the process that has gone wrong. So teaching algorithm we must focus more on the process rather than the answer. To do so, the description of the algorithm must be accurate.

Sometimes we can even try just using descriptions to break down complexed problem into smaller problems, repeat this process, which is divide and conquer algorithm, until the problem is small and simple enough for us to solve.

The story of “The Martian” a great example for students understand the importance of dividing the problem.

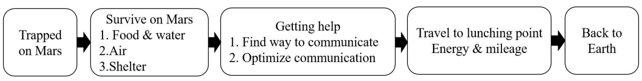


Figure 4. Dividing the Problem

Clips from a movie can light up the algorithm class, draw the students’ attentions and led the students realize the use of algorithm is beyond programming. Although the story is based on Mars, but some of the problem, such as communication, travelling etc., can also happen in real life on Earth.

4.4.More than just programming

We can find the use of algorithm in various games strategies. The rules for “The Tower of Hanoi” are simple, move the tower by layer, each layer can only land on a bigger layer. This game is a classic use of the recursive algorithm.

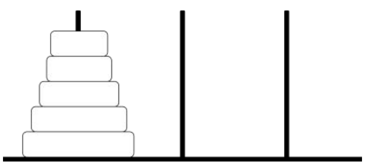


Figure 5. Model of “The Tower of Hanoi” Game

Class are carried on through these sections:

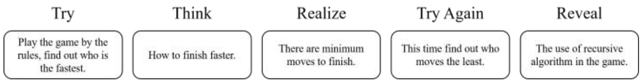


Figure 6. Activities in Class

The achievement gathered by playing games will drive the student to keep on trying, thinking, and finally understand the theory of recursive algorithm. This will lead students to realize the use of algorithm is not just about programming.

5. CONCLUSION

Algorithm is a new subject in primary school, students need certain math knowledge and programming skills to start, but it is neither math nor programming. Math is used to analyze the efficiency of an algorithm while programming is used to simulate and verify an algorithm in the computer. The essence of algorithm is the clear description of each step in the solution, by understanding every step in an algorithm and the logic between, we can adjust it to our own need, optimize it to increase efficiency, apply it to other cases.

It is important to show the advantage of using algorithm in class, it will enhance the students’ ability to solve problems, when they realize the power of algorithm, it will increase their interest and motivation in class. The feature of teaching algorithm is that we should focus on the process, the right process is the reflection of a deliberated mind. Leave the calculation to the computer since it is more efficient and accurate.

The classic problems of algorithm were brought up far before computer was invented, I would like to think that the use of algorithm comes from math, and now it is widely used in the field of computer science and technology, but is doesn’t stop here. We can’t give students the impression that algorithm is only used in programming. Thus, we should bring more elements other than computer into the classroom, games, movies etc., hopefully we can create a more productivity, interesting and fun algorithm class.

6. REFERENCES

Archambault, Susan, Gardner. (2024). Toward a new framework for teaching algorithmic literacy. *Information and Learning Sciences, 125*(1-2), 44-67.

Gonda, Dalibor, & Ďuriš, Viliam, & Tirpáková, Anna & Pavlovičová, Gabriela (2022). Teaching Algorithms to Develop the Algorithmic Thinking of Informatics Students. *Mathematics, 10*(20), 3857-3857.

Hou, Hejia. (2023). Research on the Design and Practice of Digital Game Teaching Activities from the Perspective of Large Units ——A Case Study of Information Technology Algorithm Teaching in Primary Schools [D] GanSu: Northwest Normal University, June, 2023.

Ministry of Education of the People's Republic of China. (2022). *Compulsory Education Curriculum Plan and Standards.* Beijing: Beijing Normal University Publishing Press.

Wu, Caiyun (2023). Exploration of Information Technology Education Methods in Primary Schools under the Concept of Computational Thinking. *Primary and Middle School Educational Technology*. (10), 58-60.

Research on International Trends in Information Technology Education

Tiantian Gan[1], Qianqian Chen[2*],Qian Fu[3]
[1,2,3] Beijing Normal University, China
1833440798@qq.com, 202322010211@mail.bnu.edu.cn , fredqian@bnu.edu.cn

ABSTRACT
With the advent of the digital era, nations around the world are actively exploring and implementing contemporary approaches to information technology education that align with current needs, thereby driving the continuous evolution of educational systems. This study conducts a comprehensive analysis of the current state of information technology education in the United States, the United Kingdom, and Japan, focusing on curriculum standards, policy environments, and instructional content. By delving into these aspects, the research aims to identify common trends. The primary objective is to provide valuable insights and references for the future development of information technology education in our country. This includes aspects such as computational thinking education and STEM education related to computation. The findings aspire to serve as a crucial reference point, laying a solid foundation for the cultivation of a new generation of talent equipped with advanced digital literacy.

KEYWORDS
Information technology, curriculum standards, policy environment, curriculum content

1. INTRODUCTION
As human society enters the digital age, the significance of information technology education becomes increasingly prominent. Countries worldwide are actively exploring international perspectives on information technology education that align with contemporary needs, thereby propelling the continuous development of educational systems. Considering to the global influence and advancement of the United States, the United Kingdom and Japan in the field of information technology education. This report provides an in-depth study of the current status of their information technology education. It investigates their curriculum standards, policy environments, and instructional content with the aim of providing valuable insights and inspiration for the development of information technology education in our country. Ask the following research questions:

1. What is the current status of information technology education in the US, UK and Japan?

2. What is the development trend of information technology education?

3. How to promote the vigorous development of information technology education?

2. Current Status of International Information Technology Education Development

2.1. Policies Related to Information Technology Educatioon
The United States has consistently led the world in information technology education. In the 1990s, there was a significant emphasis on computer education in primary and secondary schools. In 2011, the revised "CSTA K－12 Computer Science Standards" (CSTA & ACM, 2011) was launched, providing comprehensive standards for K-12 computer science education. The "CSTA K－12 Computer Science Standards" (CSTA, 2017) were further revised in 2017, emphasizing the introduction of basic computer science concepts to all students from elementary school onwards. It encourages schools to offer additional high school computer science courses, allowing interested students to delve deeper into various aspects of computer science.

The United Kingdom is among the earliest countries globally to incorporate information technology into its national curriculum, with the subject originating from the field of computer science. In 1987, the Department for Education and Science/Welsh Office in London (1987) released "The National Curriculum 5－16," stating that information technology should be included in the foundational curriculum for primary and secondary schools. In a new round of reforms in 2013, the ICT curriculum was transformed into the "computing" curriculum, accompanied by the successive release of drafts for computing programs of study (Department for Education,2013).

Japan began developing information technology education in the 1980s, with the government implementing a series of measures to advance information technology education, subsequently issuing curriculum standards and guidelines. In 2018, Japan revised the high school learning guidelines, introducing a new compulsory subject to strengthen the teaching content, and the revision was formally implemented in 2022(Ministry of Education, 2018).. In November 2019, the Central Education Council of Japan released a five-year timetable for campus ICT environment construction, planning to promote the development of school information and communication technology environments through a series of reforms. In September 2020, the "Information Education Curriculum Design Guidelines" were successively introduced, systematically planning both internal and external information education, achieving an integrated and coherent design of the information education curriculum system.

2.2. *Objectives and Contents of Information Technology Education Curriculum*

The United States lacks a uniform set of information technology curriculum standards, each state refers to the education plans issued and formulates its own course standards. In this context, the analysis focuses on the "CSTA K-12 Computer Science Standards" promulgated in 2017 (CSTA, 2017),which outlines five core concepts and seven core practices. It explicitly states that the curriculum objectives for computer science courses aim to enable all students to understand the concepts of computer science and engage in practical applications. Students are expected to grasp the essence of computer science, apply computer science skills, especially computational thinking, to problem-solving. The concepts section of curriculum contents includes "sub-concepts" and "interdisciplinary concepts," reflecting both the essential requirements of the discipline itself and the trend of interdisciplinary integration. From the proportion of core concepts, it is evident that "Algorithms and Programming" constitute a focal point of the computer science curriculum. Additionally, as students progress through grades, both the proportion and complexity of algorithms and programming gradually increase, intersecting with multiple points in the "Creating Computational Artifacts" core practice.

In February 2013, the United Kingdom announced a new national curriculum draft, including a computer curriculum, marking the replacement of the former Information and Communication Technology curriculum with a computer curriculum (Department for Education, 2013). The curriculum objectives aim to enable students, after completing the study of computer science principles and basic concepts, to apply computational thinking to analyze and solve problems, ultimately becoming creators of technology. It also seeks to cultivate students' digital literacy, transforming them into adept and responsible technology users in the information age. The "Computing" curriculum in the United Kingdom encompasses three domains: Digital Literacy, Information Technology, and Computer Science. Each module is divided into four key stages based on age, with detailed learning content specified. (Simon Peyton-Jones, 2014).

The newly revised "High School Learning Guidelines" in Japan established the course objectives for high school students: acquiring knowledge of information technology, developing skills in using information, understanding the role and impact of information technology in society, fostering information-oriented thinking, and cultivating the ability and attitude to adapt to the development of informatization.In the high school stage in Japan, the compulsory subject "Information I" and the elective subject "Information II" are offered. In the "Technology and Home Economics" subject in junior high school, the information technology component is comprehensively integrated. There is no separate course in elementary school; instead, the content related to information education is incorporated into comprehensive learning practices, mathematics, science, and other subjects. Simultaneously, information technology is widely and effectively applied in the teaching of various subjects . Japan plans for all elementary school students to study computer programming, equip all junior high school students with a terminal for learning, and have all students learn fundamental knowledge such as programming, networks, and databases.

The examination of curriculum objectives and contents in the aforementioned countries reveals distinctive approaches. The United States places greater emphasis on K-12 system design, the United Kingdom prioritizes phased education, while Japan focuses on establishing connections between information and society.(Refer to Table 1.) Computer science is universally recognized as a crucial knowledge domain across all nations, with a common focus on fundamentals such as data, programming, and algorithms.

Table 1. Curriculum objectives and content in each country

Country	Module	Content	
CSTA K-12 Computer Science Standards in USA	Core Concepts	1. Computer Systems 2. Networks & the Internet 3. Data & Analysis	4. Algorithms & Programming 5. Impact of Computing
	Core Practices	1. Fostering an Inclusive Computing Culture 2. Collaborating Around Computing 3. Recognizing and Defining Computational Problems 4. Developing and Using Abstractions	5. Creating Computational Artifacts 6. Testing and Refining Computational Artifacts 7. Communicating About Computing
	Crosscutting concepts	1.Abstraction 2.System Relationships 3.Human–Computer Interaction	4.Privacy and Security 5.Communication and Coordination
Computing Ccurriculum in England	Computer Science	principles of information and computation, how digital systems work and how to put this knowledge to use through programming	
	Information Technology	create programs, systems and a range of content	
	Digitally Literate	express and develop ideas through information and communication technology	
The Japanese Information Learning Guide	Information society and problem-solving	1.Information society and information technology development 2.Information security, ethics, and the law 3.Using information technology to discover and solve problems	
	Communication and Information design	1.Media and Communication	2.Information design
	Information systems and programming	1.computer system 2.Programming and algorithmic expression	3.Information system and information processing
	Information, communication network and data use	1.software development 2.Information and communication network 3.Data storage and management	4.Data processing and expression 5.Data modeling and evaluation

3. Common Trends in International Information Technology Education

From the development processes and achievements in information technology education across various countries, a synthesis of international trends in information technology education can be deduced, categorized into two main aspects: enhance external support and internal curriculum-level reform.(Refer to Table 2.)

Table 2. Enhanced external support and curriculum-level reform

Aspect	Trend
Enhanced External Support	Policy Guidance: Supporting the Development of Information Technology Education
	Environmental Optimization: Promoting Education Infrastructure Construction
	Technological Empowerment: Information Platforms Driving Educational Progress
	Professional Development: Teacher Growth Towards Modern Education
Curriculum -level Reform	Universalization of Computer Teaching across All Educational Stages
	Diversified Forms and Content Development
	Shift towards the Cultivation of Computational Thinking Abilities
	Encouragement of Technological Innovation and Practice

3.1. *Enhanced External Support*

In the realm of information technology education, countries worldwide are bolstering their efforts through enhanced external support mechanisms. This involves the formulation of comprehensive policies to provide clear directives for the development of computer science courses. Additionally, there is a concerted focus on promoting the construction of robust educational infrastructure, facilitating the seamless integration of computer technology into learning environments. Furthermore, collaborative endeavors led by government bodies, businesses, academic institutions, and civil organizations are driving the development of specialized software and hardware tailored for educational purposes. Simultaneously, considerable attention is directed towards the professional growth of educators, with an emphasis on equipping them with modern educational technology and information technology training. These multifaceted initiatives reflect a collective commitment to nurturing a technologically adept generation capable of thriving in the digital age.

3.2. *Curriculum-level Reform*

In striving for progress in information technology education, nations are undertaking comprehensive reforms at the curriculum level. This entails the widespread integration of computer teaching throughout all educational stages, underpinned by curricula that are both standardized and adaptable. Each country tailors its curriculum to its unique national context, fostering not only the acquisition of knowledge and skills but also the cultivation of critical thinking, practical abilities, and a sense of social responsibility. Moreover, there is a deliberate emphasis on instilling computational thinking skills, empowering students to approach complex challenges with systematic and abstract problem-solving techniques. Concurrently, there is a concerted push towards fostering technological innovation and practical application, involving the development of versatile software and hardware tools aligned with global standards and practical educational needs.

4. Insights and Suggestions

Through an analysis of the current status and common trends in information technology education in the United States, the United Kingdom, and Japan, there are insights for the development of information technology education in China.(Refer to Table 3.)

Table 3. Insights and Suggestions

Aspect	Branch
Curriculum Development and Integration	Layered Knowledge System
	Interdisciplinary Connections
Pedagogical Innovation for Higher-Order Thinking	Localization of Curriculum Standards
	Programming Logic Training
Localized Professional Development for Teachers	Integrated Policy Support
	Continuous Professional Growth
Strengthening Integration with Other Educational Content	Multidisciplinary Integration
	Comprehensive Literacy Improvement
Establishing a Social Participation Model	Integrated Educational Environment
	Collaborative Efforts

4.1. *Curriculum Development and Integration*

It is vital to organize the curriculum according to the students' stage of learning. Emphasis is placed on layering knowledge progressively according to the IT curriculum standards to ensure a systematic and coherent understanding of the subject. Emphasis is placed on progressively layering knowledge according to the IT curriculum standards to ensure a systematic and coherent understanding of the subject.

4.2. *Pedagogical Innovations for High-Order Thinking*

Localization of international curriculum standards plays an important role in developing higher-order thinking skills. In addition, programming logic training should be emphasized as an important means of improving students' critical thinking skills, creative problem-solving skills, logical analysis and synthesis skills.

4.3. *Localized Professional Development for Teacher*

Drawing on international experiences and models, we will formulate targeted policy support in the light of local realities. We advocate sustained professional growth, which requires collaboration among government, universities and educational research institutions, as well as the active participation of teachers themselves.

4.4. *Strengthening integration with Other Educational Content*

Encourage multidisciplinary integration and the use of multimedia to create personalized learning environments. The focus is not only on the learning of technological tools, but also emphasizes the development of teaching content based on real-life scenarios, problems or projects, thus enhancing students' overall literacy and understanding of IT applications.

4.5. *Establishing a Social Participation Model*

There is a need to create an integrated educational environment that spans the home, school and community. Collaboration between business, academia and civil society organizations is essential to provide resources, support and ensure that students adapt effectively to evolving technologies.

5. REFERENCES

Department of Education, Washington, DC. (1996). Getting America's Students Ready for the 21st Century: Meeting the Technology Literacy Challenge. A Report to the Nation on Technology and Education. Retrieved 1996, from https://files.eric.ed.gov/fulltext/ED398899.pdf.

The American Library Association and the Association for Educational Communications and Technology. (1998). Information literacy standards for student learning. Retrieved 1998, from https://www.ala.org/ala/aasl/aaslproftools/informationpower/InformationLiteracyStandards_final.pdf

Department of Education. (2003). A New K – 12 Computer Science Curriculum. Retrieved 2003, from https://files.eric.ed.gov/fulltext/EJ695762.pdf

Computer Science Teachers Association and Association for Computing Machinery. (2011). CSTA K – 12 Computer Science Standards. Retrieved 2011, from https://dl.acm.org/doi/pdf/10.1145/2593249

Computer Science Teachers Association. (2017). CSTA K – 12 Computer Science Standards. Retrieved 2017, from https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf

The National Curriculum and Science/Welsh Office London. The National Curriculum. ［EB/OL］.https://education-uk.org/documents/des/nc-consultation.html. 1987Department for Education (UK).

Education Reform Act 1988［EB/OL］. https://education-uk.org/documents/acts/1988-education-reform-act.html. 1988

Department for Education (UK). Computing programmes of study: Key stages 1 and 2. 2013.

Simon Peyton-Jones(2014).Computing in the national curriculum - Computing at School.2014

Enhancing Information Technology Teaching Through Data Visualization in Computational Thinking: An Application in Understanding Logic Gate and Half-Adder Circuits

Liu Chuang[1]
[1]Shenzhen Nanshan District Arts and Sciences Experimental School (Group) ,China
Liuchuang618@outlook.com

ABSTRACT
This article explores the application of computational thinking in the teaching of information technology. The example discussed in this article is the use of data visualization to assist in teaching logic gate circuits and binary half-adders. The focus is on using data visualization to reveal the operational units of computers, thereby constructing a clearer learning path for computational thinking. The article analyzes key elements of computational thinking, such as problem decomposition, pattern recognition, and abstract thinking, and discusses their specific applications in understanding circuits. By integrating theory with practice, this article demonstrates how abstract computational concepts can be transformed into intuitive visual forms, thereby enhancing students' learning efficiency and deep understanding. The innovation of this article lies in using data visualization as a bridge to connect the underlying logic of computers with computational thinking, providing a new perspective and method for information technology education.

KEYWORDS
Computational Thinking, Data Visualization, Information Technology Education

1. INTRODUCTION
While computational thinking originates from computer science and represents the evolution of thought processes in human-computer interaction, this article focuses more on exploring the connection between computational thinking and the underlying logic of computers. This exploration not only prompts us to deeply consider the definitions and relationships between the underlying logic of computers and computational thinking, but also guides us in understanding how these two interact with each other.

Computational thinking is considered a key skill for solving complex problems, encompassing creative thinking, problem decomposition, pattern recognition, conceptual abstraction, and effective algorithm design. However, these abstract descriptions of computational thinking have not yet provided a clear learning path. To explore a definitive learning path, this study introduces data visualization as a tool, exploring how computational thinking aids in teaching students to understand the role of logic gates in computer chips. Presenting a specific case, we use graphical methods to understand logic gate circuits. By utilizing data visualization to delve into the underlying logic of computers, our aim is to use computational thinking to solve and understand deep-seated problems.

2. COMPUTATIONAL THINKING FOR BINARY CIRCUITS
In this article, we will discuss the application of computational thinking in information technology education, particularly in understanding binary half-adder design in logic gate circuits. The core of computational thinking is to provide a novel perspective for problem-solving. In understanding logic gate circuits, we focus not only on the solution itself but more importantly on understanding the process and methodology of problem-solving.

Firstly, computational thinking emphasizes decomposing complex problems into smaller parts for better management and resolution. This approach is akin to constructing the most basic foundation of a computer, where the construction of complex computer systems also utilizes this method of breaking down into smaller problems. This way of thinking helps us deeply understand the working principles of logic gate circuit systems, making them more controllable.

Secondly, the pattern recognition capability in computational thinking enables us to identify key elements in logic gate circuit design, thus constructing complex electronic systems more efficiently. Human learning is based on pattern recognition, which gives us the ability to generalize learning, the ability to apply a principle in various contexts. Abstraction is another key element of computational thinking, allowing us to simplify and symbolize the physical connections and specific circuits of the real world, focusing on function and design at a higher level, reducing information overload. The abstraction and pattern recognition capabilities of computational thinking enable us to understand complex systems from a macro perspective, not constrained by the details of specific implementations.

Finally, algorithm design holds a central place in computational thinking, guiding us to deeply understand the working principles of circuits and translate these principles into practical, operable design solutions. Computational thinking provides us with powerful tools for understanding and optimizing electronic circuits. It is foreseeable that computational thinking, as a tool, is not only applicable to the understanding of logic gates and binary adders but also helps in solving practical technical problems, enhancing innovation, and systemic thinking abilities.

3. DATA VISUALIZATION FOR GRASPING CIRCUIT DESIGN CONCEPTS
Data visualization plays a crucial role in computational

thinking. Integrating it with information technology education, particularly in logic gates and half-adders, facilitates students' understanding of complex electronic engineering concepts. This tool can transform abstract concepts into intuitive, easy-to-understand visual forms, thereby promoting deeper understanding and learning. In the process of learning circuit design, textual and oral descriptions often fail to fully convey complex structures, whereas data visualization, through simplified graphics and charts, makes elusive circuit designs clear. It not only enhances understanding through visual transformation and pattern replacement but also presents circuit knowledge in multi-angle and multi-dimensional ways, helping students and engineers grasp the core elements of design, leading to precise and efficient decision-making.

Data visualization also promotes key computational thinking skills, such as pattern recognition and abstract thinking. Through graphical representation, designers can more easily identify repetitive patterns and key connections in circuits, which is crucial for constructing and optimizing complex circuits. Abstraction skills are also enhanced, as visual tools provide different levels of circuit presentation, from flat to three-dimensional, helping students understand circuit layouts from a macro perspective. This complex form of data visualization not only makes information hierarchical but also closer to the essence of the real world.

In education or research, data visualization makes the teaching of computational thinking more efficient and interactive. It not only enhances students' learning interest but also provides a platform for exploration and experimentation. In practical engineering projects, data visualization is crucial for communication and collaboration among student team members, especially in complex projects requiring interdisciplinary cooperation. Overall, data visualization is key in connecting theory with practice, simplifying complex concepts, and promoting in-depth understanding. It not only enhances the application of computational thinking but also brings revolutionary changes to electronic engineering education and practice. By combining computational thinking and data visualization, we can more effectively understand and innovate in electronic engineering design, preparing for future technological challenges.

4. VISUALIZING LOGIC GATES THROUGH COMPUTATIONAL THINKING

In the field of information technology education, applying computational thinking to the illustrative teaching of logic gate circuits and binary half-adders is an effective method to enhance students' understanding and application of digital logic. Logic gates are the foundation of digital circuit design, controlling signal flow through the execution of basic Boolean operations. The binary half-adder, as a basic arithmetic circuit used for adding binary numbers, serves as the starting point for understanding more complex operations.

In the teaching of these circuits, using computational thinking to decipher their working principles is crucial. For instance, when explaining the design of a half-adder, it's possible to show how the addition of binary numbers can be achieved through the combination of logic gates. This involves not only understanding the function of each logic gate but also how they can be combined to construct complex circuit systems.

Particularly in the process of translating the physical implementation of circuits into theoretical models, the pattern recognition and abstraction abilities of computational thinking become especially important. Data visualization technology plays a key role in this process, as it can transform abstract circuit concepts into intuitive graphics, making complex circuit designs more accessible and analyzable for students. Such visualization tools not only help students clearly see each part of the circuit and their interrelationships but also assist them in identifying potential problems in the design before actual implementation.

4.1. Logical Thinking

Utilizing the mindset of computational thinking, we will now adopt an illustrative approach to explain the working principles of the logic gate half-adder. Guided by this way of thinking, the first thing we need to apply is abstract thinking. In computational thinking, abstract thinking requires us to first abstract specific objects. In the case of the relay, the basic building unit of logic gates, its image is the component that needs to be abstracted. We can use simple sketching to abstract it. This prompts our thinking to move away from specific physical objects and familiar scenarios, identifying the general characteristics of things. The most important feature of a relay is that it generates a magnetic field when activated by a switch. This diagram can effectively illustrate this phenomenon. Such abstraction of real-world objects compresses information, allowing us to think more swiftly. The relay's ability to use electromagnetic fields to control switches at a distance creates incredibly versatile operational methods.

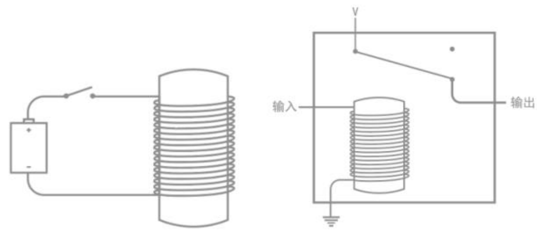


Figure 1. Relay magnetic field control switch

Under the general principles of the relay, we have abstracted it into graphic symbols. This further leads us to consider the different outcomes brought about by the differences in connection states, specifically the upper and lower states.

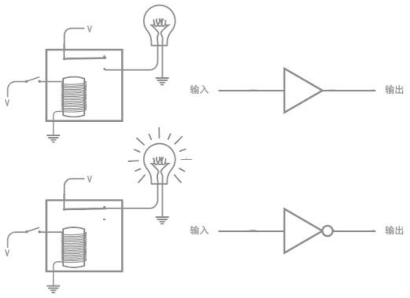


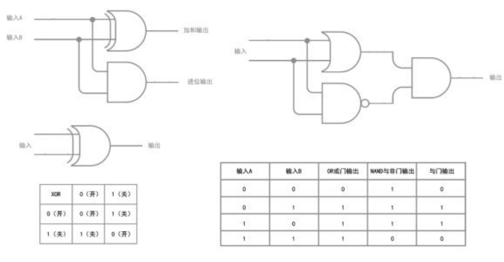
Figure 2. Relay combination abstracted as a buffer

4.2. Pattern Recognition

In the process of abstracting and extracting features from real things, which is actually the main way that artificial intelligence learns today, we can explore it from a different perspective, that is, pattern recognition. In our relay circuit switch, we actually generalize and extend the switching pattern, in which we can find the pattern is the magnetic field generated by the relay has a remote control effect on the switch, which is what we further think and learn. This is the basis of logic gate circuits, and a very important way of thinking for IT to teach students.

4.3. Problem Breakdown

Problem decomposition as a way of thinking plays an important role in understanding and constructing logic gates and half adder circuits. First, we need to understand the basic circuit components, such as the and gate (AND), the or gate (OR), the not gate (NOT), and the and non gate (NAND). By combining these basic components, we can build more complex circuit functions. For example, when building a half adder, we typically use a combination of an AND gate and a different-or gate (XOR). The different-or gate is used to generate the sum bit of the addition result, while the and gate is used to generate the rounding signal. By understanding and applying these components, we can gradually build complex circuit systems. Each component carries out a specific logic function and in their interaction they produce the desired end result. Therefore, breaking down a complex circuit design into smaller, easy-to-understand parts is a critical step in understanding and realizing a circuit design.



XOR	0（开）	1（关）
0（开）	0（开）	1（关）
1（关）	1（关）	0（开）

输入A	输入B	OR或门输出	NAND与非门输出	与门输出
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

Figure 3. Half adder logic gate circuit

4.4. Algorithmic Thinking

In teaching information technology, by visualizing data for basic Boolean logic operations such as and, or, and not, we provide students with a clear path to understanding the fundamentals of designing logic gates. When teaching how to design half-adders, the application of algorithmic thinking can help students to systematically understand how different logic gates are combined and their role in implementing binary addition. The way computational thinking visualizes knowledge not only improves students' problem-solving skills, but also enhances their deeper understanding of circuit design. By teaching students how to improve circuits using an iterative approach, we can further develop their critical thinking and innovation skills. The concept or modular design enables students to understand the construction and function of complex circuits more effectively. Breaking down complex circuits into modular parts and then integrating these modules to achieve more complex functionality not only simplifies the learning process, but also helps students construct an understanding of the whole and the details

5. SUMMARIZE

In computational thinking, the use of graphical for visualization actually reflects the ability to think abstractly about realistic scenarios and concrete objects. By applying this kind of thinking to the design of logic gate circuits, we not only explain computational thinking more deeply, but also provide clearer guidelines for learning paths. As an important part of computational thinking, thinking about the underlying logic of computers deepens our understanding of the nature of computational thinking. The innovation of this paper is that it attempts to explore the connection between underlying computer logic and computational thinking through data visualization. Modularization and in-depth teaching of computational thinking are crucial for information technology teachers when building the foundational elements of complex systems, which helps students develop a solid mindset for the subject.

6. REFERENCE

Scott, C. (2009). *But How Do It Know?*. John C. Scott. Retrieved from https://www.perlego.com/book/3289465/but-how-do-it-know-the-basic-principles-of-computers-for-everyone-pdf (Original work published 2009)

Hurt T, Greenwald E, Allan S, et al. *The computational thinking for science (CT-S) framework: operationalizing CT-S for K–12 science education researchers and educators*[J]. International Journal of STEM Education, 2023, 10(1): 1-16.

Petzold C. *Code: The hidden language of computer hardware and software*[M]. Microsoft Press, 200

Research on Project-Based Teaching Mode of Information Technology Curriculum for Cultivating Computational Thinking in Primary School Students

Qin DENG[1], Xinmin FAN[2*]
[1,2]Fujian Normal University, China/Fujian
1342323097@qq.com, xmfan@fjnu.edu.cn.

ABSTRACT
It is widely accepted that primary education is aimed to help children learn basic skills and build a solid foundation for their lifelong learning. Computational Thinking (CT) is a problem-solving approach that serves as a bridge between the creativity of the human brain and the efficiency of computational power. However, it can often be difficult for teachers to lead classes on computing science topics because they lack knowledge or experience of coding and the cognitive learning styles of young learners differ from those of adults. Project-based learning(PBL) methods have been proved to be effective in fostering students' computational thinking. In this paper, we propose a three-stage framework based on PBL methods: 1)pre-class preparation. We conduct a comprehensive assessment of students' learning styles and interests so as to group them accordingly. 2)in-class implementation. We encourage students to explore the driving question by engaging in scientific practices that intersect with CT by project selecting, decomposition, algorithm design, program debugging and evaluation. 3)post-class evaluation. we use cognitive tools to assist teacher and students in reviewing their learning process. We test our framework in the programming class of sixth-grade students and the results clearly demonstrate the effectiveness of our model on CT skills teaching.

KEYWORDS
computational thinking, project-based learning, primary school students, teaching mode

1. INTRODUCTION
The emerging technologies have a significant impact on our lives, presenting them with challenges and uncertainties. However, one thing remains certain: students must learn to think critically and resolve complex, ill-defined problems (Wing, 2006). Computational thinking (CT) is a core competency that assists young students in developing such problem-solving skills. The skills, attitudes, and approaches comprising CT are fundamental, universal, and transferable, leading to its inclusion in primary school information technology courses.

However, pupils can often be too young to understand abstract concepts and engage in reasoning in formal educational settings. Like any skill, CT is best taught and learned in context, and integrated into class subjects. Project-based learning (PBL) is an effective method that allows students to work on meaningful projects and develop advanced thinking skills.

In this paper, we integrate PBL into hybrid learning and propose a three-stage framework for IT course teaching:

1. Pre-class preparation: We use computer adaptive tests to assess children's learning styles, interests, and knowledge levels. Students are grouped into different teams based on their backgrounds.

2. In-class implementation: We encourage students to engage with various practices and big ideas around them. They are required to select a project based on their observation, decompose it into manageable parts, design algorithms for solving similar problems, find and fix error, and test their final solution for effectiveness.

3. Post-class reviewing: We use thinking tools to review the learning process so teachers can refine their teaching and students can find creative inspiration for next time.

Our framework was carried out by 99 sixth-grade students from two classes of the IT course. We conducted three experiments and demonstrated that our framework can deepen students' understanding of coding.

2. MODEL CONSTRUCTION
As Figure 1 shown, our framework consists four core elements: content, activities, context, and outcomes, and three stages: pre-class preparation, in-class project implementation and post-class reviewing. It integrates the cultivation objectives across the five dimensions of computational thinking abilities.

2.1. Pre-class Project Preparation Stage:
Before class, teachers upload micro-lessons and syllabus to the learning platform for students to assess themselves. Based on the test results, teachers can identify students' learning style and skill gaps, therefore grouping them into different teams.

2.2. In-class Project Implementation Stage:

2.2.1. Project selection
It is necessary to consider the pupils' cognitive levels, design project related to real life, and help students achieve knowledge transfer.

2.2.2. Project decomposition
Teachers guide students to decompose complex projects into smaller tasks, reducing project difficulty. By applying recent acquired knowledge to solve real-world problems, students gradually develop computational thinking.

2.2.3. Algorithm design
Group collaboration exploration involves drawing flowcharts to guide students in presenting problem-solving steps graphically. By illustrating the logical structure of algorithms through flowchart diagrams, it enhances students' comprehension and application of learned knowledge.

2.2.4. Programming debugging

Students use Scratch programming software for practical operations. Based on the drawn flowcharts, they complete coding and, through debugging and modification, achieve the goal of optimizing the program.

2.2.5. *Summative evaluation*

Students present their group projects works to their peers and received constructive feedbacks. Teachers summarize and evaluate, proposing optimal solutions.

2.3. *Post-class Project Reviewing Stage:*

Students continue to refine and innovate based on the teacher's feedback and upload their work to the learning platform. Teachers assess and grade the work uploaded by students, while also analyzing the data from the entire teaching process. Based on the results of the data analysis, teachers adjust and optimize the teaching design.

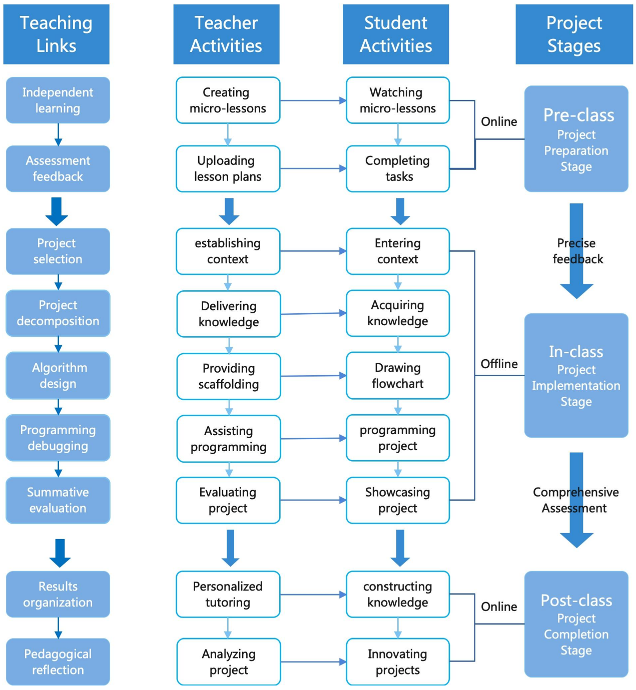


Figure 1. Project-Based Teaching Model for Fostering Computational Thinking in Primary Schools

3. MEASUREMENT TOOL

The study employs a computational thinking scale developed by Zhang Yi et al. (2020) to assess changes in students' computational thinking pre- and post-project-based learning. The scale, validated through small-sample testing with a reliability coefficient of 0.9 and a validity coefficient of 0.72, is deemed suitable for elementary school research. To enhance measurement accuracy and mitigate potential subjectivity inherent in the scale, supplementary measures in the form of Bebras test questions are integrated. Furthermore, process data pertaining to students' utilization of computational thinking in problem-solving are assessed through both flowchart evaluations and project evaluation, leveraging a combination of quantitative and qualitative evaluation techniques. Flowchart assessments focus on the dimensions of completeness, rationality, and innovation, guided by scoring criteria established in Dr. Su Qing's (2022) thesis. Project evaluation criteria are drawn from Dr. Scratch's recommendations, encompassing abstract problem-solving, parallelism, synchronization, sequence, logical thinking, user interaction, and data representation.

4. MODEL EFFECTIVENESS TEST

4.1. *Experimental Subjects and design*

The quasi-experimental study was conducted from October to December in the first semester of the 2023-2024 academic year. Two sixth-grade classes from a primary school in Fuzhou City, Fujian Province, were selected for the experiment. A pre-test was administered using the Computational Thinking Scale, and the results indicated homogeneity of variance across all five dimensions(sig>0.05), suggesting the two classes formed a homogeneous sample.

The project conducted in this study was based on three main themes of self-designed projects focusing on Scratch knowledge points from the first unit of the sixth-grade information technology textbook in Fujian. These projects were closely related to real-life situations. In the control group, traditional teaching methods involving teacher demonstrations and student imitation were employed, while the experimental group utilized a project-based teaching model designed to cultivate computational thinking skills.

4.2. *Experimental Data and Analysis*

4.2.1. *Computational Thinking Scale*

The data from the Computational Thinking Scale for the experimental and control groups were subjected to independent samples t-tests using SPSS 27.0 software (refer to Table 1). The research findings indicate that students in the experimental group demonstrated improvements across all five dimensions of computational thinking compared to the control group. Particularly noteworthy are the significant differences observed in critical thinking, collaborative ability, and algorithmic thinking.

Table 1. Comparative Analysis of Differences in Five Dimensions of Computational Thinking Scale

Dimension	Experimental (n=48) M±SD	Control (n=51) M±SD	*t*	*p*
Creativity	3.86±0.40	3.68±0.41	2.174	.032*
Critical thinking	4.21±0.56	3.66±0.69	4.290	<.001***
Problem solving	3.68±0.49	3.45±0.48	2.319	.022*
Algoritmic thinking	3.44±0.52	3.02±0.80	3.001	.003**
Cooperativity	3.43±0.57	3.36±0.82	6.742	<.001***

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

4.2.2. *Bebras Test Questions*

Before the experiment, the experimental group scored 33.229, and after the experiment, the score increased to 36.875. A paired-sample t-test was conducted on the five dimensions of the test before and after the experiment, revealing statistically significant changes in three dimensions (Refer to Table 2). Specifically, these dimensions include analytical thinking, algorithmic thinking, and evaluative thinking, with particularly significant differences observed in algorithmic thinking.

Table 2. Pre-and Post-Test Comparison of the Five Dimensions in the Experimental Class

Dimension	Pre-test (n=48) M±SD	Post-test (n=48) M±SD	*t*	*p*
Abstract	3.59±1.35	3.28±1.37	1.000	.322
Analytical	3.12±1.74	3.80±1.45	2.223	.031*
Algoritmic	2.29±1.77	3.75±1.63	4.639	<.001***
Synthetic	3.69±1.26	3.85±1.45	.535	.595
Evaluative	3.38±1.58	3.90±1.35	2.114	.040*

p<0.05，p<0.01，***p<0.001*

4.2.3. Flowchart Assessments

The average scores for three project program flowcharts in the experimental group were 6.33, 7.29, and 8.67, while the control group scored 6.21, 6.66, and 7.04 for the same three projects. It can be concluded that the performance of the experimental group students in utilizing program flowcharts to express their computational thinking gradually improved. A paired-sample t-test was conducted on the program flowchart scores before and after the experiment, resulting in $p<0.001$. This indicates a significant statistical difference in the improvement of program flowchart drawing scores for the experimental group students, suggesting that these students are proficient in using flowcharts as a tool for expressing their problem-solving strategies.

4.2.4. Project evaluation

The experimental group achieved average scores of 9.3, 12.58, and 15.33, whereas the control group scored 8.92, 10.13, and 11.94 for the same projects. The scores of the experimental group's works exhibited a significant improvement compared to those of the control group. A paired-sample t-test conducted on the scores of the works before and after the experiment yielded a result of $p < 0.001$, indicating that the students in the experimental group are able to integrate previously learned instructions, flexibly apply them to new projects, and refine and optimize their projects.

5. CONCLUSION

5.1. The research conclusion

We utilized various methods, including the Computational Thinking Scale, Bebras Test Questions, flowchart, and work analysis, to assess how our framework impact on elementary students' computational thinking. Results show there is a significant improvement in not only overall computational thinking. Detailed examination of flowcharts and works reveals clearer logical reasoning and richer, more complex outputs.

5.2. Practical recommendations

1. Conduct detailed data analysis before and after class. Before class, utilize the online learning platform to assess students, and lay the foundation for following project implementation. After class, analyze students' works and process data to conduct treflections.

2. Design projects with authentic real-world relevance. Craft teaching scenarios that align project contexts with students' practical experiences, fostering engagement and resonance with real-life contexts. This approach stimulates interest, facilitates connections with practical experiences, and encourages active cognitive engagement and exploration.

3. Leverage visual thinking tools to facilitate cognitive organization. Flowcharts, as visual representations, aid in comprehension of abstract algorithms, streamlining the learning process for students and providing educators with valuable insights for targeted instruction.

6. REFERENCES

ISTE. (2024). Computational thinking leadership toolkit. Retrieved March 2nd, 2024, from https://id.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4

Ministry of Education. (2022). *Compulsory Education Information Technology Curriculum Standards* (2022 ed.). Beijing: Beijing Normal University Press.

Su, Q., Zhang, W. L., & Wang, H. (2022). Research on Project-based Learning of Information Technology Curriculum for Cultivating Senior High School Students' Computational Thinking. *Research on Educational Technology, 43* (8), 109.

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

Zhang, Y., Mo, W., & Zhang, Y. (2020). Development and Application of Chinese Pupils' Computational Thinking Scale. *China Digital Education*, (10), 49-57.

A Study of Micro: bit Instructional Design Based on CS Unplugged and Use-Modify-Create

WenYuan Mu[1], XiaoJun Xia[2*], Yi Bai[3*]
[1,2,3]Key Laboratory of China Development and Learning Science, China
Southeast University, Nanjing
mwy18203@163.com, xxj.rcls@seu.edu.com, by.rcls@seu.edu.com

ABSTRACT

Emerging as a critical skill in contemporary society, computational thinking represents a product of the new era. Its gradual integration into K-12 education necessitates a transformation of traditional teaching methods. This paper deviates from the conventional classroom teaching model by amalgamating theoretical learning with hands-on practice and formulating an innovative lesson plan. The plan comprises two sessions: cs unplugged activities designed to teach students basic concepts, and plugged activities using "Use-Modify-Create" learning progression in a micro: bit device. The aim is to enhance students' computational thinking skills and serve as a reference for educators in fostering computational thinking development among their students.

KEYWORDS

Computational thinking, CS Unplugged, Micro: bit, Use-Modify-Create, Instructional Design

1. INTRODUCTION

Wing (2006) defined computational thinking as the skill of solving problems, designing systems, and understanding human behavior based on the concepts fundamental to computer science. Computational thinking (CT) is an important set of skills that human beings utilized for problem-solving despite the rapid changes in technology (Denning, 2016). While a unified definition of computational thinking has not yet been established within the academic community, different definitions for computational thinking often include concepts and practices such as abstraction, decomposition, automation, algorithmic thinking, and modeling and simulation.

In recent years, some research on CT has mostly focused on integrating CT into mathematics (Ye et al., 2023) and the K-12 curriculum (e.g., Sengupta et al., 2013; Yadav et al., 2016), and developing programming environments and tools to promote CT skills (e.g., Grover & Pea, 2013; Lye & Koh, 2014). To reach all students, it is necessary to integrate CT into the required k-12 education setting. However, traditional teaching mainly focuses on theoretical explanations and lacks practical application. This approach is not conducive to students fully mastering computational thinking. To address this issue, alternative teaching methods should be explored.

Thus, This paper breaks through the traditional teaching methods and designs a lesson design based on the combination of theoretical learning and practical application. The lesson plan is designed in two sessions and employs two strategies to teach students computational thinking. Begin with an unplugged learning activity that introduces the basic concepts. Continue with computational activities using the Makecode editor and Micro: bit devices following the steps of Use-Modify-Create. It is hoped that this can provide reference for teachers to carry out calculation activities in the future.

2. THEORETICAL FRAMEWORK

2.1. *CS Unplugged Activities*

Unplugged Activities (UA) is a set of learning activities designed to explain CS concepts without needing a computer device. Unplugged is particularly aimed at developing a fundamental understanding of concepts about algorithm design, and to change conceptions about the nature of computer science (Caeli & Yadav, 2020). The main characteristics of the Unplugged activities are(Nishida, 2009): (1) No programming ability or computer is required as a prerequisite for studying computer science. (2) Students actively engage in various activities, fostering peer cooperation and promoting collaboration and communication among their peers. (3) These activities are easily prepared with low-cost implications.

In addition, the integration of computational thinking into the K-12 curricula, especially in economically underdeveloped developing countries, often poses challenges for schools and students in accessing information and communications technologies. In our country, many rural areas lack facilities and have not yet popularized computer education, unplugged learning activities can address this challenge and scaffold student learning.

2.2. *Micro: bit*

The BBC micro: bit was developed in 2015 as part of the BBC's Make it Digital Initiative.

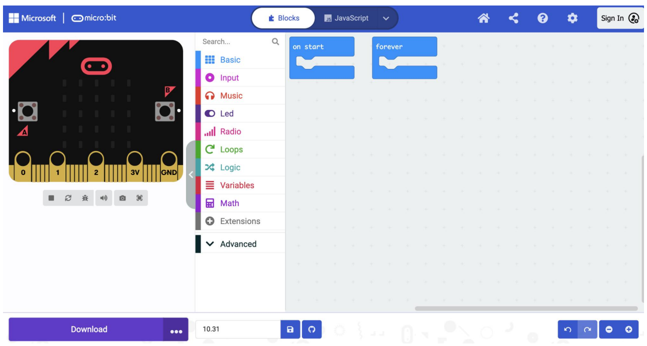


Figure 1. The MakeCode editor for the micro: bit (https://makecode.microbit.org).

It aims to encourage digital literacy and core skills in STEM subjects among young people. The micro: bit is a

32-bit ARM microcomputer that includes a built-in display, buttons, motion detection, temperature and light sensors, and supports Bluetooth low-power wireless communication (Ball et al., 2016). The graphical programming interface is user-friendly, particularly for younger students. The project supports both online and local integrated development environment programming for mainstream languages such as Python and Javascript.

2.3. *Use-Modify-Create*

The Use-Modify-Create progression has been widely advocated as a strategy to facilitate student engagement in CT. It has been utilized to provide learners with a comprehensive and progressive experience in computational thinking within an immersive computing environment. The instructor provides a pre-designed program for students to explore (Use), and students then modify programs developed by others (such as teachers), personalizing the code to their needs (Modify). Finally, students generate(Create)their programs independently through a series of rigorous testing procedures, careful analysis, and repeated improvement (Figure 2). In this process, it is important to maintain a level of challenge and difficulty to support growth while limiting frustration and anxiety, considering that the approach is not completely linear, switching back and forth from use to modification to creation (Lee et al., 2011).

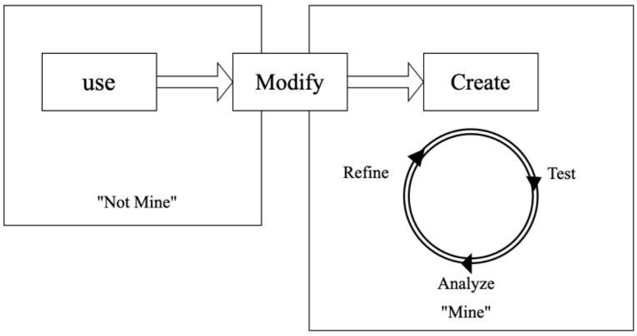


Figure 2. Use-Modify-Create model

(adapted from Lee et al., 2011).

3. INSTRUCTIONAL DESIGN

3.1. *Program Design*

The instructional design comprises two sessions (Figure 3). The first session of the unplugged learning activity involves the teacher discussing the learning objectives and the prior knowledge required to complete the activity. This section enables students to manage their expectations, make connections with previous knowledge, and reflect on it at the end to facilitate the metacognitive process. The second session involves plugged activities where students develop their program using a 'use-modify-create' strategy on micro: bit. Initially, students explore a sample program that predicts the outcome of the lesson or explains it to a partner (i.e., Use). They then make specific modifications to the example program and participate in scaffolding problem-solving activities (i.e., Modify). Finally, students will apply what they have learned from the example by using the Micro: bit to create a solution from scratch to solve the the problem (i.e., Create).

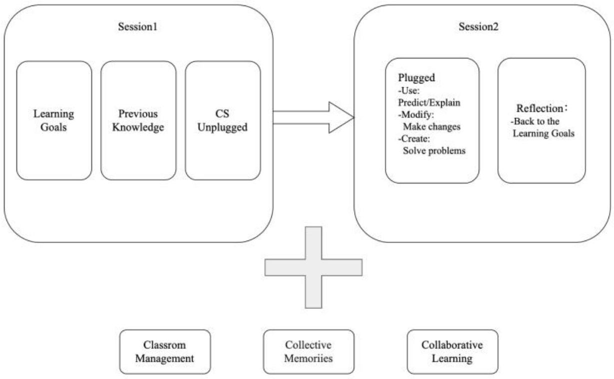


Figure 3. Structure of the Program design.

3.2. *Example Module: Autopilot Simulation in Caves*

Table 1. Key Contents of the Task in Session 1.

Learning Task	Learning Goals	Computational Thinking Goals
Autopilot simulation in caves	Use boolean input variables, logical operations, and loop structure to finish the task	Understanding of abstraction, decomposition, and algorithms

This paper presents a pedagogical example of a course that simulates an autonomous vehicle exploring a cavern scientifically. Table 1 illustrates the pedagogical and computational thinking objectives to be achieved in the first session of the activity. Through the unplugged activity, students will acquire fundamental programming concepts and methods, providing them with the necessary previous knowledge for the second session of the practical exercise. Later, students will need to program using Micro: bit input variables such as buttons and LED displays.problem (i.e., Create).

Use: Predict what the program is doing.

Modify: Extend the program to move in all possible directions(i.e., East, West, South, North) based on the buttons pressed.

Create: Use the micro: bit compass for the automated line following towards the North.

Figure 4. Sample task of the Use-Modify-Create progression in Session 2.

Figure 4 illustrates the process of Use-Modify-Create in the second session of the teaching case, which uses a plugged activity in which the students use the micro: bit with two buttons to simulate proximity sensors and a micro: bit display that will show the vehicle's route. Students first need to predict what the example program is doing and run the program to verify the prediction (i.e., Use). The program recognizes a pressed button and displays an arrow pointing north. The next step is to modify the program so that it shows all directions on the display based on the button pressed (i.e., Modify). Finally, the student uses the compass sensor to create a program that guides the vehicle in one direction (e.g., North) (i.e., Create). Both the "Modify" and "Create" steps in the process require students to perform, test, reflect, and improve repeatedly, often in discussion with their peers.

Unplugged activities and the Use-Modify-Create progression can help students form early schemas about problems and algorithms, which in turn can help them improve their computational thinking skills and reframe their internal thinking. At the end of the second session, the lesson plan includes an additional element: reflection - linked to the learning objectives. Sequencing activities in a learning design may help students achieve self-accomplishment (Katai, 2020) and motivation to learn, as they can see their progress through the outputs of each activity.

4. CONCLUSION

This study presents a lesson plan designed to enhance students' computational thinking skills in K-12 education. This lesson plan comprises two sessions. The first session includes a series of unplugged learning activities to prepare students for the second part. In the second session, students follow a step-by-step process of 'Use-Modify-Create' on a Micro: bit board to facilitate their learning.

This instructional design combines theoretical learning with practical application, breaking away from the traditional single-teaching mode. It makes learning more interesting for students and provides teachers with a reference for carrying out Micro: bit teaching practice. Additionally, this study offers new ideas for cultivating computational thinking, which can aid in the innovation of classroom teaching in the K-12 stage. It has become an unstoppable trend to cultivate computational thinking skills in K-12. Computational thinking is not only traditional programming thinking but also literacy and ability that all people need to have in the future. As research continues, it is hoped that new methods and technologies will be developed to foster computational thinking.

5. REFERENCES

Ball, T., Protzenko, J., Bishop, J., Moskal, M., De Halleux, J., Braun, M., ... & Riley, C. (2016, May). Microsoft touch develop and the BBC micro: bit. In Proceedings of the 38th International Conference on Software Engineering Companion (pp. 637-640).

Bodin, Ö., Alexander, S. M., Baggio, J., Barnes, M. L., Berardo, R., Cumming, G. S., ... & Sayles, J. S. (2019). Improving network approaches to the study of complex social–ecological interdependencies. Nature sustainability, 2(7), 551-559.

Caeli, E. N., & Yadav, A. (2020). Unplugged approaches to computational thinking: A historical perspective. TechTrends, 64(1), 29-36.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational researcher, 42(1), 38-43.

Katai, Z. (2020). Promoting computational thinking of both sciences-and humanities-oriented students: an instructional and motivational design perspective. Educational Technology Research and Development, 68, 2239-2261.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32-37.

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.

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. ACM Sigcse Bulletin, 41(1), 231-235.

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, 351-380.

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In Proceedings of the 16th Koli Calling international conference on computing education research (pp. 120-129).

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

Wing, J. M. (2016). Computational thinking, 10 years later.http://www.microsoft.com/enus/research/blog/computational-thinking-10-years-later.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. Computer science education, 26(4), 235-254.

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning. International Journal of STEM Education, 10(1),

Enhancing Computational Thinking in Knowledge Building Community: Analyzing ChatGPT's Role and Impact Among Undergraduates

Zhenhai He[1], Shaoming Chai[2*], Fuying Zhou[3], Han Wu[4], Nannan Lai[5], Tongwu jiang[6]
[1,3,4,5,6]School of Information Technology in Education, South China Normal University, China
[2*]Aberdeen Institute of Data Science and Artificial Intelligence, South China Normal University, China
zhenhai@m.scnu.edu.cn, charmingchai@m.scnu.edu.cn, 2022020961@m.scnu.edu.cn, 2022020938@m.scnu.edu.cn, 2023020849@m.scnu.edu.cn, 2023020848@m.scnu.edu.cn

ABSTRACT

The swift advancement of generative AI, notably ChatGPT, has markedly influenced education. Despite initial studies into its educational applications, the capacity of ChatGPT to boost computational thinking (CT) in undergraduates remains an area for further investigation. Therefore, this study investigates how undergraduates in CT courses utilize ChatGPT and its effects on their skill development. We employed text analysis and social network analysis methods to investigate the discourses among 48 sophomore and junior students from various disciplines, who were enrolled in a *Computational Thinking and Problem-Solving* course conducted within the Knowledge Forum. These students were organized into seven distinct groups for this course. Results indicate that students frequently engaged with ChatGPT, critically evaluating its responses with teacher-provided cognitive scaffolds instead of merely copying them. Groups that utilized ChatGPT more extensively improved CT skills, evidenced by their focus on critical CT terms and tighter linkage between discussion content and these terms. This study demonstrates that judicious ChatGPT use in higher education can effectively enhance CT skills, offering insightful case studies for generative AI's educational applications and providing effective guidance for improving students' CT with ChatGPT in higher education.

KEYWORDS

Computational Thinking, Knowledge Forum, ChatGPT, Higher education

1. INTRODUCTION

The rapid advancement of generative artificial intelligence, with large language models like OpenAI's ChatGPT and Anthropic's Claude at the forefront, is reshaping educational methodologies. These tools offer novel conveniences and support for both students and teachers, facilitating the creation of tailored teaching materials and interactive learning experiences (Kasneci et al., 2023; Lee, 2023). Ethically employed, ChatGPT can foster a conducive learning environment, enhancing deep learning and outcomes. Effective integration of ChatGPT into education requires consideration of leadership, personality development, and authenticity assessments (Crawford et al., 2023). In academic settings, such as anatomy courses, ChatGPT shows promise as both a teaching and assessment tool (Talan & Kalinkara, 2023). However, its widespread use poses challenges to traditional educational methodologies, historically validated for their effectiveness (U.S. Department of Education, 2023).

In the digital era, Computational Thinking (CT) is crucial for solving problems across disciplines, enabling individuals to analyze challenges with the precision of computer scientists (Wing, 2006; Güven & Gulbahar, 2020; Tekdal, 2021). Despite the interest in weaving CT into higher education, its definitions and application strategies still demand further exploration (Lyon & J. Magana, 2020). CT's importance transcends STEM, underscoring the necessity for its widespread adoption across all academic fields (Czerkawski & Lyman, 2015). Studies using the knowledge-building approach to foster CT skills in higher education have shown promising results, offering educators valuable strategies for curriculum design that enhance students' CT abilities and identify areas for improvement (Zhu et al., 2023). Celik (2023) posits that CT proficiency can improve individuals' comprehension and recognition of AI systems. Although the contribution of AI to CT enhancement is recognized, there is a significant gap in research regarding ChatGPT's role in nurturing CT skills in undergraduate education. Therefore, this study primarily investigates the following questions: (1) To what extent do students use ChatGPT in CT courses? (2) What is the relationship between the extent of ChatGPT use by different groups and the development of CT among group members?

2. METHODOLOGY

In this study, we employed text and social network analysis methods to code students' Notes to explore the extent of ChatGPT usage among students. We used KBDeX to analyze the degree of centrality of discourse and keywords in the knowledge forum to assess the development of students' CT.

2.1. Participant

The participants of this study were 48 undergraduate students (17 females) enrolled in a course named Computational Thinking and Problem Solving during the fall semester of 2023 at a public university in China. This 14-week general education course aimed to develop students understanding of the concept, nature, and essential elements of CT, enhancing their ability to solve complex problems in their professional and interdisciplinary fields through exploring algorithms, constructing models, and designing programs. The students had diverse program backgrounds, including software engineering, artificial intelligence, electronic information engineering, cultural industry management, and financial management, covering science and liberal arts disciplines. An instructor with a background in educational technology and rich teaching experience taught the course.

2.2. *Course design and implementation*

Our course adopted constructivism and situated learning theories to cultivate CT in a context-rich setting. Utilizing knowledge-building pedagogy and cognitive scaffolds within the Knowledge Forum, we aimed to advance students' higher-order thinking and analytical interaction with ChatGPT. This strategy intends to transcribe simple query resolution, promoting critical scrutiny and profound engagement with CT.

Guided by these pedagogical frameworks, the curriculum was designed in four evolutionary stages to foster CT by profoundly engaging with ChatGPT and applying critical course theories. Initially, students were organized into seven cross-disciplinary teams, engaging in forum discussions to build a basic comprehension of CT. The following stages involved team-based problem-solving, starting from algorithmic exercises like the shortest path problem and advancing to complex AI case analyses, where ChatGPT served as both a tool and a virtual team member. The final stage required students to apply CT principles to real-world challenges, leading to the development of tangible projects. Assessment across these stages was conducted through peer and instructor reviews, emphasizing the extent of CT application in the teams' deliverables.

2.3. *Data collection and analysis*

Data for this study was obtained from 415 notes collated from group activities on the Knowledge Forum platform. In evaluating ChatGPT's utilization by students, two researchers initially reviewed the notes for content categorization, subsequently dividing them into five distinct types, as depicted in Table 1. This categorization facilitated the assessment of ChatGPT's role in the collaborative learning process. To ensure reliability, the researchers independently coded a notes sample, achieving an intercoder agreement of 96%. The proportion of ChatGPT-related notes to the total was a quantitative measure of the tool's use. Additionally, we applied KBDeX social network analysis to examine the centrality of discourse and CT keywords within the discussions, underpinning our analysis with established CT frameworks. This method quantified the prominence of CT concepts in dialogues and their correlation with group engagement with ChatGPT.

Table 1. Coding framework for student ChatGPT usage

Code	Targets
SI	Students share ideas without turning to ChatGPT
PC	Copy and paste ChatGPT's outputs without citations
SC	Share ChatGPT's ideas directly by cognitive scaffolding
RC	Critique ChatGPT's responses
IC	Improve ChatGPT's responses

3. RSEULT AND DISCUSSION

3.1. *The use of ChatGPT by students*

After coding, we obtained results as shown in Table 2. To quantify the degree of ChatGPT engagement across different groups, we computed the ratio of ChatGPT-related notes to the total number of notes within each group. This ratio was then expressed as a percentage, serving as an indicator of the relative extent to which each group utilized ChatGPT in their collaborative activities. The results illustrate a discernible variance in the utilization of ChatGPT among the groups. Group 6 emerged as the most engaged, with ChatGPT-related discourse constituting 57% of their total notes, while Group 3 followed closely with 48%. Further analysis revealed that Groups 7 and 1 exhibited substantial interaction, with ChatGPT-themed contributions making up 24% and 23% of their notes, respectively. Conversely, Group 5 displayed a lower degree of engagement at 18%. It is noteworthy that Groups 4 and 2 utilized ChatGPT the least, each incorporating it into merely 12% of their overall communications.

Table 2. The use of ChatGPT by different groups in the course

	Notes_Total	PC	SC	RC	IC
Group1	113	3	4	10	9
Group2	42	0	2	0	3
Group3	42	2	12	2	4
Group4	51	0	0	0	6
Group5	60	3	6	2	0
Group6	56	6	4	8	16
Group7	51	0	2	1	9

3.2. *The impact of groups' usage extent of ChatGPT in their development of CT*

We conduct social network analyses on the discourse and keywords, obtaining the degree of centrality for each group at both the discourse and keyword levels. Degree centrality measures an individual's importance in the collaborative learning discussion network. In social network analysis of discussions or learning, a highly centralized network indicates that a few nodes (possibly individual students or specific keywords) occupy central positions with numerous direct connections, suggesting their significant importance or influence in the discussion or learning process (Oshima et al., 2012). Thus, a relatively high degree of centrality indicates the importance of relevant keywords or discourse in the collaborative learning process. As shown in Figure 1, it is observed that Groups 1, 3, 6, and 7 have a higher level of ChatGPT use, and correspondingly, the degree of centrality of their discourse and keywords is also relatively high. In contrast, Groups 2, 4, and 5 have a lower level of ChatGPT use, and accordingly, their discourse and keyword centrality is also relatively low. This indicates that the extent of ChatGPT use is directly proportional to the development of CT. The groups with higher usage of ChatGPT are more likely to engage in discussions that closely revolve around the core keywords of CT, with tighter connections.

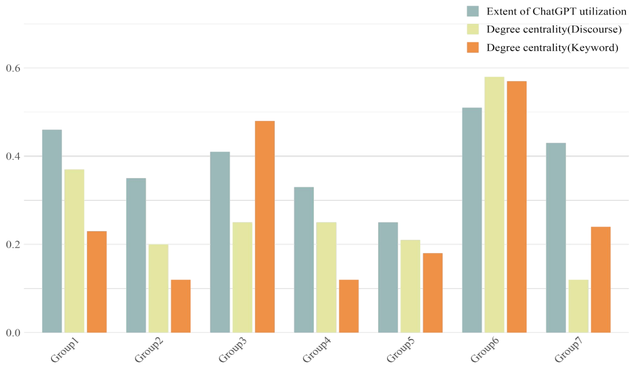


Figure 1. The relationship between the extent of ChatGPT use by groups and the degree centrality of discourse and keywords

4. CONCLUSION

This study investigates the extent to which undergraduates use ChatGPT in CT courses and the impact of this usage on the development of CT within groups. Study shows that: (1) In the CT courses, students exhibited a high level of engagement with ChatGPT, with significant variations in usage across different groups. Students do not merely copy and paste answers from ChatGPT but more frequently utilize the cognitive scaffolding provided by teachers for ChatGPT use to distinguish their ideas from ChatGPT's responses and evaluate ChatGPT's answers based on this distinction. (2) Groups with higher usage of ChatGPT perform better in developing CT. These groups tend to engage in discussions centered around the core keywords of CT, with tighter connections between the discussions and keywords.

This research illustrates the logical and feasible integration of ChatGPT in higher education, showing its potential to elevate students' engagement with ChatGPT and, in turn, enhance their CT abilities. The results provide actionable insights for embedding ChatGPT into academic settings and shed light on the broader implications of AI tools in education. We highlight the imperative for educators to deliberate on effectively incorporating AI technologies into curriculum planning and instructional methods, aiming at the comprehensive development of student competencies. However, there are certain limitations, as the assessment of students' CT skills did not include a pre-and post-test comparison, and the use of social network analysis alone cannot effectively demonstrate the development of students' CT skills. Qualitative analysis should also be integrated to explore the detailed development of students' CT skills. Future research could explore how students apply ChatGPT in CT courses and the relationship between these applications and the development of their CT skills.

5. REFFERENCES

U.S. Department of Education. (2023). Artificial intelligence and the future of teaching and learning, Office of Educational Technology.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, … Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. Learning and Individual Differences, 103, 102274, from https://doi.org/10.1016/j.lindif.2023.102274

Lee, H. (2023). The rise of ChatGPT: Exploring its potential in medical education. Anatomical Sciences Education, ase.2270, from https://doi.org/10.1002/ase.2270

Celik, I. (2023). Exploring the determinants of artificial intelligence (AI) literacy: Digital divide, computational thinking, cognitive absorption. Telematics and Informatics, 83, 102026, from https://doi.org/10.1016/j.tele.2023.102026

Crawford J., Cowling M., & Allen K.-A. (2023). Leadership is needed for ethical ChatGPT: Character, assessment, and learning using artificial intelligence (AI). Journal of University Teaching and Learning Practice, 20(3), from https://doi.org/10.53761/1.20.3.02

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring Issues About Computational Thinking in Higher Education. TechTrends, 59(2), 57–65, from https://doi.org/10.1007/s11528-015-0840-3

Güven, I., & Gulbahar, Y. (2020). Integrating Computational Thinking into Social Studies. The Social Studies, 111(5), 234–248, from https://doi.org/10.1080/00377996.2020.1749017

Lyon, J. A., & J. Magana, A. (2020). Computational thinking in higher education: A review of the literature. Computer Applications in Engineering Education, 28(5), 1174–1189, from https://doi.org/10.1002/cae.22295

Talan, T., & Kalinkara, Y. (2023). Yükseköğrenimde Yapay Zekânın Rolü: Anatomi Dersi için ChatGPT Değerlendirmesi. Uluslararası Yönetim Bilişim Sistemleri ve Bilgisayar Bilimleri Dergisi, 7(1), 33–40, from https://doi.org/10.33461/uybisbbd.1244777

Tekdal, M. (2021). Trends and development in research on computational thinking. Education and Information Technologies, 26(5), 6499–6529, from https://doi.org/10.1007/s10639-021-10617-w

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33–35, from https://doi.org/10.1145/1118178.1118215

Oshima, J., Oshima, R., & Matsuzawa, Y. (2012). Knowledge Building Discourse Explorer: a social network analysis application for knowledge building discourse. Educational Technology Research and Development, 60(5), 903–921, from https://doi.org/10.1007/s11423-012-9265-2

Zhu, G., Kow, J. F., Fan, X., Chit, L. S., & Ong, Y. S. (2023). Pedagogical Design Matters: Integrating Engineering Design and Knowledge Building for Undergraduates to Practice and Improve Computational Thinking Skills, from https://doi.org/10.35542/osf.io/2ugbf

Students' Representational Activities in a Programming-Enhanced Environment: The Case of Linear Function

Yuting HUANG[1*], Biyao LIANG[2]
[1,2] The University of Hong Kong, Hong Kong SAR, China
yutingh96@gmail.com, biyao@hku.hk

ABSTRACT

With the development of information technology, integrating computational thinking (CT) in mathematics education has been widely highlighted by educators and researchers. Scratch is one of the child-friendly blocks-based programming tools that enables access to both mathematical and computational ideas. In order to effectively facilitate student-centered instruction with Scratch, understanding students' thinking processes in this learning environment is essential. The Lesh Translation Model (LTM) provides an analytical tool to interpret and characterize students' thinking processes across multiple representations. In this study, we conducted task-based interviews with three Chinese junior secondary students as they explored the idea of linear function in a programming-enhanced environment. We report on two themes characterizing the students' mathematical representational activities, including their representations of amounts of change and representations of slope as steepness. Accordingly, we describe the students' thinking processes by characterizing representational fluency through the LTM model and discuss the implications of integrating Scratch programming in mathematics education.

KEYWORDS

Computational thinking, Mathematics education, Scratch, Student thinking, Representation.

1. INTRODUCTION

Computational thinking (CT), which refers to thinking "like a computer scientist", is regarded as an essential skill in the 21st century (Wing, 2014). Integrating CT in mathematics education has been highlighted by educators and researchers (Fang, Ng, Tam, & Yuen, 2023). However, educators have experienced challenges integrating CT and mathematics into practice (Ng & Cui, 2021). This difficulty drives researchers to explore affordable learning tools, such as user-friendly programming tools, to support CT-based mathematics learning. Scratch, a block-based programming tool that helps students understand and apply mathematical and computational concepts, is one of the most prevalent CT tools in mathematics education (Chan, Looi, Ho, & Kim, 2023). Research is needed to investigate how it can be used effectively and adequately to enhance mathematics teaching and learning.

Moreover, educators call for a student-centered teaching and learning environment where people switch attention to understanding students' thinking processes. Students' understanding of a mathematical concept requires flexibility in moving across multiple representations that describe different aspects of a concept. The Lesh Translation Model (LTM) (Lesh, 1979) (*Figure 1*) provides an analytical tool to interpret and characterize students' thinking processes, which specifies five types of representations of a mathematical concept and explains their relationships. We adopt the LTM model as the theoretical framework to unpack students' thinking processes.

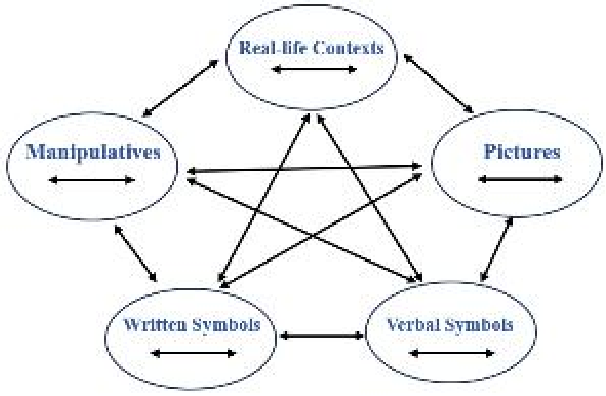


Figure 1. The Lesh Translation Model (Lesh, 1979).

Function is one of the key mathematical concepts, the understanding of which entails a variety of representations (Makonye, 2014). In this study, we situated a group of junior secondary students learning linear functions in Scratch to unpack students' thinking processes and identify the affordances of Scratch. The study is driven by the following research question: *How do students think and reason with linear functions in a programming-enhanced mathematics environment?*

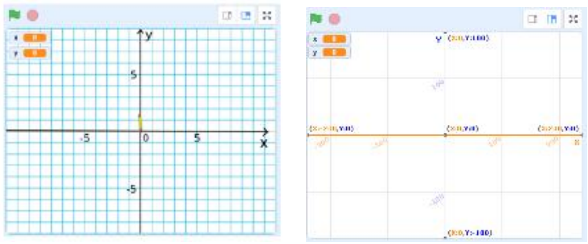
2. METHOD

To answer the research questions, we conducted three one-on-one semi-structured task-based interviews. Each interview lasted for about one hour and a half. Three junior secondary students in China, Adam (pseudonym; Grade 8), Brandon (pseudonym; Grade 8), and Calvin (pseudonym; Grade 7), participated in this study. All students had some prior experiences with Scratch, and none were officially introduced to the concepts of linear function before.

2.1. Task Design

Focusing on the concept of linear function, we design a task involving two subtasks: Part I and Part II. For Part I, Adam and Brandon were presented with an animation (programmed by Scratch) showing how a row of tiles appears one after another. For Calvin, considering his strong background in Scratch programming, we instead asked him to create a program to generate the titling. All students were asked to describe what they saw in the animation and discuss the relationships between the areas and amounts of tiles, which could be modeled by a linear function, $y=kx$, in which y refers to the Total Area Tiled, k refers to the Area of a Single Tile, and x refers to the Number of Tiles.

For Part II, students were asked to produce coordinate graphs of linear functions with different methods and talk about the properties of the graphs of linear functions, such as 1) rate of change ($k=\Delta y/\Delta x$) and 2) slope as steepness (degree of the "tilt" of a line). Adam and Brandon were asked to draw the graphs on paper first and then on the given coordinate system in Scratch (*Figure 2a*). Calvin was asked to draw the graphs directly on Scratch and chose a default stage set up in Scratch (*Figure 2b*).



(a) (b)

Figure 2. Different Stages Used by Different Participants.

2.2. Data Collection and Analysis

The data sources included the video recordings of all the interview sessions from two camera angles, one capturing students' actions and the other focusing on the papers. The Scratch works are also screen-recorded.

We report on the students' activities and thinking based on the data of Part II, the graphing tasks in Scratch. We adapted the LTM model and removed the *Real-life Contexts* representations, as it is addressed in Part I. We focused on four types of representations (*Figure 3*), which are relevant to characterizing students' thinking of mathematical and computational objects in an integrated environment.

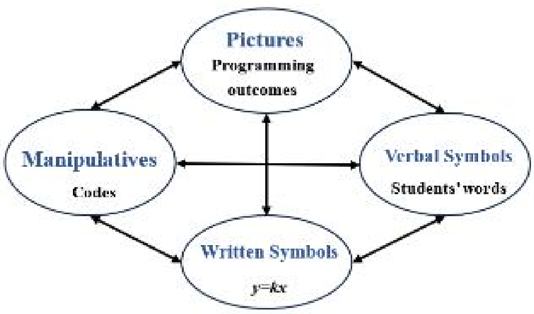
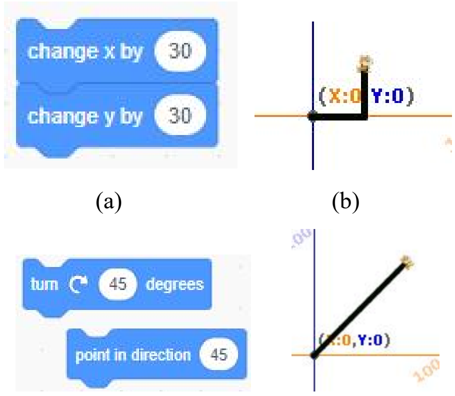


Figure 3. The Adapted LTM Model this Study Used.

We defined the Scratch coding blocks as *Manipulatives* representation and the programming outcomes as *Pictures* representation, which represent different aspects of linear functions. For example, the codes "change x by ( )" and "change y by ( )" (*Figure 4a*) and the outcome (*Figure 4b*) represent how much the two quantities, x and y, change by. The codes "turn ( ) degrees" and "point in direction ( )" (*Figure 4c*) and the outcome (*Figure 4d*) represent the angle between the drawing direction of the Pen and the reference 0-degree facing at the 12 o'clock position (*Figure 5*).



(a) (b)

(c) (d)

Figure 4. The Coding Blocks and the Corresponding Programming Outcome as Representations.

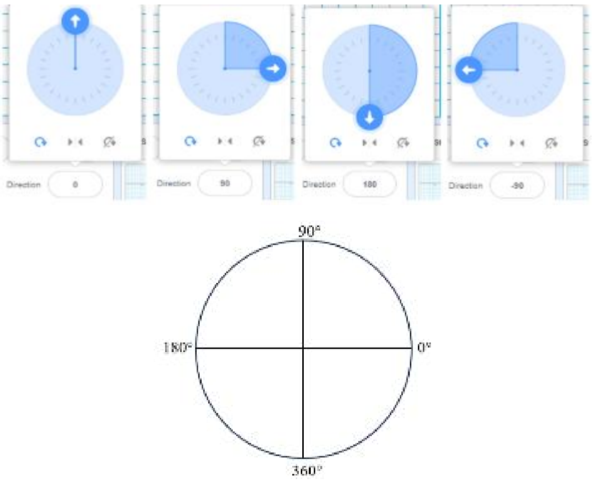


Figure 5. The Setting of Direction in Scratch and Angle in Mathematics.

We conducted interactive coding analysis based on the adapted LTM model to capture the mathematical representations students used and the translation pattern among different representations.

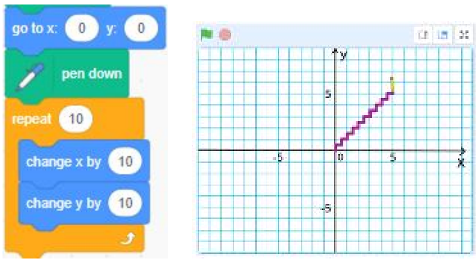
3. RESULTS

In this session, we report on students' activities and thinking in Part II. We identify two themes that occurred during the Scratch tasks to characterize students' representational activities.

3.1. Representations of Amounts of Change

The rate of change (k=Δy/Δx) is an essential idea in the mathematics curriculum in secondary schools. When being asked to draw a graph of a linear function $y=x$ in Scratch, participants used the codes "change x by ( )" and "change y by ( )" to draw the graph.

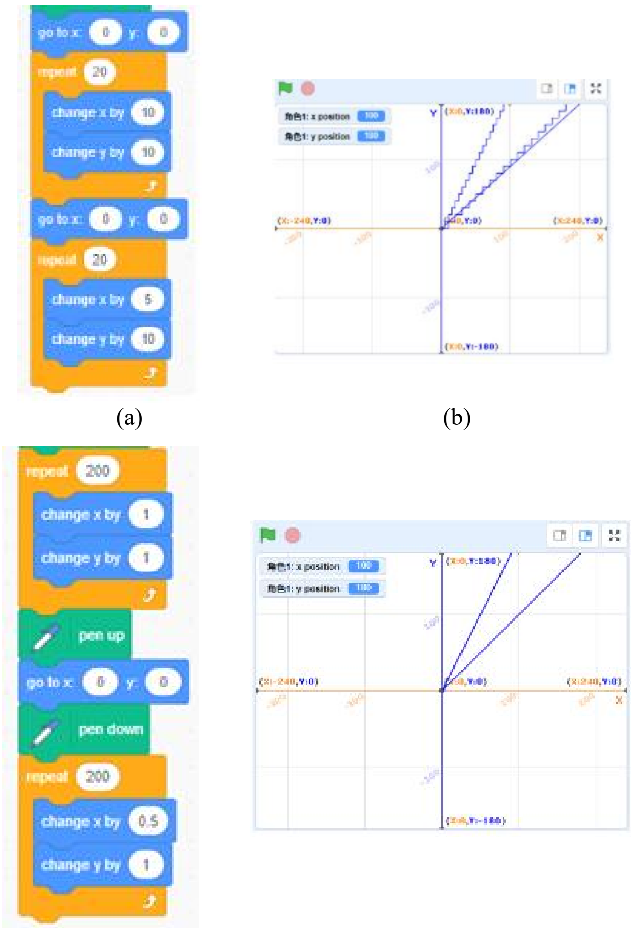
Adam used the loop to draw the graph. He manipulated the codes to change y by 10 and x by 10 by sequence in the loop and repeat it 10 times (*Figure 6a*). However, he refused to accept the outcome (*Figure 6b*), saying, *"Why is it curly, not straight?"*



(a) (b)

Figure 6. Adam's Codes Represent the Amount of Change.

Calvin was asked to draw both $y=x$ and $y=2x$. He used similar codes as Adam to draw a graph of $y=x$, changing x and y by 10 and repeating it 20 times; he then duplicated the codes and changed x by 5 and y by 10 to produce a graph of $y=2x$ (*Figure 7a*). However, he was also unsatisfied with the programming outcome (*Figure 7b*) because the produced graph was not straight enough. He then revised the codes, as shown in *Figure 7c*, and conveyed satisfaction with the graph produced (*Figure 7d*) since he believed that if the intervals were small enough, it could be a straight line. We note that in modifying the codes (*Figure 7a and Figure 7c*), Calvin was able to minimize the values of change in y and change in x by maintaining their ratio, which is an essential property of linear functions.

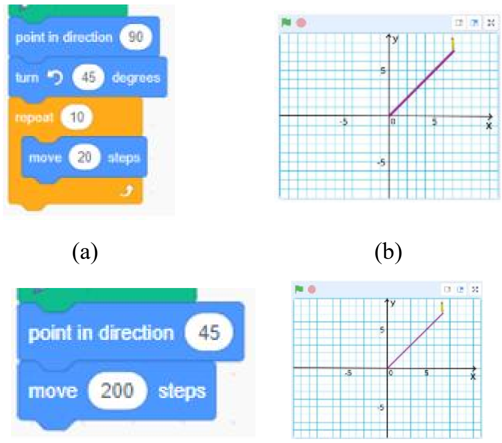


(c) (d)

Figure 7. Calvin's Codes Represent the Amount of Change.

3.2. Representations of Slope as Steepness

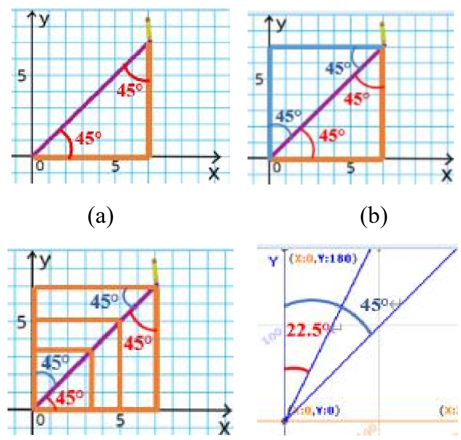
Slope is a core concept in mathematics that characterizes the "steepness" of a linear function graph. We observed that students used the codes "turn ( ) degrees" and "point in direction ( )" to draw graphs. For example, Adam used the code "turn 45 degrees" (*Figure 8a*), and Brandon used the code "point in direction 45" (Figure 8c) to control the Pen's direction, as they had discussed the steepness of the graph of $y=x$.



(c) (d)

Figure 8. Adam's and Brandon's Code Represent the Slope.

When explaining the reason for 45 degrees, Adam and Brandon had different mathematical reasoning. Adam explained, *"The degree is 45 because there hides an isosceles right triangle with two interior angles of 45 °"* (*Figure 9a*). However, Brandon explained that *"there are two triangles here, and they are congruent. So these four angles are all 45 °"* (*Figure 9b*). He also used another explanation for the 45 degrees, *"x and y are all changing by the same number. Every point on the graph of y=x can be the vertex of a square. So they are 45 °"* (*Figure 9c*). Recall that Calvin drew both the graphs of $y=x$ and $y=2x$ together (*Figure 7d*). When asked about the difference between them, he answered, *"I think the angle of y=x is 45°, and the angle of y=2x is 22.5°. Because when y=100, in y=x, x is 100. But in y=2x, x is 50. 50 is half of 100. So, the angle of y=2x is also a half of 45°"* (*Figure 9d*).



(c) (d)

Figure 9. Students' Explanation about the angles.

We draw the reader's attention to the different angles these students were referring to. Calvin referred to the angle formed by the *y*-axis and the linear graph moving clockwise, which aligns with the direction setting of Scratch (*Figure 5*). In contrast, Adam referred to the angle moving counterclockwise, while Brandon's case was special in that he seemed to conceive all angles within a square.

4. DISCUSSION AND CONCLUSION

In this study, we describe students' representational activities in a programming-enhanced mathematical environment through the adapted LTM model and bring implications about integrating Scratch in mathematics education.

4.1. Students' Thinking Processes

We identified empirical evidence of students translating different representations of mathematical concepts when working with Scratch. When asked to draw a graph of $y=x$, the *Written Symbolic* representation of a linear function, the participants first constructed a mental image of the graph (the *Pictures* representation) and manipulated the coding blocks (the *Manipulatives* representation) to produce an intended graph. For example, Calvin used the codes and the corresponding programming outcomes to express his understanding of the rate of change of two variables in linear functions. He could fluently translate from the *Manipulatives* representation to the *Pictures* representation, as he quickly changed the codes to meet his needs (*Figure 7*). Meanwhile, Calvin explained why he used those coding blocks and why he was satisfied or unsatisfied with the outcomes in his own words, switching across *Manipulatives* and *Pictures* representations to *Verbal Symbols* representation.

However, evidence also shows students struggled to translate from Manipulatives representation to Pictures representation. For example, Adam had difficulty accepting the programming outcome produced by his codes (*Figure 6b*). Researchers and educators can help students improve their representational fluency through reasonable task design and scaffolding design.

4.2. Difficulties Arising from the Scratch Environment

In this study, we support the idea that Scratch offers an opportunity to explore the properties of mathematical objects. For participants in this study, the notion of amounts of change became accessible in the context of using Scratch to draw graphs of linear functions.

However, we also raise awareness of the proper use of Scratch in mathematics education. For example, the coordinate system, along with its scale in Scratch, is fixed (*Figure 2b*). When we created a new coordinate system with a different unit length (*Figure 2a*), students may have trouble handling the mismatch between the unit length of a given stage and what they should fill in the coding block to control movement in Scratch. This situation poses challenges to students translating from *Manipulatives* representation to *Pictures* representation.

Moreover, the conventional reference systems in Scratch and mathematics differ. In Scratch, the default direction is set to 90 degrees facing toward the 3 o'clock position (*Figure 5*), and 0-degrees is defined as facing toward the 12 o'clock position. However, in the conventional Cartesian coordinate system, 0-degree is defined as facing towards the 3 o'clock position (*Figure 5*). Therefore, Calvin would need to do a conversion if he were asked to discuss the ideas of slope in a non-Scratch context. We thus called for educators and researchers to consider this mismatch when designing graphing tasks with Scratch.

5. REFERENCES

Chan, S.-W., Looi, C.-K., Ho, W. K., & Kim, M. S. (2023). Tools and Approaches for Integrating Computational Thinking and Mathematics: A Scoping Review of Current Empirical Studies. *Journal of Educational Computing Research*, *60*(8), 2036–2080.

Fang, X., Ng, D. T. K., Tam, W. T., & Yuen, M. (2023). Integrating computational thinking into primary mathematics: A case study of fraction lessons with Scratch programming activities. *Asian Journal for Mathematics Education*, *2*(2), 220–239. SAGE Publications.

Lesh, R. (1979). Mathematical learning disabilities: Considerations for identification, diagnosis, and remediation. *Applied mathematical problem solving* (pp. 111–180).

Makonye, J. P. (2014). Teaching functions using a realistic mathematics education approach: A theoretical perspective. *International Journal of Educational Sciences*, *7*(3), 653–662. Taylor & Francis.

Ng, O.-L., & Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Towards computationally enhanced mathematics education. *ZDM – Mathematics Education*, *53*(4), 847–860.

Wing, J. M. (2014). Computational thinking benefits society. *40th anniversary blog of social issues in computing*, *2014*, 26. Academic Press New York, NY, USA.

One Step Forward towards the use of Human Language to Instruct Computers to Work: A Reflection on an Example of Applying Prompts in Text-based Generative AI for Programming

Siu-Cheung KONG[1*],Enoch Chun Yan SIT[2] ,Nicole Yin YANG[3] ,Wing Kei YEUNG[4]
[1,2,3,4]The Education University of Hong Kong,China
[*]sckong@eduhk.hk,ecysit@eduhk.hk, yyin@eduhk.hk,yeungwk@eduhk.hk

ABSTRACT

In this paper, we suggested that an end-to-end machine code generation framework using a natural language model should involve an intermediate code validation step to ensure the code generated is at least valid and less prone to error. The advancement of large language models such as ChatGPT has shown a promising result in high-level programming generation. However, high-level programming languages, like JavaScript, which can be created from extensive language models, are not necessarily valid and prone to errors during execution. To highlight this issue, we evaluated the current JavaScript quality from ChatGPT with different prompts. We then execute JavaScript using the micro:bit platform during the evaluation process. Although code quality can be improved using carefully crafted prompts, the code generated is not necessarily error-free. As such, we suggested that the status of high-level programming generation using ChatGPT still has much room for improvement. One possible improvement towards the end-to-end code generation is through producing an intermediate abstract syntax tree for code validation using graph and tree-related neural networks.

KEYWORDS

generative AI, humanizing AI, micro:bit, programming, prompts

1. INTRODUCTION

Recent advancement of large language model has shown promising results in the realm of coding generation (Xu et al., 2022). Programmers can generate code using natural languages. However, very often, high-level programming languages such as JavaScript, which can be generated from large language models, are not valid and prone to errors during execution. In this paper, we used an example to illustrate some major challenges we have observed when using prompt to generate JavaScript.

We have designed a program called “Sports Expert” system using the MakeCode platform. The MakeCode platform enables programmers to use JavaScript (or Python), as well as coding blocks for controlling micro:bit. The system can be decomposed into two. The first part refers to the control of servo’s status (movable or non-movable) with button A and B. The second part refers to the control of servo’s swinging angle according to the sound level after button B is pressed. To evaluate the code generation capability, we start off with a simple prompt and then gradually improve the prompt until there are no errors during the testing phases.

We have discovered that the quality of the code generated from ChatGPT can be improved. However, it is worth mentioning that although detailed prompting can make the output pass our test case, ChatGPT’s answer is still not necessarily optimal.

In the deep learning era, end-to-end (Chib & Singh, 2023) deep learning techniques for solving engineering problems have emerged. It is a technique in which we use a deep neural network for solving complex tasks. The input data is raw without any manual feature extraction and the output is the end result that we desire. For instance, an end-to-end solution for meeting transcription would be using a single neural network that takes in audio wave file and outputs meeting transcription. Thus, we propose an end-to-end machine code generation framework by generating intermediate abstract syntax tree would be a way to improve the accuracy and validity of the code generated by large language model.

2. LITERATURE REVIEW

Humanizing AI: The Convergence of Language, Programming, and Learning

One of the applications of machine learning (ML) focuses on how machines interpret and generate code (Dehaerne et al., 2022). For example, Dehaerne et al. (2022) reviewed how algorithms can learn patterns analogous to human language acquisition. Programming language and language learning both rely on understanding syntax, semantics, and context. Machine learning serves as a conduit for insights to flow between these fields. As machines become more adept at understanding and generating code, they simultaneously become more sophisticated in processing human language. Thus, the integration of linguistic methods, coding principles, and educational strategies through LLMs is fostering a novel ecosystem where the lines between human intellect and machine functionality are increasingly interwoven. Human language is the essence of human cognition. By incorporating linguistic features into language models (LLMs), more people can now participate in programming tasks that were previously accessible only to those with specialized training (Xu et al., 2022).

3. AN EXAMPLE OF APPLYING PROMPTS IN TEXT-BASED GENERATIVE AI FOR PROGRAMMING

The following shows an example of using human language to prompt with gpt-3.5-turbo in generating JavaScript for MakeCode on the system of “Sports Expert”. The system includes micro:bit, 180 degree servo motor, three crocodile clip wires, a power source, the “Sports Expert” model (made with cardboard, straw and paperclips) and a paper roll as stand. The system is illustrated at Figure 1 below.

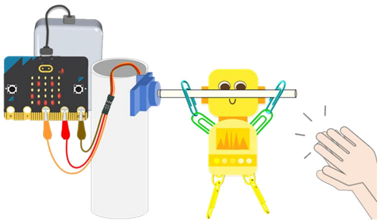
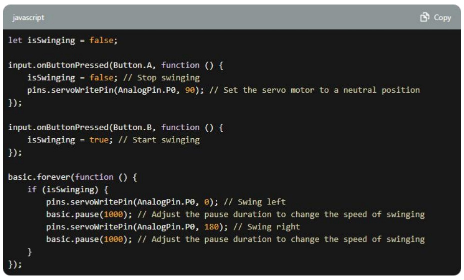


Figure 1. “Sports Expert” system.

The programme of “Sports Expert” system is designed and developed on MakeCode, which enables an interchange of JavaScript (or Python) and coding blocks for controlling micro:bit. When button B on micro:bit is pressed, its LED screen counts down from 3 to 1 and shows a smiley face, before micro:bit starts measuring the sound level of surrounding environment. If the sound level is larger than 30, the servo will swing continuously in a larger angle; if the sound level is less than or equal to 30, the servo will swing continuously in a smaller angle. When button A is pressed, the servo stops moving while micro:bit’s LED screen turns off.

The system was decomposed into two parts when it was built. The first part refers to the codes for changing the status of servo (movable or non-movable) with button A and B. The second part refers to the change of servo’s swinging angle according to the sound level after button B is pressed. On the first step, we requested ChatGPT to generate the code with a simple prompt, “Write the JavaScript of MakeCode for micro:bit. When button B is pressed, the servo motor swings continuously. When button A is pressed, servo motor stops swinging.”. Figure 2 and 3 show a successful example of JavaScript generated by ChatGPT, and its corresponding coding blocks on MakeCode.



javascript Copy

```javascript
let isSwinging = false;

input.onButtonPressed(Button.A, function () {
    isSwinging = false; // Stop swinging
    pins.servoWritePin(AnalogPin.P0, 90); // Set the servo motor to a neutral position
});

input.onButtonPressed(Button.B, function () {
    isSwinging = true; // Start swinging
});

basic.forever(function () {
    if (isSwinging) {
        pins.servoWritePin(AnalogPin.P0, 0); // Swing left
        basic.pause(1000); // Adjust the pause duration to change the speed of swinging
        pins.servoWritePin(AnalogPin.P0, 180); // Swing right
        basic.pause(1000); // Adjust the pause duration to change the speed of swinging
    }
});
```

Figure 2. Example of JavaScript Generated by ChatGPT.

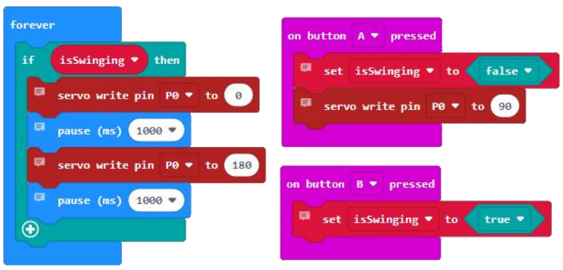


Figure 3. Coding blocks corresponding to the generated codes.

ChatGPT does not always extract appropriate JavaScript according to the prompts given. One of the cases is the use of the event “OnButtonReleased”, which is common in other JavaScript projects. However, it cannot be used by micro:bit with MakeCode, but instead triggers error message. There are various similar cases where codes generated doesn’t exist in MakeCode’s coding block list. Figure 4 shows an example of blocks that include a part of JavaScript blocks (in grey), which can be run by MakeCode, but doesn’t exist in MakeCode’s coding block list. In another example, the servo in the system only moves when button B is pressed down continuously, which does not fit to our expectation that servo swing continuously after button B is pressed once.

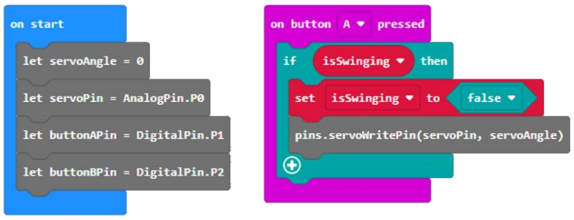
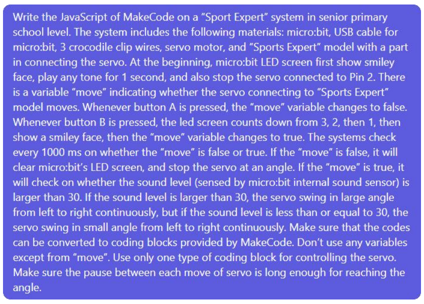


Figure 4. Coding blocks in grey cannot be found on coding block list.

Throughout the process of prompting, we attempted to add extra instructions and information to the prompt for improving ChatGPT’s output. For example, after adding the instruction “Make sure that the codes can be converted to coding blocks provided by MakeCode.” and “Do not use JavaScript blocks”, ChatGPT has reduced the use of non-existing coding blocks. A similar approach is used when prompting the full system in a single prompt. As an example, in the progress of repeated testing on requesting the servo to swing in smaller angle rather than swinging between 0 and 180 degrees, it was also found that ChatGPT’s performance varies greatly on the sequence of words in a sentence. The prompt “servo motor swings in a small angle, left and right continuously” gives better result than the prompt of “servo motor swings left and right continuously, in a small angle”. ChatGPT also failed to use newly-added blocks which don’t exist in its database, such as the music playing coding blocks using speaker on micro:bit version 2. The refined prompt for generating a complete system with one single prompt is shown in Figure 5 below.



Write the JavaScript of MakeCode on a “Sport Expert” system in senior primary school level. The system includes the following materials: micro:bit, USB cable for micro:bit, 3 crocodile clip wires, servo motor, and “Sports Expert” model with a part in connecting the servo. At the beginning, micro:bit LED screen first show smiley face, play any tone for 1 second, and also stop the servo connected to Pin 2. There is a variable “move” indicating whether the servo connecting to “Sports Expert” model moves. Whenever button A is pressed, the “move” variable changes to false. Whenever button B is pressed, the led screen counts down from 3, 2, then 1, then show a smiley face, then the “move” variable changes to true. The systems check every 1000 ms on whether the “move” is false or true. If the “move” is false, it will clear micro:bit’s LED screen, and stop the servo at an angle. If the “move” is true, it will check on whether the sound level (sensed by micro:bit internal sound sensor) is larger than 30. If the sound level is larger than 30, the servo swing in large angle from left to right continuously, but if the sound level is less than or equal to 30, the servo swing in small angle from left to right continuously. Make sure that the codes can be converted to coding blocks provided by MakeCode. Don’t use any variables except from “move”. Use only one type of coding block for controlling the servo. Make sure the pause between each move of servo is long enough for reaching the angle.

Figure 5. Refined prompt used for generating a complete “Sports Expert” system.

Although the above prompt is written in detail, which makes the output closer to our expectation, ChatGPT’s answer is still not necessarily correct. In the following example of system generated by ChatGPT on Figure 6, it has oversighted the request of showing a smiley face at the end of counting down after pressing button B. ChatGPT has also used "forever" loop instead of the requested "every 1000 ms" loop. Besides, an extra pause with 1000 ms is added to the end of the forever loop even though it was not requested.

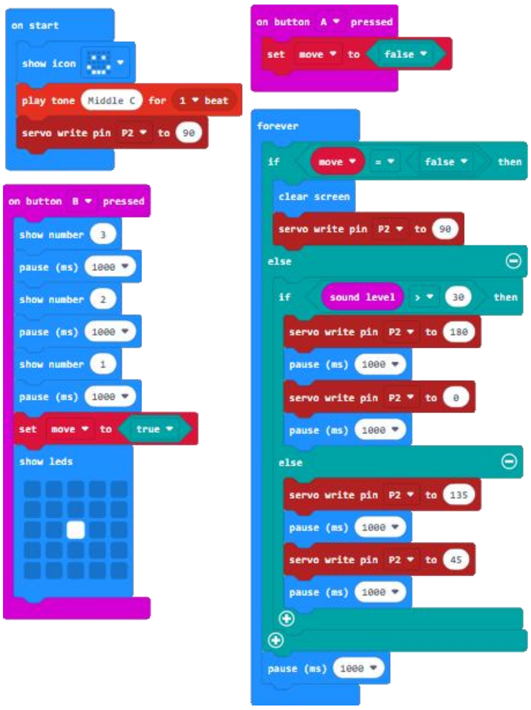


Figure 6. Coding blocks of the complete system corresponding to the codes generated by ChatGPT.

4. DISCUSSION

From the example illustrated above, we observed that ChatGPT can only generate basic JavaScript code for micro:bit MakeCode when provided with a simple prompt describing the desired functionality. However, the code generated from ChatGPT is prone to incompatible syntax errors. Furthermore, detailed context about the specific hardware, coding environment and coding platform are essential to help ChatGPT to generate correct code. However, the phrasing of the prompt matters and small tweaks of prompt are necessary. The variation of codes is also restricted due to the limited training data. All in all, ChatGPT demonstrates great potential for instructed code generation yet requires human validation for code validity and quality.

Deep learning models have been well known for their ability to perform end-to-end tasks. For example, Open AI whisper architectures (Radford et al., 2022) is a simple end-to-end approach that directly translates audio spectrogram to text. Similarly, future deep learning models should be able to generate valid executables or machine code directly from natural language. One possible steppingstone towards the natural-language-to-valid-executables is generating intermediate abstract syntax tree for code validation. Graph Neural Networks (Zhou et al., 2020), Graph-to-Tree (Li et al., 2020), Tree-to-tree (Chen, 2018) promise deep-learning architectures to produce valid executables from natural language.

5. CONCLUSION

In this paper, we used ChatGPT to generate JavaScript code for micro:bit platform MakeCode on the system of "Sports Expert". We have discovered even with a well written prompt, ChatGPT's answers are still not necessarily correct. We proposed that one possible improvement towards the natural-language-to-valid-executables is through producing intermediate abstract syntax tree for code validation using graph and trees related neural network. This study is tentative, and more empirical studies are needed to validate the findings and to ensure that the generated code is not only syntactically correct but also semantically functional across various scenarios. Further research is required to explore the generalizability of the approach to different programming languages as well as to compare the performance of ChatGPT with other large language models in code generation tasks.

6. Acknowledgements

The authors would like to acknowledge this artificial intelligence literacy project with the funding support from the Li Ka Shing Foundation and the Research Grants Council, University Grants Committee of the Hong Kong Special Administrative Region, China (Project No. EdUHK CB326, CB357).

7. REFERENCES

Chen, X., Liu, C., & Song, D. (2018). Tree-to-tree Neural Networks for Program Translation. *ArXiv.Org*. https://doi.org/10.48550/arxiv.1802.03691

Chib, P. S., & Singh, P. (2023). Recent Advancements in End-to-End Autonomous Driving using Deep Learning: A Survey. *IEEE Transactions on Intelligent Vehicles*, 1–18. https://doi.org/10.1109/tiv.2023.3318070

Dehaerne, E., Dey, B., Halder, S., De Gendt, S., & Meert, W. (2022). Code generation using machine learning: A systematic review. *IEEE Access*, 10, 82434–82455. https://doi.org/10.1109/ACCESS.2022.3196347

Li, S., Wu, L., Feng, S., Xu, F., Xu, F., & Zhong, S. (2020). Graph-to-Tree Neural Networks for Learning Structured Input-Output Translation with Applications to Semantic Parsing and Math Word Problem. *arXiv* (Cornell University). https://doi.org/10.48550/arxiv.2004.13781

Radford, A., Kim, J. W., Xu, T., Brockman, G., McLeavey, C., & Sutskever, I. (2022). Robust speech recognition via Large-Scale Weak Supervision. *arXiv* (Cornell University). https://doi.org/10.48550/arxiv.2212.04356

Xu, F. F., Alon, U., Neubig, G., & Hellendoorn, V. J. (2022). A systematic evaluation of large language models of code. *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming, MAPS 2022*. https://doi.org/10.1145/3520312.3534862

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open, 1*, 57–81. https://doi.org/10.1016/j.aiopen.2021.01.001

The Application and Practice of Computational Thinking in Primary and Secondary Education

Liang-Hui Wu[1], Xiao-Qian Wu[2]
[1,2]Liang-Hui Wu,Shenzhen Institute of Education Sciences,China/Shenzhen；
Xiao-Qian Wu，Shenzhen Shekou Yucai Education Group Yucai No.2 Middle School，China/Shenzhen；
93519051@qq.com, 214287860@qq.com

Abstract

This article outlines the theoretical framework, characteristics, and importance of computational thinking in K-12 education. It defines computational thinking and its unique aspects, emphasizing the necessity of cultivating students' problem-solving skills. The research examines the four main components of computational thinking: abstraction, problem decomposition, system modeling, and algorithmic strategies, along with their interrelationships. Through a systematic analysis of these key elements, the paper aims to provide a scientific basis and operational guidelines for the theoretical construction and practical application of computational thinking education in K-12 schools. The goal is to promote students' abilities in problem-solving and innovative thinking.

Keywords

Computational Thinking, Basic Education, Abstraction, Problem Decomposition, System Modeling, Algorithmic Strategizing

1. Introduction

Computational thinking significantly contributes to the progression of contemporary society (Zhu, 2009). The Compulsory Education Information Technology Curriculum Standards (2022 edition) emphasize the importance of cultivating computational thinking. Despite the growing interest in computational thinking education from 2015 to 2023, as evidenced by publications on the China National Knowledge Infrastructure (CNKI), primary and secondary education in China is still in the early stages of exploring this field, lacking mature strategies and applications. This study is dedicated to exploring the application of computational thinking in primary and secondary education, aiming to enhance students' problem-solving and digital literacy skills.

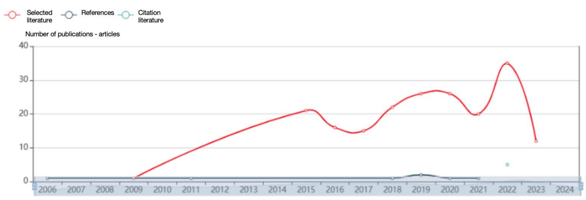


Figure 1. Literature distribution of Computational thinking from 2015 to 2023.

2. An Overview of Computational Thinking

Computational thinking, essential for solving problems via computer science and information technology, is deemed a necessary literacy in the information society (Chen, Zhang, & Yang, 2023). Diverse definitions of computational thinking have been offered by entities such as the International Association for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) in the United States, the Royal Society in the United Kingdom, and the Chinese Ministry of Education between 2011 and 2016, showcasing notable discrepancies (Yu, Xiao, & Wang, 2018). This paper, drawing from the definition in the Compulsory Education Information Technology Curriculum Standards (2022 edition), examines strategies for integrating and enhancing computational thinking within educational frameworks.

3. Core Elements of Computational Thinking

To enhance teaching practices, various institutions and scholars have highlighted the essential components of computational thinking. In 2011, CSTA and ISTE released the Computational Thinking Leadership Toolbox, outlining nine key elements: data collection, analysis, representation, problem decomposition, abstract thinking, algorithm design, automation, simulation, and parallelism. The Computing At School of the United Kingdom (CAS) also emphasized abstract thinking, problem decomposition, algorithmic thinking, pattern recognition, and evaluation skills as central to computational thinking.

Table 1. Elements of Computational Thinking published by authoritative organizations.

	CSTA ISTE	CAS UK	MT CS	China's Standards 2017	China's Standards 2022
Abstraction	√	√	√	√	√
Breakdown	√	√			√
Modeling		√		√	√
Algorithm design	√	√	√		√
Automation	√				
Data Collection	√		√		
Data analysis	√				
Data representation	√		√		
Simulation	√				
Parallelism/ concurrency	√				
Review		√			

Similarly, Australia's Mathematical Technology Curriculum Standards (MTCS) focus on abstraction, data handling, and algorithm implementation as crucial concepts (Xiao & Gao, 2015). In China, the Information Technology

Curriculum Standards for General High Schools (2017 edition) and the Compulsory Education Information Technology Curriculum Standards (2022 edition) stress the importance of problem definition, feature refinement, structural modeling, and data organization, alongside abstraction, decomposition, modeling, and algorithm design as key processes in computational thinking.

Based on the above research results on the core elements of computational thinking, we can integrate them into a detailed table, as shown in Table 1.

We will discuss the four core elements of computational thinking: abstraction, decomposition, modeling, and algorithm design, as proposed in the domestic information technology curriculum standards.

3.1. Abstraction

The concept of abstraction, corresponding to the concrete, refers to a thinking process that refines, extracts, or generalizes the common and essential attributes, relations, and other elements from concrete things and abandons individual non-essential aspects. This way of thinking makes problems more concise and general, which helps us understand and solve problems more efficiently. For example, in programming, abstractions can help us define common functions and classes, thus avoiding code duplication and improving the maintainability and extensibility of programs.

3.2. Decomposition

The concept of decomposition in computational thinking refers to breaking down a complex problem into multiple relatively simple, tractable problems that can be better understood and solved. By breaking it down, we can work through each subproblem step by step, and eventually combine the solutions of the subproblems to solve the whole problem. This approach not only makes the problem easier to understand and handle but also promotes teamwork because different people or teams can work on different subproblems in parallel.

3.3. Modeling

Modeling is the process of creating an abstract representation of a problem, and it usually involves building mathematical or logical models that simulate and analyze real-world situations. Through modeling, we can test our understanding under different assumptions and conditions, and predict the outcome of practical actions. This is a powerful tool that can help us evaluate different solutions before implementation, thus avoiding potential errors and risks.

3.4. Algorithm Design

Algorithm design, another core element of computational thinking, involves formulating a clear, ordered sequence of steps to solve a problem or perform a task. Algorithms are the foundation of the computational process, and they must be detailed and precise enough so that they can be executed by a computer. The design of algorithms requires not only logical rigor but also efficiency and optimization to ensure the performance and reliability of the solution.

3.5. The Relationship Between the Elements

The Order of the Four Elements In addressing specific problems, the usual sequence of these four elements is abstraction, decomposition, modeling, and algorithm design. Of course, this sequence may be adjusted according to the nature and requirements of the problem at hand. Below is a typical process for solving a problem:

Abstraction: First, extract key information from the problem and eliminate irrelevant details to form a simplified description of the problem. This helps us focus on the essence of the problem and lays the foundation for subsequent steps.

Decomposition: Next, break down the problem into smaller, more manageable sub-problems. This allows us to delve into the problem step by step, tackle challenges one by one, and also provides a clear path for subsequent modeling and algorithm design.

Modeling: Express the problem or sub-problems using mathematical, logical, or computer science methods. The purpose of this step is to transform the problem into a computable form so that it can be processed using tools such as computers.

Algorithm Design: Finally, design a series of clear, precise, and executable steps to solve the problem or sub-problems. This enables us to transform the results of the previous abstraction, decomposition, and modeling into a practical, executable solution.

In practice, there is often interaction and iterative cycling between these four elements. For example, during the modeling phase, we may need to revert to the abstraction phase to refine core information; similarly, during the algorithm design phase, we might encounter the need for further detailed decomposition of the problem. Therefore, it is crucial to flexibly use and adjust the application order of these four elements according to the actual situation when solving specific problems. This is illustrated in Figure 2.

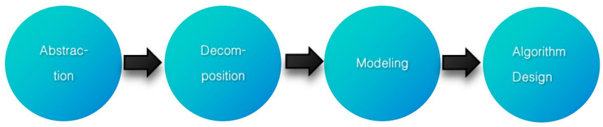


Figure 2. The sequence of the four elements.

4. Teaching Practices of Computational Thinking in Primary and Secondary Schools

Since computational thinking education focuses on the development and training of a mode of thinking, it has different requirements and objectives from traditional teaching. These requirements and objectives pose new challenges to existing educational concepts and methods (Li, 2012). However, implementing computational thinking teaching practices, educators can effectively enhance students' logical thinking capabilities and digital literacy, laying a solid foundation for their lifelong learning and career development. Currently, educational practices around the globe have adopted various methods to ensure that students can effectively engage with and master computational thinking skills during their primary and secondary education stages.

Information Exchange and Sharing: In primary and secondary school information technology courses, the teaching practice of computational thinking emphasizes effective information exchange and sharing by students through the use of information technology tools. For example, the American K-12 Computer Science Framework mentions that third-grade students should be able to use writing tools, digital cameras, and drawing tools, among others, to express ideas, viewpoints, and events in a step-by-step manner (Zhao, Li, Jiang, & Lang, 2017).

Information Privacy and Security: In computational thinking education, students need to understand and master the basic knowledge of information privacy and security. Teachers can guide students to recognize the importance of personal information protection and teach them how to safely manage and share personal data in daily life, as well as how to identify and prevent cybersecurity threats.

Online Learning and Living: Online learning has become an important part of computational thinking education. Teachers should encourage students to use online resources for self-learning, while teaching them how to effectively manage time and evaluate resources in the online environment to meet the needs of the information society.

Data and Coding: At the K-12 level, educators should design learning content that is suitable for students to understand and master, such as teaching sixth-grade students the basic steps of solving problems using algorithms. Through programming activities, students can learn how to collect, analyze, and utilize data, as well as how to automate information processing through coding.

Algorithms in Daily Life: Teachers use real-life examples, like search engines, to demystify algorithms, helping students grasp their application in problem-solving.

Processes and Control: Learning programming languages enables students to grasp program flow control, enhancing their problem-solving skills through structured processes.

Internet Applications and Innovation: The internet offers a vast platform for learning programming and engaging in innovative projects, enriching students' understanding of computational thinking and its practical applications.

IoT Practices and Exploration: IoT technology integration allows hands-on exploration of smart device programming, making computational concepts tangible and enhancing tech mastery.

Artificial Intelligence and Smart Society: AI education introduces students to cutting-edge problem-solving and decision-making, emphasizing computational thinking's role in modern intelligent systems.

Data and Computation: Teaching the importance of data and computation equips students with the skills to analyze and utilize data, applying computational methods to real-world problems.

Information Systems and Society: Understanding information systems' structure and societal impact underlines computational thinking's significance, fostering a sense of responsibility and innovation.

5. Conclusions

Through the widespread recognition of the importance of computational thinking, various countries and scholars have proposed their own standards and frameworks. To better adapt to the specific needs of the Chinese educational environment, this study, based on a comprehensive analysis, re-examines the core elements of computational thinking and their interrelationships. These elements are not only the cornerstone of computer science education but also an indispensable part of the modern educational system, especially in cultivating students' critical thinking abilities to face future challenges.

However, from a practical operational perspective, the implementation of computational thinking education faces multiple challenges, including the uneven distribution of educational resources, the need for professional growth among teachers, and how to effectively integrate computational thinking with students' daily lives and other disciplinary knowledge. In light of this, this study delves into more than a dozen computational thinking teaching practices nationwide, including information exchange and sharing, information privacy, and security. By showcasing these teaching practices, the aim is to provide educators with a broader perspective, enabling them to more deeply integrate computational thinking teaching strategies and methods into daily teaching activities.

6. References

Chen, X., Zhang, H., & Yang, Y. (2023). Comparison of Research Context and Practice of Computational Thinking Education in China and Abroad. Primary & Secondary Schooling Abroad, (01), 148-161.

Li, L. (2012). Computational thinking: Concepts and challenges. China University Teaching, (01), 7-12.

Xiao, G., & Gao, D. (2015). Cultivating Computational Thinking: New Choice of ICT in High School. Modern Educational Technology, (07), 38-43.

Yu, X., Xiao, M., & Wang, M. (2018). Cultivation of Computational Thinking in Progress: Practice and Evaluation in K-12. Journal of Distance Education, (02), 18-28.

Zhao, W., Li, S., Jiang, Q., & Lang, X. (2017). Cultivating Computational Thinking and Developing STEM Education. China Educational Technology, (05), 47-53.

Zhu, Y. (2009). Computational Thinking: its Scientific Position, Basic Principles and Innovation Methods. Computer Science, (04), 53-55+93.

A Teacher's Professional Learning of the Computational Practice of Abstraction Through Coaching: Changes and Challenges

Zhichun LIU[1*], Biyao LIANG[2], Chandra ORILL[3], Shakhnoza KAYUMOVA[4], Ramprasad BALASUBRAMANIAN[5]
[1, 2]The University of Hong Kong, Hong Kon SAR, China
[3]ReThink Learning Labs, MA, USA
[4, 5]University of Massachusetts Dartmouth, MA, USA
*lukasliu@hku.hk, biyao@hku.hk, chandra.orrill@rethinklearning.com, skayumova@umassd.edu, r.bala@umassd.edu

ABSTRACT
Although computational thinking (CT) professional learning opportunities are becoming more prevalent in elementary schools, in-service teachers face persistent challenges in learning and implementing CT due to the complex changes in their beliefs, knowledge, and practices. In this study, we explore the coaching experiences of an elementary science teacher over 10 months of CT professional learning and investigate the changes in her beliefs and practices related to the notion of abstraction in CT. Findings suggest that prolonged coaching is an effective approach to encourage the teacher to initiate CT practices in her classrooms. However, coaching also introduced misconceptions and potential confusion about CT. Implications for the design of CT professional learning programs to support teachers are discussed.

KEYWORDS
Computational Thinking, Professional Learning, Coaching, Abstraction

1. INTRODUCTION
Computational thinking (CT) has gained increasing attention as an important skill for K-12 students to develop (Kafai & Proctor, 2022). As many schools start to incorporate CT into their curricula, supporting teachers' learning and effective integration of CT principles poses a significant challenge (Wang et al., 2022). Existing research has highlighted teachers' lack of familiarity with CT concepts and struggles translating CT-focused professional learning experiences into classroom practices (Ketelhut, et al., 2020).

One promising approach for overcoming these challenges is extending professional learning support over time through coaching (Lefstein et al., 2020). Coaching allows for prolonged, individualized feedback and guidance that can help address gaps as they emerge for teachers navigating complex changes in beliefs and pedagogy. Although past studies have confirmed the positive impact of coaching on teachers' professional learning in CT (Gibbons et al., 2021), researchers and practitioners tend to assume that coaching is the "silver bullet" for bringing CT into classrooms. Little research to date has explored the longitudinal experiences of teachers participating in extended CT coaching. This study seeks to address that gap by investigating the professional learning journey of an elementary science teacher through a 10-month one-on-one coaching program focused on CT integration. An in-depth examination of her experiences may provide insights into both the affordances and limitations of prolonged coaching models for CT professional development.

2. Computational Thinking and Teachers' Professional Learning
There has been a considerable research effort in promoting and examining teachers' professional development through CS/CT-focused professional learning programs. However, past research has found that in-service teachers frequently lack an understanding of CT concepts and struggle to implement lessons (Angeli & Giannakos, 2020). As such, successfully integrating CT into existing curricula often represents a significant pedagogical shift for many teachers. For instance, Ketelhut et al. (2020) investigated how a PL program with workshops, coaching, and an inquiry community changed elementary teachers' views on CT. Among the increasing efforts in teachers' PL for CT, many lack sustained and targeted support for teachers (Liu, 2023). Acknowledging that teacher change in knowledge and instructional practices is a long and complex process and short-term workshops have proven insufficient for facilitating complex changes in teachers' beliefs and practices (Desimone, 2009). Coaching offers long-term, individualized support shown to be effective for science and literacy integration (Lefstein et al., 2020). By addressing gaps as they emerge over time, coaching holds promise for supporting teachers' CT learning. However, few studies have examined coaching's role in developing teachers' CT knowledge and skills.

3. Coaching
From a sociocultural perspective, learning occurs through social interaction and collaborative experiences. Coaching provides opportunities for teachers to learn CT concepts through discussions with experts, as well as hands-on experimentation with and reflection on practices in their own classrooms (Gibbons et al., 2021). However, coaching also involves a process of meaningful change for teachers. Complex challenges may arise when (a) revealing and probing problems of practice, (b) providing evidence or reasoning, (c) making connections to general principles, (d) building on others' ideas, and (e) offering different perspectives (Lefstein et al., 2020). This study therefore also adopts a perspective of teacher learning as a complex process of change that is facilitated through social interactions between the teacher and the coach. The coaching conversations will be examined for how they influence teacher change over time. To this end, we aim to understand how the coach supported an elementary science teacher to develop computational thinking practices. In this study, the research question we seek to answer is: What are the different understandings of

abstraction that were evidenced in the coaching sessions over time?

4. Method

4.1. Context

This study describes the partial effort of a larger research project aimed at exploring a sustainable professional learning model for elementary school teachers to learn and implement CT in their classrooms. The professional development program in this study consists of online modules of CT learning, monthly workshops, monthly coaching sessions, and collaborative design and implementation of CT-integrated lessons in real classrooms. The entire professional development program was led by coaches and facilitators from a professional development company.

4.2. Case Study

In this study, we take a case study approach (Stake, 1995) to focus on one elementary science teacher, Cassie (pseudonym). Cassie has been very active and engaged compared to the other participants. In this report, we focus on the one-on-one coaching sessions in 10 months. In these sessions, Cassie had sustained interactions with her coach, Jackson, who played a major role in supporting Cassie in her professional learning and teaching practices.

4.3. Data Sources and Analyses

To answer the research question, we drew data from Cassie's interviews, coaching sessions, and the field notes that researchers have taken. We coded the video episodes of those research sessions to identify instances where Cassie and the coach discussed the notion of abstraction and generated interpretive memos describing the instances. Then, we adopted open and axial coding methods to identify emergent themes characterizing Cassie's diverse understanding of abstraction.

5. Results

Over the course of the professional learning program, we observed (a) Cassie has transitioned from a gross understanding of abstraction to (b) multiple elaborated understandings and finally transformed these understandings into (c) pedagogical understandings that afforded her implementation of abstraction in her classrooms. The following sections describe these three types of understanding of abstraction.

5.1. Gross Understanding of Abstraction

In the first coaching session, in responding to Cassie's wondering about the meaning of abstraction, Jackson explained that abstraction is about "identifying the important information" or "picking out which information is the key information" from a given situation. They discussed the example of why something melts, and considered what makes different objects able to melt. Cassie was asked to further inquire into the condition under which ice and cheese melt, and she considered temperature as the key condition. Jackson agreed that being able to identify temperature as the key information is an indicator of abstraction. That is, selecting temperature from many other factors and properties involves abstraction. Upon completion of this coaching session, although we observed that Cassie and Jackson seemed to arrive at an agreement about the meaning of abstraction, we also observed uncertainty in Cassie's utterances, especially in terms of what should abstraction activity look like in her classroom and what she should do to support students' learning of abstraction.

5.2. Elaborated Understandings of Abstraction

In the subsequent sessions, we have identified two different, but related elaborated understandings of abstraction that emerged from the discussion between Cassie and Jackson. The first type of elaborated understanding views abstraction as achieving a goal or succeeding in solving a problem. This understanding emerged during the conversations about differentiating abstraction from decomposition. In the fourth coaching session, Cassie and Jackson discussed an unplugged classroom activity Cassie uses called Rossie's Runtime. In this activity, students use cardboard to direct Rosie, the robotic dog, through a maze. Following the idea of abstraction means achieving the goal, Cassie explained that decomposition would mean the students break down what they need to do, such as avoiding the puddles, finding the bone, and getting to the park, while abstraction would mean Rosie successfully arrives at the doghouse and dry with a bone, meaning "it's all over".

In addition to abstraction meaning achieving the goal, we inferred another related understanding of abstraction from the instances. That is abstraction is the process of identifying the qualities of a constituting element in order to achieve a certain goal, while decomposition is the process of identifying the element itself. This understanding implies that abstraction has to come after decomposition when solving a problem. One needs to decompose a situation into subcomponents, and abstraction helps tackle each to achieve the goal. This understanding is not entirely separate from the previous but related to it since it is also a goal-oriented understanding with a hallmark of abstraction accomplishing the goal.

Building upon these two understandings, and in their attempt to come up with a diagram to illustrate abstraction for students (during the seventh coaching session), Jackson asked "what is a thing that only works when you put all the pieces together?" Cassie proposed a puzzle diagram or a pie might be good illustrations. Jackson responded with enthusiasm that these are perfect ideas, saying "if your pie has all these pieces, you've probably met that abstraction goal." In the eighth coaching session, Cassie reflected on how the puzzle diagram was helpful for her students when they used it as a checklist or guidelines for their activities. We consider this to be additional evidence of Cassie primarily interpreting abstraction as associating with outcomes, which is discrepant from a view of abstraction as a process or CT practice in extant literature (e.g., Shute et al., 2017).

5.3. Pedagogical Understanding of Abstraction

When considering how to teach abstraction in classrooms, Jackson recommended that Cassie could directly introduce the vocabulary of abstraction at the very beginning, such as saying, "We are going to use the thinking skill of abstraction to identify what is the most important information we are looking for." Jackson further explained,

"I believe the key for this… is that when we prompt students, and we make it part of their problem solving routine." Jackson's suggestion implies a view of abstraction as a cue guiding students' activities and thinking. We observed that Cassie picked up on this strategy and used the terminology of abstraction explicitly in her classrooms. When asked by Jackson how incorporating the notion of abstraction might influence her task design, Cassie admitted that it did not influence the design itself but merely for communicating with or providing instructions to students. This observation is also supported by the interview with Cassie at the end of the coaching sessions. Although she claimed that she felt much more confident in integrating abstraction, she did not provide any concrete explanations of what abstraction is and admitted that she is still confused with abstraction and decomposition.

6. Discussion

The findings illustrated the changes and challenges Cassie has experienced regarding the notion of abstraction in CT. While Cassie has demonstrated progress in understanding and operationalizing abstraction in CT and implemented lessons with abstraction and concrete examples, some conceptual challenges remain. Cassie's elaborated understandings of abstraction as achieving goals or identifying qualities of elements can be discrepant from literature definitions that position abstraction as a cognitive process (Shute et al., 2017). We posit that the discrepancy is not necessarily a result of miscommunication. In fact, Jackson helped Cassie to understand abstraction through various concrete examples. The discrepancy may reflect the complexity of the notion of abstraction and CT at a cognitive level. CT itself can be interpreted differently depending on the theoretical foundations and the contexts (Kafai & Proctor, 2022; Lodi &Martini, 2021). Although many practitioners may not be able to engage in in-depth discussions about what CT means to them immediately, it is non-trivial for them to understand that CT can be operationalized in their own context (Liu, 2023). In this study, seeing abstraction only through its outcomes may hamper Cassie's ability to design learning experiences that effectively support her students. In future professional learning, coaching should consider prompt critical reflection on how conceptual understandings influence teaching beyond superficial adaptations (e.g., offering different perspectives in Lefstein et al., 2020).

7. Conclusion

This study explored an elementary teacher's CT learning of abstraction through one-on-one coaching over 10 months. Findings reveal both progress and ongoing challenges in Cassie's evolving understandings and implementation of abstraction. Future research should continue examining how to improve professional support structures for enabling meaningful teacher change in learning and teaching computational thinking.

Acknowledgment

This work is supported by the National Science Foundation under Grant 2331742. Any opinions, findings, conclusions, or recommendations expressed in this work do not necessarily reflect the views of the NSF.

8. REFERENCES

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior, 105*, 106185. https://doi.org/10.1016/j.chb.2019.106185

Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. *Educational Researcher, 38*(3), 181-199. https://doi.org/10.1016/S0742-051X(02)00053-7

Gibbons, L. K., Lewis, R. M., Nieman, H., & Resnick, A. F. (2021). Conceptualizing the work of facilitating practice-embedded teacher learning. *Teaching and Teacher Education*, *101*, 103304. https://doi.org/10.1016/j.tate.2021.103304

Kafai, Y. B., & Proctor, C. (2022). A revaluation of computational thinking in K–12 education: Moving toward computational literacies. *Educational Researcher, 51*(2), 146-151. https://doi.org/10.3102/0013189X211057904

Ketelhut, D. J., Mills, K., Hestness, E., Cabrera, L., Plane, J., & McGinnis, J. R. (2020). Teacher change following a professional development experience in integrating computational thinking into elementary science. *Journal of science education and technology*, *29*, 174-188. https://doi.org/10.1007/s10956-019-09798-4

Lefstein, A., Louie, N., Segal, A., & Becher, A. (2020). Taking stock of research on teacher collaborative discourse: Theory and method in a nascent field. *Teaching and Teacher Education, 88*, 102954. https://doi.org/10.1016/j.tate.2019.102954

Liu, Z. (2023). Integrating computational thinking into K-12 education: Bridging the gap between theories and practices. *STEM Education Review, 1*. https://doi.org/10.54844/stemer.2023.0467

Lodi, M., & Martini, S. (2021). Computational thinking, between Papert and Wing. *Science & Education*, *30*(4), 883-908. https://doi.org/10.1007/s11191-021-00268-1

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

Stake, R. E. (1995). *The art of case study research*. Sage.

Wang, C., Shen, J., & Chao, J. (2022). Integrating computational thinking in STEM education: A literature review. *International Journal of Science and Mathematics Education*, *20*(8), 1949–1972. https://doi.org/10.1007/s10763-021-10227-5

Advancing Mobile App Development and Generative AI Education through MIT App Inventor

David Y. J. Kim[1], Anqi Zhou[1], Yasuhiro Sudo[2], Kosuke Takano[2]
[1]Massachusetts Institute of Technology
[2]Kanagawa Institute of Technology

ABSTRACT

This study assesses the MIT App Inventor's effectiveness in instilling Computational Thinking and Generative AI skills through a five-day workshop focused on mobile app development. With its block-based coding system, App Inventor proved to be user-friendly, significantly enhancing programming accessibility for beginners. Participants experienced a marked increase in confidence and expressed enthusiasm for applying these new skills in real-life contexts, exemplifying the principles of Computational Action. The results underscore the potential of App Inventor as a valuable educational resource, fostering technical aptitude and innovation among an international array of students.

KEYWORDS

MIT App Inventor, Computational Action, Generative AI

1. INTRODUCTION & BACKGROUND

This paper investigates how technology's transformative influence can be democratized through educational initiatives like block-based programming platforms such as Scratch and MIT App Inventor, the latter enabling users to create mobile apps through a visual interface (Maloney et al., 2010; Wolber, Abelson, and Friedman, 2015; Patton, Tissenbaum, Harunani, 2019). While effective in piquing students' interest in app development (Perdikuri, 2014; Grover and Pea, 2013), the predominance of U.S.-based evaluations leaves a gap in understanding its global impact. Addressing this, the paper describes a workshop in Japan, with participants who typically had no coding experience, providing insight into the global potential of block-based coding for fostering computational thinking.

1.1. Computational Thinking to Computational Action

Computational Thinking (CT) is a critical skill in education, akin to traditional literacy, emphasizing problem-solving and logic that transcends programming (Wing, 2006). As CT fosters creativity and critical thinking, it prepares students for a digitized future (Kong and Abelson, 2022). Computational Action extends CT into the real world, where learners apply abstract concepts to develop tangible digital solutions through an iterative, collaborative process (Tissenbaum, Sheldon, and Abelson, 2019). This educational approach enhances students' understanding and confidence, preparing them for future challenges in technology and innovation (Du et al., 2023).

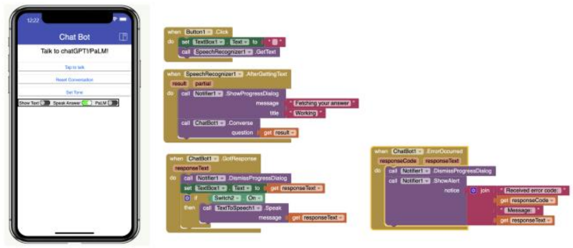


Figure 1. Example of App Inventor's new Gen AI component

1.2. Educating Students about Generative AI

Generative AI refers to artificial intelligence systems that can generate new content, ideas, or data that are novel and not merely a reshuffling of existing information. This field has seen a significant surge in both interest and development in recent years, primarily due to advances in machine learning and neural network technologies (OpenAI, 2016). Generative AI holds the capacity to profoundly transform numerous facets of human society, bringing with it a spectrum of both positive and negative impacts. It is crucial for an increasing number of people to not only become aware of this transformative technology but also to possess the skills and understanding necessary to integrate it into their daily lives effectively. The importance of educating about these technologies becomes increasingly critical. Education in generative AI not only involves understanding the technical workings of these systems but also encompasses a broader comprehension of their ethical, societal, and practical implications (Sharples, 2023). Recently, the MIT App Inventor acquired an innovative addition to its platform - a chatbot/imagebot component (Figure 1). This new feature abstracted the integration of advanced generative AI models, like OpenAI's ChatGPT and Dall-E (Shi et al. 2020), into mobile applications built with App Inventor. With just a few programming blocks, developers can now tap into the power of these AI models, opening a wide range of possibilities for app functionality.

2. METHOD

The workshop titled "Harnessing Generative AI in Mobile Application Development" was conducted at the Kanagawa Institute of Technology. It spanned five days, with each session lasting three hours. The participant group comprised 23 in-person students at the Kanagawa Institute of Technology and 60 to 100 remote students, primarily students from Malaysia and Indonesia.

<table>
<tr><th></th><th>DAY 1</th><th>DAY 2</th><th>DAY 3</th><th>DAY 4</th><th>DAY 5</th></tr>
<tr><td>0 min ~ 30 mins</td><td rowspan="2">Introduction to MIT App Inventor</td><td>Talk2Me Tutorial</td><td rowspan="2">Digital Doodle Tutorial</td><td rowspan="6">Generative AI class: Learning Intutiion and the design of generative AI neural networks</td><td rowspan="6">Student Presentation</td></tr>
<tr><td>30 mins ~ 1 hour</td><td rowspan="2">TodoList Tutorial</td></tr>
<tr><td>1 hour ~ 1 hour 30 mins</td><td>Hello Purr Tutorial</td><td>Introduction to Dall-E</td></tr>
<tr><td>1 hour 30 mins ~ 2 hours</td><td rowspan="2">MoleMash Tutorial</td><td>Introduction to ChatGPT</td><td rowspan="3">Dall-E with App Inventor</td></tr>
<tr><td>2 hours ~ 2 hours 30 mins</td><td rowspan="2">ChatGPT with App Inventor</td></tr>
<tr><td>2 hours 30 mins ~ 3 hours</td><td>Extra Questions</td></tr>
</table>

Figure 2. Curriculum of the workshop

The workshop's primary objective was to introduce students, many of whom had minimal to no experience in coding to the basics of mobile application development using App Inventor. A special emphasis was placed on the integration of generative AI components, showcasing the potential of block-based coding in teaching computational thinking and practical application skills.

As shown in Figure 2, the first three days of the workshop were dedicated to hands-on tutorials in App Inventor, focusing particularly on utilizing its new chatbot and imagebot components. These sessions were designed to provide step-by-step guidance, enabling students to become familiar with block-based coding and the essentials of mobile app creation. On the fourth day, the workshop shifted its focus to the foundational concepts of generative AI. This segment included both theoretical and practical elements, aiming to enhance students' understanding of how generative AI operates and how it can be incorporated into mobile applications. This was particularly relevant given the use of AI components in the App Inventor activities. The workshop culminated on the fifth day with student presentations. Each participant or group was tasked with presenting a simple mobile application they had developed using App Inventor, which incorporated elements of generative AI. This session provided an opportunity for students to demonstrate their understanding and creative application of the skills acquired during the workshop.

An assessment of the workshop's effectiveness was primarily based on the feedback provided by the students and the evaluation of the projects presented on the final day. These projects served as a practical indicator of the student's grasp of the concepts and skills imparted during the workshop. The IRB approval was obtained from Kanagawa Institute of Technology, ensuring that all research methods, participant recruitment, and data handling procedures complied with ethical standards and regulatory guidelines.

3. RESULTS

The students' final presentations were particularly impressive, considering that most of them had never heard of MIT App Inventor before the workshop and for many, English was not their primary language. Despite these challenges, they showcased remarkable ingenuity in integrating generative AI with mobile application development into their everyday lives. For instance, highlighted in Figure 3, a standout project was an app developed by a student using a chatbot to determine a random 'lucky color'. This color then inspired the generation of images of items in that hue, along with information on where to find these items. The student noted, "This app helps me choose the color of my shirt each day" brilliantly demonstrating the practical use of chatbot and imagebot functionalities. This example underscores the students' capacity to creatively utilize AI tools, significantly enhancing their daily routines and decision-making processes, all achieved within the context of navigating a new programming language and working in a non-native language.

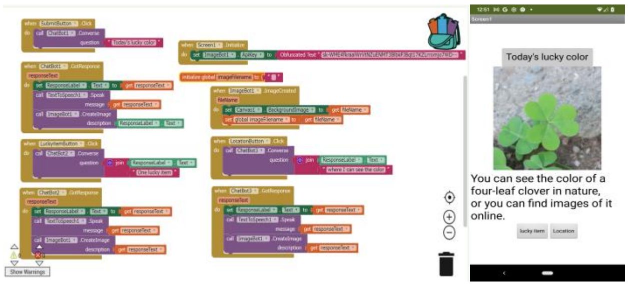


Figure 3. Example of an app a student created.

Did students become more confident in programming?

The MIT App Inventor workshop significantly reshaped participants' perceptions of programming over five days. Those with prior interest noted minimal change, yet for others, the workshop demystified programming's complexity, showcasing its simplicity and accessibility. This led to a noticeable boost in confidence among attendees, particularly those who had viewed programming as an insurmountable challenge. As one participant reflected, they transitioned from being overwhelmed to developing an active interest in app development.

The workshop's user-friendly, block-based programming was a game-changer, igniting enthusiasm and dispelling the notion that programming is inherently difficult. It introduced a novel, less intimidating approach to coding that contrasts sharply with traditional line coding, making software development more tangible and engaging, especially for novices. Attendees appreciated the straightforwardness of block programming, which required no prior experience, and found that it provided a clear understanding of essential programming concepts like event handling and data storage.

Moreover, the workshop opened new vistas for integrating AI in app development, fostering an interest in programming and AI. Participants were exposed to visual programming and the use of pre-built components, revealing the broad potential and adaptability of AI in various applications. This exposure not only illuminated new programming methodologies but also underscored the importance of innovative educational tools in making technology more approachable and versatile.

3.1. Was App Inventor an effective tool to learn?

Participants unanimously lauded MIT App Inventor for its user-centric, accessible interface, highlighting its appeal to novices and those with minimal coding background. Its simplicity, a stark contrast to traditional coding approaches, stood out as a significant benefit. The platform's block-based, drag-and-drop interface was celebrated for demystifying the app development process, as encapsulated by one participant's remark, "It is easy to tinker around with blocks, making programming far less daunting than traditional line coding."

The ease with which users could navigate and utilize App Inventor was a recurring theme among feedback. Its direct, no-frills functionality facilitated a seamless and swift app creation experience, devoid of the complexities often associated with coding. The platform's design, inherently accommodating to those without a coding pedigree, enables the swift and straightforward development of mobile applications. This accessibility is pivotal, positioning App Inventor as an invaluable resource across a broad spectrum of users, particularly those venturing into programming for the first time. Moreover, the platform's intuitive structure allows users to quickly comprehend both the logic behind app development and its design aspects. This feature was especially attractive to participants who, despite finding traditional coding barriers, were keen on venturing into mobile app development.

Also, some praised the geometric tinkering process of App Inventor. One student noted *"The workshop ignited my interest in programming, particularly because I tend to avoid tasks that require extensive memorization, like learning a programming language. The transformation of programming into a puzzle-like format simplified the learning process for me, allowing me to grasp the underlying concepts of the project more intuitively"*. The visual nature of App Inventor, where coding is akin to solving puzzles, was highlighted as a feature that enhances learning and retention, especially for those who struggle with writing code from memory. The platform was also lauded for its ability to facilitate understanding of technological developments and for making programming a more approachable and enjoyable experience.

Furthermore, the platform was recognized for its efficiency in frontend development and its broad functionality, supporting various features needed for smartphone application creation as one mentioned *"The breadth of functionality that allows for the implementation of a complete set of functions needed to create a smartphone application, as well as support for external hardware such as pose estimation, ChatBot, cloud, Lego, etc."* The convenience of real-time programming checks and the reduced need for high-end equipment were also mentioned.

3.2. What can App Inventor improve?

The feedback from participants on MIT App Inventor was varied, focusing on enhancements in user interface (UI) design, additional features, and educational resources.

UI/UX Design Improvements: Several respondents suggested more flexibility and customization options in the UI design of the platform. This included a desire for more UI components and the ability to edit code directly for customizing UI and logic. Improvements in UI/UX design were a recurring theme, with suggestions like a more user-friendly interface and the introduction of features like dark mode.

Enhanced Features: Participants expressed interest in seeing more advanced features in App Inventor. Specific suggestions included improved functionality for the chatbot and imagebot components, image recognition AI, and an in-built emulator for quick app testing. Some users also requested more variety in components for editing user interfaces and a desire for the platform to support hardcoding.

Educational Resources: Requests for more comprehensive educational resources were common. This included more advanced tutorials, both in video and PDF formats, complete documentation about the blocks, and additional tutorials on diverse topics, including game development. The idea of making tutorials more accessible and inclusive for various learning environments was also highlighted.

Accessibility and Language Support: Enhancements in accessibility features, such as Japanese language support and a clearer display of warnings and commands, were mentioned. Suggestions for an offline mode and improvements in the website's UI/UX were also proposed.

Performance and Bug Fixes: Addressing performance issues and fixing bugs were noted as areas for improvement. This includes dealing with issues where blocks do not display or the display freezes.

Community and Collaboration Features: Some participants suggested features to facilitate sharing and collaboration directly within the app, such as enabling multiple users to work on an app simultaneously and hosting activities to promote App Inventor's growth globally.

Transparency in Coding: A few responses indicated an interest in seeing the block-based code translated into standard programming language notation, which could help those interested in transitioning to traditional coding.

4. CONCLUSION

This study assesses MIT App Inventor's role in teaching Computational Thinking and Generative AI to a global student audience. Over a five-day workshop that combined theoretical learning with practical exercises, students were introduced to key programming principles, using App Inventor to learn coding visually and interactively, a method that proved especially beginner friendly. Analysis indicates a strong preference for the platform's block-based programming, citing improved accessibility and a boost in confidence, which in turn inspired students to apply these skills practically. The student-created applications during the workshop demonstrate not only a solid understanding of programming concepts but also their application for personal and community betterment, embodying the concept of Computational Action. The success of App Inventor in this educational context emphasizes its value in enhancing technological understanding and creativity among diverse student groups, offering insights into the tool's potential for broad application in education globally.

5. REFERENCES

John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. ACM Trans. Comput. Educ. 10, 4, Article 16 (November 2010), 15 pages. https://doi.org/10.1145/1868358.1868363

David Wolber, Harold Abelson, and Mark Friedman. 2015. Democratizing Computing with App Inventor. GetMobile: Mobile Comp. and Comm. 18, 4 (October 2014), 53–58. https://doi.org/10.1145/2721914.2721935

Katerina Perdikuri. 201). Students' Experiences from the use of MIT App Inventor in classroom. In Proceedings of the 18th Panhellenic Conference on Informatics (PCI '14). Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2645791.2645835

Shuchi Grover and Roy Pea. 2013. Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. In Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13). Association for Computing Machinery, New York, NY, USA, 723–728. https://doi.org/10.1145/2445196.2445404

Jeannette M. Wing. 2006. Computational thinking. Commun. ACM 49, 3 (March 2006), 33–35. https://doi.org/10.1145/1118178.1118215

Siu-Cheung Kong, Harold Abelson. (2022). Computational Thinking Education in K–12: Artificial Intelligence Literacy and Physical Computing. The MIT Press

Mike Tissenbaum, Josh Sheldon, and Hal Abelson. 2019. From computational thinking to computational action. Commun. ACM 62, 3 (March 2019), 34–36. https://doi.org/10.1145/3265747

Xiaoxue Du, Robert Parks, Selim Tezel, Jeff Freilich, H. Nicole Pang, Hal Abelson, and Cynthia Breazeal. 2023. Designing a Computational Action Program to Tackle Global Challenges. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2 (SIGCSE 2023). Association for Computing Machinery, New York, NY, USA, 1320. https://doi.org/10.1145/3545947.3576267

Patton, E.W., Tissenbaum, M., Harunani, F. (2019). MIT App Inventor: Objectives, Design, and Development. In: Kong, SC., Abelson, H. (eds) Computational Thinking Education. Springer, Singapore. https://doi.org/10.1007/978-981-13-6528-7_3

OpenAI. (2016) Generative Models. Generative models (openai.com)

Sharples, M. (2023). Towards social generative AI for education: theory, practices and ethics. Learning: Research and Practice, 9, 159 - 167.

Shi, Z., Zhou, X., Qiu, X., & Zhu, X. (2020). Improving image captioning with better use of captions. arXiv preprint arXiv:2006.11807

STEM as a Whole: Designing an Out-of-School STEM Program to Empower Underrepresented Minority Girls in STEM

Chaoyi Wang[1], Yilun Zhao[1*]
[1]College of Child Development and Education, Zhejiang Normal University, Hangzhou, China
cywang@zjnu.cn, zhaoyilun@zjnu.cn

ABSTRACT
This study developed and implemented an out-of-school (OST) STEM program, named the GEM STEM program, specifically designed for underrepresented minority girls in the southern United States. Utilizing a qualitative research design, the study investigated the learning experiences of forty-one underrepresented minority girls, ranging from middle to high school (grades 7-12), all of whom have engaged in this program for more than two academic years. Following the Social Cognitive Career Theory (SCCT) model, the results revealed the OST program benefits girls by creating a conducive learning environment, encouraging STEM participation, and facilitating meaningful interactions with adult mentors. Key factors include engaging conversations with professionals, collaborative opportunities, and the integration of diverse subjects. Participants also shared their perspectives on gender bias and common stereotypes in STEM fields, with socioeconomic status coming up as a barrier to their professional growth.

KEYWORDS
STEM Education, After-school STEM Program, Underrepresented Minorities in STEM Education, Pathways to STEM Career, Comparative Analysis of STEM Education

1. INTRODUCTION
In line with the rapidly changing technological era, the United States places much effort into refining and promoting STEM education. The U.S. government, along with Congress, State legislatures, and school STEM programs, has implemented extensive efforts to reform K–12 STEM education and cultivate the next generation of skilled scientists, engineers, technicians, and science and mathematics educators (Kennedy & Odell, 2014). All efforts emphasize the importance of preparing youth to apply knowledge and skills to solve problems, make sense of information in future careers, and use them in real-life situations.

As the largest ethnic minority in the United States, Latinos encounter unique challenges in STEM education, demonstrating lower academic achievement rates, reduced degree attainment, limited educational opportunities, and a diminished presence in STEM professions (Taningco et al., 2008). Gender disparities persist among Latinos in STEM careers, particularly affecting Latina girls. Influenced by ethnic identities and socioeconomic status, Latina girls may face challenges in STEM interest, perceived ability in STEM fields, and career motivation (Modi, Schoenberg, and Salmond, 2012). Cultural influences, such as the preference for attending schools close to home, further impact educational choices for Hispanic youth (Dowd, Malcom, & Bensimon, 2009).

To address these challenges, STEM enrichment programs emerge as a pivotal solution. These programs, including afterschool activities, summer camps, and competitions, bridge educational gaps by providing content knowledge, fostering real-world connections to STEM, and promoting informal learning opportunities (Young and Young, 2018). Research has demonstrated that STEM enrichment programs boost students' interest in STEM content and careers (Mohr-Schroeder et al., 2014), STEM self-efficacy (Leonard et al., 2016; Mann et al., 2015), school connectedness, self-identity, and excitement about STEM subjects (Mohr-Schroeder et al., 2014; Yanowitz, 2016). When looking into the results of girl-focused STEM programs, research demonstrated positive outcomes in girls' self-efficacy in STEM, interest in STEM-related subjects, and excitement of STEM-related careers (Levine, Serio, Radaram, Chaudhuri, & Talbert, 2015).

Despite these positive findings, there are fewer programs specifically designed to cater to the needs and interests of Latina girls, aiming to promote their active participation and success in STEM fields. Prior research was limited in terms of methodological issues to study OST STEM programs. Most study samples used only single-item survey measures to assess the importance of STEM summer programs on student outcomes, which have a low level of measurement reliability and validity (Saw et al., 2019). Thus, this study employs a qualitative case study design to delve into Latinas' experiences participating in an OST STEM program, aiming to offer a comprehensive understanding of their journey in the STEM educational landscape.

2. SITE AND SETTINGS

2.1. Overview of GEMS OST STEM Program
The GEMS (abbreviated from Girls in Engineering, Mathematics, Science) STEM program, initiated in 2015 at a South Texas private university, strives to inspire and empower more girls from underrepresented minorities to pursue careers in STEM. Providing complimentary access to robotics and STEM learning opportunities, the program has established an all-girls learning environment, with a special focus on those from underrepresented minority backgrounds in low-income families. The program offers a diverse range of initiatives, spanning from a 2-week summer camp to a 6-week research camp and a year-round program.

The miniGEMS program, a cost-free two-week summer camp, emphasizes the fields of engineering and programming (Figure 1). It aims to expose middle school girls to STEM disciplines through hands-on experiences, including robotics, computer programming, graphic design, and inspiring guest speakers. Over the years, miniGirls has evolved from a one-week STEM camp to a two-week STEAM and programming camp, integrating arts to accommodate a broader range of interests. The program extends its impact through miniRobots, a year-round robotics club that serves as an extension of miniGirls, aiming to enhance STEM interests and programming skills for middle school girls.

Table 1. Overview of the GEMS STEM Program

Grade Level	Length	Core Curriculum
miniGEMS (6th to 8th) megaGEMS (9th to 12th)	Two-week	Robotics Programming Game design Hands-on STEM activities Career Exposure Guest speakers Nutrition and gardening Art activities
megaResearch (9th to 12th)	Six-week	Research-based projects Academic writing course Presentation training Other STEAM hands-on activities
miniRobots (6th to 8th)	Full year	Weekly hands-on robotic Programming practice

Launched in 2019, megaResearch is a six-week summer camp designed for juniors and seniors in high school. The program borrows the NSF REU model but introduces high school girls to faculty-guided research on a STEM project and culminating in a research paper, presentation, and poster. The goal of the camp is to provide high school girls with valuable research learning experience, which in turn may improve their research skills, STEM interests, and 21st century skills. Participants engage in practical research projects under faculty guidance, utilizing a Project-Based Learning (PBL) approach that grants girls flexibility in managing their time, encouraging their interest in pursuing STEM majors and independence.

For example, the goal of the Drone project (Figure 2) was to use Unity3D and a Geomagic touch haptics device to learn how to design a simulation of explosive ordnance disposal (EOD). The main goals of this project were to: (1) introduce students to cutting-edge technologies such as haptics; (2) provide them with practical experience in using the Unity3D game engine while creating simulations using the C# programming language; and (3) create a practical real-world application for training bomb disposal squads. The purpose of this project was to give students hands-on experience and to investigate the options and possibilities for building a haptics-based simulation. This project also helped the girls understand the hazards bomb disposal squads encounter and what is the necessary steps for developing this application, including touch, feel, and sense capabilities.

3. RESEARCH DESIGN

3.1. Research Questions

RQ1: What are participants' learning experiences in the GEMS OST STEM program?

RQ2: What are the influences of participating in the GEMS OST STEM program on girls' career interests?

3.2. Sampling and Data Collection

The study's sample comprised 41 middle and high school girls, ranging from 7th to 12th grade. These participants had extensive involvement in the program, with each having participated in at least two rounds. This included options such as two instances of a 2-week summer camp, a 6-week research camp, or a comprehensive year-round program, all characterized by a high level of engagement. All participants identified their ethnicity as Hispanic.

Semi-structured interviews with open-ended questions are the major source to gain participants' experiences within the research scope. The researcher used three-interview series and treated all interviews as opportunities to have participants reconstruct their experiences within the study contexts. Considering the participants were all middle school and high school students, each interview took fifteen to twenty minutes in this study. The researcher is also preparing for other rounds of parents and teachers interviews if needed. The study protocol follows the federal government's "Common Rule" for the protection of human subjects and was approved by the University Ethical Review Board for the Humanities and Social and Behavioral Sciences at the institution of the first author.

3.3. Data Analysis

Followed by Stake (1995) and Yin's (1993) data analysis strategies, a constant-comparative method (Corbin & Strauss, 2008) was employed to compare participants' learning experiences. The interview data were audio-recorded and transcribed using Temi software. To ensure accuracy, the researcher made the final corrections to the transcripts. Additionally, during data collection, the researcher created short memos for each participant, coding each memo in Dedoose. The analysis began with content analysis, focusing on analyzing text and documents. Subsequently, thematic analysis was conducted, with a special emphasis on developing themes. The final phase gave more attention to the selection of themes.

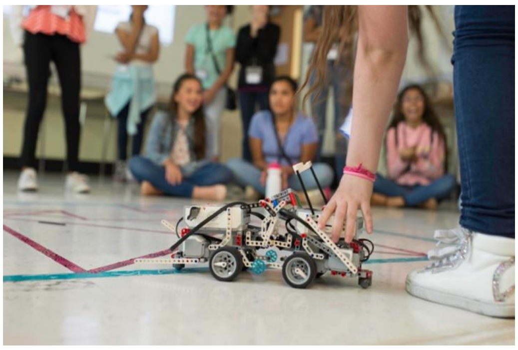


Figure 1. Programming in miniGEMS Program

Figure 2. Drones Research Project in megaResearch Program

4. RESULTS

4.1. Learning Experiences in GEMS

Girls participating in the GEMS STEM program had positive and varied learning experiences. The focus on hands-on activities and projects, as described by Amanda, *"The camp is like a science and math based club where we do different projects and it's, it's fun. We have to be creative and think outside the box of like that."* The diverse levels and types of the program offered distinct learning experiences for participants. This variation was deemed beneficial for high school girls entering postsecondary education.

The program created an all-girls learning environment that differed from formal schools, providing necessary materials like robots and drones for hands-on experience. The OST learning environment emphasizes the broader opportunities and behavioral options for middle school girls. Girls felt more comfortable making mistakes and trying new ideas in this setting, contrasting with the constraints of formal schooling. As Clara demonstrated, *"I feel more open here and I feel like, oh this is, I'm comfortable here. I guess I feel more relaxed. And at school I'm more like tenants. I'm more close and I don't, I just listened to the teacher and that's it. Like I don't really like talking but here I feel better. I feel like I have a voice."*

Communication and cooperation were essential components of the program, fostering effective teamwork. Girls worked together to solve problems and develop necessary skills, leading to significant improvements in solutions. For example, one girl described her experience of working in teams by saying, *"For the first one, it was my experience or like something that describes miniGirls summer camp. Um, and overall I learned how to work with people and friendship and just like working together to like come up with the new ideas."* Mentors played crucial roles in the program, providing positive mentorship, and creating a supportive learning environment. The level of mentor involvement varied, with heavy involvement potentially limiting students' engagement, but overall, mentoring contributed to a positive student-mentor relationship and a sense of community.

Students experienced personal growth through the program, gaining familiarity with activities and overcoming initial challenges. Elisa mentioned the robotic competition was less stressful for her and she achieved better performance in the competition by saying "*I've gotten like more calm about the robots and it felt easier this [second] year when we were like putting together the EV3 and doing the coding the first time"*. Longer-term participants, such as those attending two summer camps, exhibited increased calmness and better performance, indicating personal development. Some girls returned as mentors, campers, or volunteers, further contributing to the program's success and forming stronger bonds through shared experiences. Overall, the GEMS program facilitated positive learning experiences, personal growth, and a supportive community for Latina Girls.

4.2. Influences of the GEMS Program on Girls' Career Interests

This study on the Girls in STEM program revealed its positive impact on students' STEM learning and the development of their interests in STEM fields. The program fostered a fun learning environment, covering various STEM concepts such as robotics, programming, and graphic design. Middle school girls improved their math efficacy, while high school participants gained familiarity with a college environment, motivating them to pursue additional STEM activities and classes. The exposure to different STEM concepts through presentations by guest speakers broadened participants' perspectives and influenced their career interests, fostering aspirations in engineering, coding, and related fields.

The data also highlighted the influence of gender disparities and stereotypes on girls' career choices. Many girls in this study realized the gender bias and stereotypes towards women in STEM. As one girl expressed her perspectives of gender bias by saying: *"With them being sexist, it's kind of bad because we females we probably don't have as much style as the men cause due to them like having been very picky. We're also picky but we like, work harder to achieve that cause just 'cause people think that men do better than women when actually women work harder. We, we try harder."*

Despite facing gender bias in STEM, participants expressed a willingness to consider STEM-related careers as alternative choices. Career aspirations centered around helping others, with girls aspiring to become dermatologists, therapists, and pursue careers that contribute positively to society. The findings indicated that the GEMS Program contributes to challenging stereotypes and encourages girls to explore diverse career paths, creating a pathway for them to pursue STEM-related goals.

5. DISCUSSION

The results of this study highlight the beneficial effects of the GEMS OST program on raising participants' aspirations for STEM careers. However, it raises the crucial question: ***What can researchers and practitioners do to sustain and enhance the STEM interests of underrepresented minority girls?***

As middle school and high school girls are at an age where they desire to learn new things, it is beneficial for them to have long-term STEM exposure so they can explore various subjects and develop their interests in STEM learning. Creating a supportive learning atmosphere is crucial for helping students share their viewpoints and acquire necessary skills, which may impact their interests, self-efficacy, and professional development. Similar to numerous middle schools, engineering is not offered as a course option for students. In the classroom, engineering skills are hardly developed at all. Certain after-school programs incorporate multiple disciplines and could serve as an additional source of education beyond what is typically offered by schools.

Enhancing program outcomes also requires programmatic structure and continuous emotional and educational support. Based on the programmatic structure, the OST STEM program for middle school students could offer more engaging and interactive activities, while the high school curriculum may place more emphasis on education and professional experiences. It is significant to remember that middle school girls are going through physical, mental, emotional, and social changes as they make the move from elementary to middle school. They are also growing in terms of self-identification and self-awareness at this age.

Therefore, it would be advantageous for girls to receive assistance from OST program facilitators to enhance their academic learning and develop their social and emotional competencies. These facilitators can take on the role of teachers, encouraging students to learn and stimulating their curiosity in math and science. Students' motivation and efficacy may decline in the absence of support during the learning process, which also results in negative emotions. Programs for STEM OSTs may need to concentrate more on "how much" support than on "how many" activities and topics they can cover.

It is also critical to recognize how socioeconomic status and cultural backgrounds impact the educational and professional choices of Latinas. Gender bias and stereotypes in STEM are pervasive, and aligning social environments such as school, family, community, and the program can better support and encourage girls in persisting and developing interests in STEM fields. Collaborative efforts can foster an inclusive environment that empowers Latina students to pursue STEM education and careers despite existing societal challenges.

6. Funding

This work was supported by the Youth Project of Humanities and Social Sciences Financed by Ministry of Education of China (22YJC880072); General Scientific Research Project of the Zhejiang Provincial Department of Education (Y201943047), and Education Scientific Planning of Zhejiang Province (2022SCG383);

7. REFERENCES

Corbin, J., & Strauss, A. (2008). Strategies for qualitative data analysis. Basics of Qualitative Research: Techniques and procedures for developing grounded theory (3rd ed., pp. 65-86). Thousand Oaks, CA: Sage.

Dowd, A. C., Malcom, L. E., & Bensimon, E. M. (2009). Benchmarking the success of Latina and Latino students in STEM to achieve national graduation goals. Los Angeles: University of Southern California. Retrieved from http://cue.usc.edu/ news/NSF-Report.pdf

Kennedy, T. J., & Odell, M. R. L. (2014). Engaging students in STEM education. Science Education International, 25(3), 246-258.

Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T., & Almughyirah, S. (2016). Using robotics and game design to enhance children's self-efficacy, STEM attitudes, and computational thinking skills. Journal of Science Education and Technology, 25(6), 860–876.

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.

Levine M., Serio N., Radaram B., Chaudhuri S., & Talbert W. (2015). Addressing the STEM gender gap by designing and implementing an educational outreach chemistry camp for middle school girls. Journal of Chemical Education, 92(10), 1639-1644.

Modi, K., Schoenberg, J., & Salmond, K. (2012). Generation STEM: What girls say about science, technology, engineering, and math. New York, NY: Girl Scouts of the USA.

Mohr-Schroeder, M. J., Jackson, C., Miller, M., Walcott, B., Little, D. L., Speler, L., & Schroeder, D. C. (2014). Developing middle school students' interests in STEM via summer learning experiences: See Blue STEM camp. School Science & Mathematics, 114(6), 291–301.

Saw, G. K., Swagerty, B., Brewington, S., Chang, C. N., & Culbertson, R. (2019). Out-of-School time STEM program: Students' attitudes toward and career interests in mathematics and science. International Journal of Evaluation and Research in Education, 8(2), 356-362.

Taningco, M. T. V., Mathew, A. B., & Pachon, H. P. (2008). STEM professions: Opportunities and challenges for Latinos in science, technology, engineering, and mathematics. Los Angeles, CA: Tomas Rivera Policy Institute.

Young, J., & Young, J. (2018). The structural relationship between out-of-school time enrichment and black student participation in advanced science. Journal for the Education of the Gifted, 41(1), 43-59.

Fostering Computational Thinking in preK-12 Education: A Bibliometric Analysis and Visualization of the Literature

Hanlingxiu Li[1], Jinghan Xu[2], Na Zhang[3*]
[1,2,3]Capital Normal University, China
2233102013@cnu.edu.cn, xujinghan202203@163.com, zhangna@cnu.edu.cn

ABSTRACT

Computational thinking is vital to the abilities of children to solve problems effectively and overcome increasing challenges efficiently in this technological era. Using criteria of publication date, source, and relevance, 576 computational thinking (CT) publications were identified in the WoS database from 2013 to 2023. Bibliometric analysis and visualization were conducted on this literature set using VOSviewer and CiteSpace to map CT research hotspots, trends, frequency and centrality of themes, as well as gaps. The bibliometric analysis in this study revealed several key findings:(1) Despite strong advocacy for widespread CT integration into K-12 STEM curricula, empirical guidance lags on translating theories into scalable student learning. (2) CT curricula must constantly evolve to reflect the stages of development of young learners while also leveraging educational robotics and emerging technologies such as generative AI. (3) As keywords like "early childhood education" and "games" surge, future work should harness the interactive, scaffolded learning unique to developmentally-oriented games for CT skills. (4) Embedded, automated assessments show promise for monitoring authentic applications of CT tools within regular classroom environments.

KEYWORDS

Computational Thinking, K-12, Preschool, Educational robotics, Bibliometrics Analysis

1. INTRODUCTION

Computational Thinking (CT) refers to the ability to understand the underlying notions and mechanisms of digital technologies to formulate and solve problems. Wing(2006) defined CT as one of the most important problem-solving skills, it is essential to differentiate and relate between CT and other fields, such as programming and coding for the reason that applying the CT method to different scientific areas was the impetus for developing computational science (Kong & Abelson, 2022). In this study, scientometric analysis contributes to comprehend data which can encourage researchers to (1) get a view of the relationship between CT and other disciplines; (2) identify influences and challenges of educational robots to children' CT curricula in different contexts; (3) find innovative methods like game-based learning to explore the potential advantage to the cultivation of CT; and (4) consider the formative framework of CT assessment of children.

2. METHOD

The database utilized in this research is the Web of Science (WOS) Core Collection, the search criteria were restricted to three editions: "SCI-EXPANDED", "SSCI" and "ESCI". A total of 618 documents were reviewed, the terms were searched using the edited query "TS=(("child" OR children OR student NOT higher education)AND("computational thinking" OR "programming" OR coding) AND("Artificial Intelligence" OR robot OR chatbot OR ChatGPT))". Finally, a total of 539 records were retrieved from the literature database. This study uses the quantitative method of bibliometrics to explore the evolution of the literature in this research area by counting the documents related to children's CT with the use of the data processing function of the CiteSpace (6.2.R2).

3. RESULT

3.1. Bibliometric and visualization analysis of keyword co- occurrence

A total of 1552 keywords were relevant to the analysis. The main keywords were computational thinking, robotics, programming and coding. Studies in this cluster suggests that utilizing technologies related to educational robots to assist children in the programming process and designing constructed scenarios in which children use computational thinking to collaborate with robots are research directions that should be focused on.

3.2. Analysis of keyword emergence based on CiteSpace

In this study, keyword emergence analysis was used to understand the change in the number of citations of hot words related to children's CT development at a certain time period. There is a greater emphasis on refining the CT education of children through interdisciplinary games, with innovatively designed instructional paradigms to intrigue students in further pursuit of domain learning (Kong SC, 2022), developing children's problem-solving skills in the process of learning, as well as the competence to plan and evaluate solutions.

4. DISSCUSSION

4.1. The challenges of integrative CT to K-12 STEM education

CT is crucial for STEM education, mathematics and science learning as there are many connections between CT and compentences required in disciplines. According to the OECD (2023), the 2022 PISA framework for mathematics incorporates CT as one of the fundamental mathematical capabilities students need. The mathematic curricula relevant with CT are conducted in interdisciplinary educational settings, which implies that the competency of designing STEM teaching and learning activities is indispensable(Hsu, Irie & Ching, 2019; Ye et al., 2023).

4.2. Evolving Computational Thinking Curricula for Educational Robotics and Children's Developmental Stages

Educational robotics has gained significant attention in different education contexts, programming curricula are mostly implemented using plugged-in or unplugged programming(Sigayret et al., 2022). When robots engaged in the design of children's CT curricula based on Scratch, they enable children to better understand how to personalize the process of learning(Bers et al., 2019; Pugnali, Sullivan& Bers, 2017). In subsequent studies we should take into account children's knowledge, acceptance and willingness to operate the robot and guide them to solve problems independently.

4.3. The Innovative Construct of Game-Based Learning for Improving CT skills

With the development of science and technology and the emergence of new educational methods, the interactive nature of games can promote students' initiative engagement and reduce the cognitive load during learning (Sigayret et al., 2022). It's necessary to clarify how to innovatively design and implement game-based CT teaching appropriate to students characteristics(Hsu, Chang & Hung, 2018). However, there are studies did not yield significant positive results for some CT concepts, including sequence, conditions, and loops(Howland & Good, 2015).

4.4. Embedded Assessments for Authentic Computational Thinking Performance in Classroom Contexts

Assessing the CT performance of children is crucial for educators to analyze their development level of CT abilities and provide appropriate educational support in real-life situations(Grover&Pea, 2013). Existing research indicates that framework of accessing students' CT skills are systematic, which can be investigated further from the perspective of complex education settings, and report reliability and validity evidence to confidently qualify the assessment.

5. REFERENCES

Bers, M. U., González-González, C., & Armas–Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education, 138*, 130-145.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher, 42*(1), 38-43.

Howland, K., & Good, J. (2015). Learning to communicate computationally with Flip: A bi-modal programming language for game creation. *Computers & Education, 80*, 224-240.

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.

Hsu, Y.-C., Irie, N. R., & Ching, Y.-H. (2019). Computational Thinking Educational Policy Initiatives (CTEPI) Across the Globe. *TechTrends: Linking Research & Practice to Improve Learning, 63*(3), 260–270.

Kong, S. C., & Abelson, H. (Eds.). (2022). *Computational thinking education in K-12: Artificial intelligence literacy and physical computing.* MIT Press.

Kong, S. C. (2022). Problem formulation in computational thinking development for nurturing creative problem solvers in primary School. *Education and Information Technologies, 27*(9), 12523-12542.

Organization of Economic Co-operation and Development. (2023). *PISA 2022 Mathematics Framework in PISA 2022 Assessment and Analytical Framework.* Retrieved Aug 31, 2023, from https://www.oecd-ilibrary.org/education/pisa-2022-assessment-and-analytical-framework_7ea9ee19-en.

Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The impact of user interface on young children's computational thinking. *Journal of Information Technology Education. Innovations in Practice, 16*, 171.

Sigayret, K., Tricot, A., & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education, 184*, 104505.

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

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education, 10*(1), 3.

Evaluation of Computational Thinking in Chinese Elementary School Teaching: A Critical Review

Song Anyu[1]
[1]Beijing Normal University, China
202221010192@mail.bnu.edu.cn

ABSTRACT

This document critically reviews research on teaching computational thinking in Chinese elementary schools. Utilizing the CNKI database, nine relevant articles were identified and analyzed. The evaluation of computational thinking typically revolves around three dimensions: computational concepts, computational practices, and computational perspective. Nevertheless, challenges such as the ongoing evolution of computational thinking definitions and the insufficient emphasis on process evaluation in assessments remain unaddressed. Recommendations include clarifying teaching objectives, adopting diverse evaluation methods, and emphasizing process evaluation to refine teaching activities. By addressing these issues, the evaluation of computational thinking in primary education can effectively promote students' computational abilities and skills.

KEYWORDS

computational thinking, primary education, evaluation methods

1. INTRODUCTION

Computational thinking serves as the foundational mindset of the information age and occupies a fundamental position in the realm of information technology education. It stands as a focal point in the foundational education of information technology during the primary education stage(Jin et al., 2020). Primary school students are at a crucial stage for forming behaviors and habits. It is essential to give special attention to cultivating their computational thinking skills. Scientifically effective assessment of computational thinking is key to evaluating the effectiveness of cultivation efforts and monitoring the development of students' relevant abilities. By assessing students' levels of computational thinking, teachers can understand their strengths and weaknesses in this area. This understanding enables teachers to formulate targeted teaching strategies to help students improve their computational thinking skills.

Presently, the cultivation of computational thinking, a primary objective within information technology curricula, has attracted significant recognition and focus among educators. However, due to the lack of textbooks and other teaching materials, frontline teachers often face confusion regarding how to assess students' computational thinking skills during the teaching process. This issue poses a challenge to teachers in conducting teaching research activities.

2. RESEARCH DESIGN

This study utilizes the China National Knowledge Infrastructure (CNKI) database to sample relevant academic journals. It uses "computational thinking" as the keyword and "primary education" as the subject for retrieval. The literature screening criteria were studies that implemented teaching in primary schools and evaluated computational thinking. In the end, 9 articles were obtained (as shown in the table below).

Table 1. Literature Screening Results

Number	Title	Evaluation Dimensions of Computational Thinking	Sub-dimensions	Evaluation Method/Tools
1	Research on Internal Mechanism and Teaching Practice of Programming Education for Development of Children's Computational Thinking	Computational Thinking	Decomposition Thinking, Abstract Thinking, Procedural Thinking, Iterative Thinking, Generalized Evaluation Thinking	the Computational Thinking Scale (CTS) adapted by the author
2	The Impact of a Design-based Integrated STEM+C Teaching on Students' Computational Thinking	Computational Thinking	Problem Solving, Creative Thinking, Critical Thinking, Algorithmic Thinking, Cooperative Thinking	the CTS adapted by the author
3	Research on the Influence of Pair-Programming on Elementary School Students' Computational Thinking Based on Learning Style and Partnership	Computational Concepts	None	the Computational Thinking Test CTt)
		Computational Practices		Bebras questions
		Computational Perspective	Self-efficacy	except from scale
			Programming attitude	questionnaire
4	Development and Implementation of Goal-oriented STEM School-based Curriculum in Primary Schools	Computational Thinking	Creativity, Critical Thinking, Problem Solving, Algorithmic Thinking, Cooperative Thinking	the CTS adapted by the author
5	Design and Application of Visual Programming Activities towards Computational Thinking	Computational Concepts	None	Bebras questions
		Computational Practices		Bebras questions
		Computational Perspective		the CTS adapted by the author
6	Research on Precision Teaching Model of Human-Computer Collaboration for the Cultivation of Computational Thinking —— Taking the Sixth Grade Information Technology Class "Silk Road	Computational Concepts	None	excerpt from the CTt
		Computational Practices		classroom observation, interviews, Dr. Scratch
		Computational Perspective	Creativity, Algorithmic Thinking, Collaborative skills，Critical Thinking, Problem Solving,	the CTS adapted by the author

	Breakthrough" as an Example			
7	An Application of a PBL+CT Teaching Model in Primary Mathematics for Cultivating Students' Computational ThinkingTaking "How to Enclose the Largest Area" as an Example	Computational Thinking	Decomposition, Abstraction, Algorithmic Thinking, Critical Thinking, Problem-solving, Collaborative Learning	the CTS adapted by the author
8	An Empirical Study of Scratch Gamification Programming for Cultivating Elementary School Students′ Computational Thinking	Computational Concepts	Flow Control，Data Representation，Abstraction，User Interactivity，Synchronization，Parallelism，Logic	examination tests，student works, Dr. Scratch
		Computational Practices	Incremental Iteration, Testing and Debugging, Reuse and Recreation, Abstraction and Modularization	classroom observation, interviews
		Computational Perspective	Expression and Creativity, Communication and Collaboration, Understanding and Questioning	adjustment questionnaire, classroom observation, interviews
9	Research on DBL Instruction in STEM Courses to Develop Primary School Students' Computational Thinking	Computational Thinking	Creativity, Critical Thinking, Problem Solving, Algorithmic Thinking, Cooperative Thinking	adapted questionnaire based on the CTS and the Creativity Scale

Researcher analyzed computational thinking evaluation methods in the literatures, summarized the characteristics of computational thinking evaluation, aiming to provide assistance for evaluation issues in teaching research and teaching practice.

3. EVALUATION OF COMPUTATIONAL THINKING

3.1. Dimensions of Evaluating Computational Thinking

Brennan proposed measuring computational thinking along three dimensions: computational concepts, computational practices, and computational perspectives (Brennan & Resnick, 2012). Özgen Korkmaz developed a computational thinking scale (CTS). The questionnaire investigates learners' composite computational thinking abilities across five sub-dimensions: creativity, critical thinking, problem-solving, algorithmic thinking, and collaborative thinking (Korkmaz et al., 2017).

These classification methods represent the predominant frameworks for assessing computational thinking across its diverse dimensions. Additionally, the CTS is also used to measure levels of computational concepts and computational practices.

3.2. Categories of Computational Thinking Evaluation

Summative evaluation is a method commonly used in research, and all studies have chosen scales as evaluation tools. Although some computational thinking assessment tools are formative evaluations, they have not been used extensively due to their high correlation with teaching content. Researchers still prefer summative evaluation scales as the method for evaluating computational thinking levels. In addition, qualitative evaluation methods such as interviews and observations are used to collect supplementary data. These data, obtained from formative assessments within student tasks, are rarely regarded as primary analytical data.

4. ISSUES IN EVALUATING COMPUTATIONAL THINKING

4.1. The Understanding of Computational Thinking Affects the Evaluation

The academic community has not yet reached a consensus on the essence of computational thinking. Moreover, some scholars have pointed out that the essence and methods of computational thinking will continue to evolve (Zhang, 2019). Shifts in the conceptual understanding of computational thinking pose substantial challenges to its evaluation in educational settings. Researchers in computational thinking teaching research have utilized existing computational thinking scales. However, they have modified existing scales based on their understanding of computational thinking, which may differ from that of the scale developers (Duo et al., 2022).

In computational thinking teaching research, the development and validation of computational thinking scales are often not the main focus. However, modifying existing scales without validating the changes and using them with only a small sample size makes it challenging to verify the universality of the scales. This approach can indeed have a detrimental effect on the results of teaching research.

4.2. Lack of Process Evaluation

When evaluating computational thinking, the majority of cases rely solely on scale assessments, lacking process evaluation. While teaching research needs to demonstrate the effectiveness of teaching through evaluating computational thinking, this approach overlooks students' thinking processes and strategy choices during problem-solving. As a result, it may not comprehensively reflect students' computational thinking abilities and levels. Some studies utilize classroom observations and interviews as methods to assess computational thinking. However, the analysis of the collected data often lacks depth, failing to fully leverage these findings.

In learning environments, dialogue and communication lead to co-constructed activities that result in cognitive changes. Evaluation activities such as analysis, review, and reflection play a crucial role in the internalization of learners' knowledge and concepts (Zhou, 2018). Researchers need to emphasize process evaluation to promptly identify issues in teaching and refine teaching activities.

4.3. The complexity of evaluation

The complexity and diversity of evaluation content can complicate the evaluation process, making it challenging for evaluators to identify key points and critical indicators. This difficulty can impact the accuracy and consistency of the evaluation results. First, an abundance of evaluation content may prevent researchers from conducting comprehensive analyses of the collected data, especially in terms of in-depth analysis of textual content, which may seem inadequate. Moreover, excessive data collection and analysis can obscure evaluation standards, increasing subjectivity and uncertainty in the evaluation process. As a result, this could diminish the credibility and fairness of the evaluations.

In terms of evaluating computational practices and concepts, although student work and interview materials have been collected, many studies still only scratch the surface, and some research does not collect data on the teaching process.

5. RECOMMENDATIONS

The assessment of computational thinking instruction in elementary schools is not just an important means to gauge the development of students' computational abilities; it's also key to enhancing the quality of education. This process involves a comprehensive understanding of students' computational thinking levels and uses diverse and targeted assessment content and methods to stimulate and foster the continuous growth of students' computational thinking abilities.

5.1. Clarifying Instructional Goals for Computational Thinking

In light of computational thinking's dynamic nature, establishing precise instructional objectives constitutes a crucial preliminary step within the educational process. Teachers should base their specific instructional goals on the core components of computational thinking. These goals should match students' learning needs and cognitive development stages and reflect the broad application of computational thinking in real life. Clearly defined instructional goals help guide the selection and design of teaching activities, ensuring consistency between teaching content and assessment methods, thereby more accurately reflecting students' computational thinking abilities. Indirect evaluation of computational thinking levels through assessing students' grasp of the instructional content can simplify the assessment challenge.

5.2. Utilizing a Variety of Assessment Methods

Cognitive abilities are closely tied to knowledge levels, and the development of thinking abilities often requires a solid knowledge foundation. Relying solely on scales may not fully or accurately capture students' thinking levels. Teachers need to employ a variety of assessment methods, including traditional testing, project-based learning assessments, peer assessments, self-assessments, and teacher observations. By assessing students' knowledge, cognitive abilities, and emotional attitudes from multiple angles, teachers can comprehensively understand students' computational thinking levels, promptly adjust teaching strategies, meet students' individual learning needs, and enhance their problem-solving and innovative thinking skills.

5.3. Establishing Unified Computational Thinking Assessment Standards

The crucial aspect of establishing unified standards for assessing computational thinking is to integrate insights from current research and educational practice to identify core dimensions of computational thinking, such as abstraction, decomposition, algorithmic thinking, critical thinking, and creativity. This step involves an extensive review of literature and a deep understanding of educational practices, aiming to capture the entirety of computational thinking abilities. Following this, based on these core dimensions, a detailed assessment framework should be developed. This framework should define specific assessment criteria for each dimension and provide clear descriptions of levels to ensure the accuracy and consistency of assessments. Additionally, the framework's design should take into account the cognitive development stages of students of different ages, ensuring its broad applicability and flexibility for implementation in various educational contexts. By adopting this approach, a unified and effective set of computational thinking assessment standards can be established, promoting educational equity and offering a comparable benchmark for educational research.

6. REFERENCES

Duo, Z.J., Liu, Y.S., & Ren, Y.G. (2022). Research on Internal Mechanism and Teaching Practice of Programming Education for Development of Children's Computational Thinking. E-education Research, 43(8), 101–108.

Jin, J.S., Di, C.Y., & Zhou, Q.G. (2020). The Emphasis of Foundational Education in Information Technology Lies in Cultivating Computational Thinking. People's Education, Z3, 112–113.

Zhang, J.B. (2019). Computational Thinking Education: Concept Evolution and Challenges. Modern Distance Education Research, 31(6), 89–101.

Zhou, Y.C. (2018). Adding 'Higher-Order Thinking' to the Classroom: A Case Study of STEM Curriculum Implementation. People's Education, 12, 69–72.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking.

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.

An Initial Exploration of Artificial Intelligence Large Unit Curriculum onstruction in Junior High Schools under the Perspective of Computational Thinking - Taking Machine Learning" as an Example

YingZHOU[1*], DengzhuoFU[2]
[1]Jiujiang NewTown Primary school, China
[2]Beijing University of Aeronautics and Astronautics, China
[1]2456487813@qq.com, [2]1574261343@qq.com

ABSTRACT

Computational thinking is considered a fundamental skill essential for everyone in the 21st century. Current research almost focuses on cultivating and developing students' computational thinking through curricula, yet there is limited research on constructing curricula under the guidance of computational thinking. From another perspective, we believe that computational thinking can play a guiding role in the content design and implementation of AI curricula in primary and secondary schools. This paper builds upon the project-based learning model for junior high school programming curriculum that points to core literacy, and further extends it to a five-step design method for Thinking Progressively in junior high school AI curriculum large units, and use this model to design the progression of problem chain and task chain for the AI curriculum large unit. We then construct a junior high school AI curriculum large unit from a computational thinking perspective, offering a framework for other educators to reference when designing and teaching AI curriculum units.

KEYWORDS

computational thinking, junior high school, artificial intelligence, large unit curriculum

1. INTRODUCTION

According to Professor Jeannette M. Wing's definition, "Computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science." Therefore, computational thinking is often regarded internationally as the process of teaching students to analyze and solve problems.

Our literature review reveals that the vast majority of research on computational thinking is focused on fostering and developing students' computational thinking through the curriculum, and very little research has been done on building a curriculum guided by computational thinking. From another perspective, We contend that computational thinking can effectively guide in the content design and teaching implementation of AI curricula for primary and secondary schools, and developing a widely applicable junior high school AI curriculum that aligns with local conditions, thus fostering good computational thinking in students.

2. PROJECT-BASED LEARNING MODEL FOR JUNIOR HIGH SCHOOL PROGRAMMING CURRICULUM POINTING TO CORE LITERACY.

Through repeated research and practice, we believe that the elements of junior high school students' computational thinking are decomposition ability, abstraction ability, algorithmic ability and migration (pattern) ability. Under the guidance of the above ideas, we constructed a junior high school programming project learning model pointing to the core literacy, including four linkages and six steps , as shown in Figure 1.

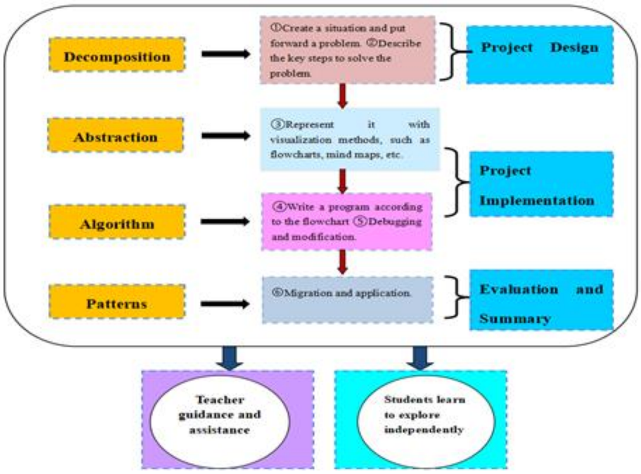


Figure 1. Project-based Learning Model for Junior High School Programming Curriculum Pointing to Core Literacy.

Through the implementation of the curriculum over the course of more than two years, students have notably significant improvement in the above four competencies, successfully cultivating and enhancing students' ability to analyze and solve problems under the guidance of computational thinking.

3. CONSTRUCTING AN ARTIFICIAL INTELLIGENCE LARGE UNIT CURRICULUM FOR JUNIOR HIGH SCHOOLS IN THE PERSPECTIVE OF COMPUTATIONAL THINKING - TAKING "MACHINE LEARNING" AS AN EXAMPLE

Building on the foundation of "Research on the Development and Practice of Junior High School Programming Curriculum Pointing to the Core Literacy of Information Technology Discipline", we investigated the current state of AI education in junior high schools in Wenjiang District from January to March, 2023. Based on the investigation and the accumulation of the results of the previous project, combined with the fact that the current IT teaching materials lack AI-related content, other developed AI courses require substantial hardware investment but there is limited funding for AI specialization in our district, etc. In order to achieve the objective of widespread AI education across regional junior high schools, we planned and designed a regional universal junior high school AI large unit curriculum. Taking "Machine Learning" as an example, the following is a detailed account of how to construct a learning large unit under the perspective of computational thinking.

3.1. Distill the Big Concepts of Large Unit

In this large unit, the big concept we distilled is "Machine Learning", that is, "how computers simulate or implement human learning behavior in order to acquire new knowledge or skills."

3.2. Create a Big Task (Big Problem) throughout the Large Unit

Our students are all honorable members of the Young Pioneers, and correctly wearing the red scarf is an obligation and responsibility of being a Young Pioneers. However, there are still many students who forget to wear the red scarf or do not wear it correctly. Relying solely on teachers and class cadres for inspection may not be very feasible. Could students use artificial intelligence technology to assist teachers in automatically checking whether students are wearing and correctly wearing the red scarf?

3.3. The Five-Step Design Method for Thinking Progressively in Junior High School Artificial Intelligence Large Unit

Based on the " Project-based Learning Model for Junior High School Programming Curriculum pointing to Core Literacy ", we have summarized the "Five-Step Design Method for Progressive Thinking in Junior High School Artificial Intelligence Large Unit " through continuous learning and reflection, comprising the following five stages, as shown in Figure 2.

The first step is decomposition, where the tasks of a large unit are broken down into smaller, more manageable parts; The second step is abstraction, where real-world problems are converted into problems that a computer can handle; The third step is algorithm development, involving writing a series of executable code to solve these problems; The fourth step is pattern, which involves evaluating, summarizing, and generalizing to identify common methods for solving this kind of problem; The fifth step is iteration, where the solution is summarized and improved to enhance its quality. Among these steps, the emphasis is on the internal logical connections between them, which facilitates the progressive and in-depth training of students' thinking processes.

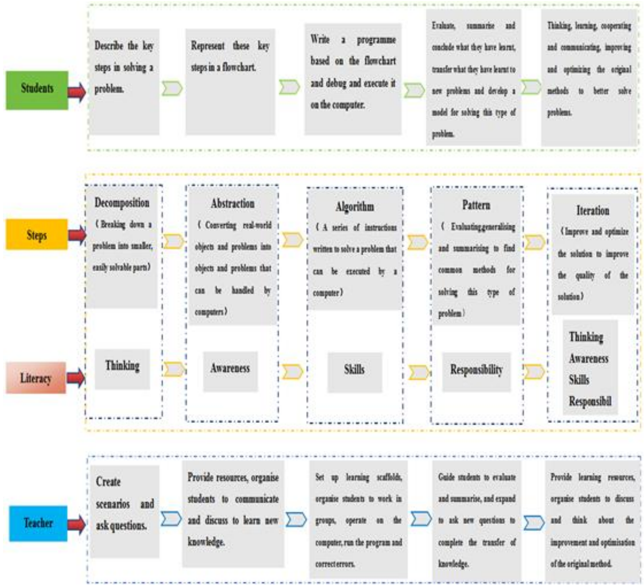


Figure 2. The Five-Step Design Method for Progressive Thinking in Junior High School Artificial Intelligence Large Unit

3.4. Constructing the Problem Chain and Task Chain of the Junior High School Artificial Intelligence Large Unit for Thinking Progression

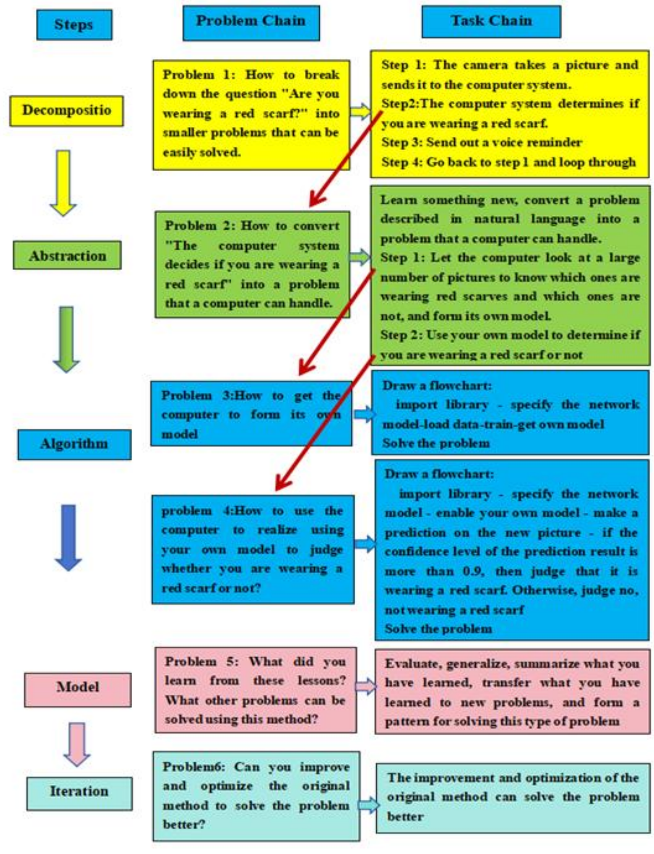


Figure 3. Problem Chain and Task Chain of "Machine Learning" Large Unit for Thinking Progression.

According to "The Five-Step Design Method for Progressive Thinking in Junior High School Artificial Intelligence Large Unit", we construct a problem chain and task chain for thinking Progressively in junior high school artificial intelligence units, as shown in Figure 3. The "problem chain" is a sequence of problems designed to address the tasks of the large unit. We transform knowledge into a series of clear and systematic teaching problems based on students' existing knowledge or experience, targeting the confusions that may arise during their learning process. The "problem chain for thinking progression" emphasizes that the problems are progressively deeper and more coherent. Subsequently,, through the " problems chain", the "tasks chain " is constructed, which, driven by tasks, fosters students' progressive thinking and achieves the cultivation and development of their computational thinking.

3.5. Based on the Problem Chain, Clarify the Logical Structure of the Unit's Knowledge and Determine the Content for Each Lesson within the "Machine Learning" Large Unit, as Shown in Figure 4

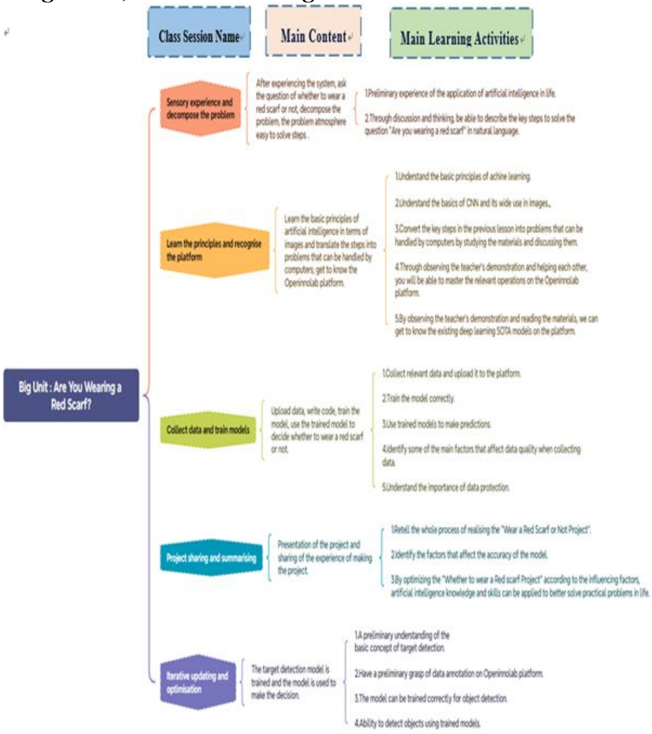


Figure 4. "Machine Learning" Large Unit Lesson Design.

4. CONCLUSION

Wenjiang District is located outside the main urban area of Chengdu City, and there are some urban schools with advanced equipment and many township schools with limited hardware. we have constructed a universal junior high school AI large unit curriculum under the guidance of computational thinking. After more than a year of implementation, our curriculum has proven to be effective and suitable for the regional context, achieving positive outcomes even in schools with limited hardware resources, and it also offers valuable insights and guidance for teachers in other regions to design and teach AI large units curriculum.

5. REFERENCES

Jeannette M. Wing. Computational Thinking[J]. Communications of ACM,2006,49(3):33-35

Research on the development status and countermeasures of computational thinking of junior high school students[1]

Luyao Zhao[1],Jihong Xu[2*]
Hohhot, Inner Mongolia Autonomous Region
Inner Mongolia Normal University
2536435482@qq.com,2848266323@qq.com

ABSTRACT

In the education stage of junior high school, cultivate students' ability to computational thinking, master the methods of computational thinking, in order to improve students' academic performance, through continuous learning and training, in order to stimulate junior high school students' sense of innovation. This paper uses the questionnaire survey method to investigate the development status of computational thinking of 43 students in the first grade of a junior high school, and the survey results show that there are certain problems in the development of computational thinking, and the research results of this paper have a certain role in promoting the cultivation of computational thinking.

KEYWORDS

junior high school students, computational thinking, questionnaires

1. INTRODUCTION

With the continuous development of information technology in China, junior high school information technology has been valued and recognized, and computational thinking has significant advantages in junior high school students' information technology [1]. As one of the four core literacies of information technology disciplines, computational thinking has an important discipline attribute identification. It is of great significance to grasp the current situation of junior high school students' computational thinking to welcome the implementation of the Information Technology Curriculum Standards for Compulsory Education (2022 Edition)[2].

Therefore, by investigating the development of computational thinking among junior high school students, this survey aims to find out the problems existing in the development of computational thinking among junior high school students, so as to put forward countermeasures and methods to solve the problems, so as to lay the foundation for subsequent training.

2. LITERATURE ACQUISITION

As shown in Figure 1, it is found that the preliminary research on this topic began in 2014, and the published literature in the following years showed an upward trend, indicating that this topic has gradually become a hot topic in recent years, which is worthy of exploration and further research.

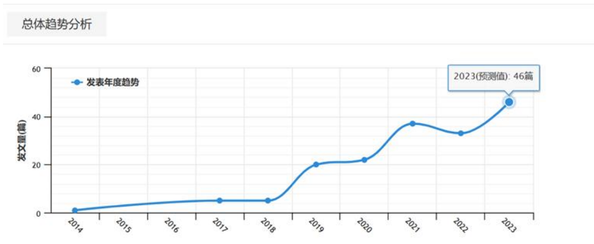


Figure 1. Literature trend chart

From the perspective of subject areas, as shown in Figure 2, they are divided into three aspects: secondary education, educational theory and educational management, and computer software and computer application.

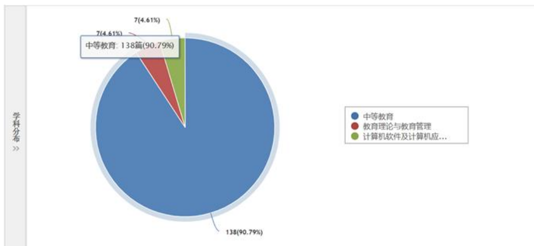


Figure 2. Subject area distribution

Through the search of the keyword "junior high school students, computational thinking", two research and analysis papers on the development of computational thinking of junior high school students were obtained, and it was found that these two articles belonged to the investigation and analysis of the development of computational thinking of junior high school students. These two articles were published in 2018 and 2020, respectively. Two years have passed since the publication of the two articles, but through reading, it is found that there is not much breakthrough in the content, and they are both investigated and analyzed from the two aspects of cognition and computational thinking ability.

Therefore, I focused on investigating and analyzing the development of computational thinking among junior high school students in more aspects, so as to draw suggestions for cultivating computational thinking in junior high school students.

3. RESEARCH METHODS AND PROCESSES

3.1 Research Methods

In this study, we used a questionnaire method, which is a structured questionnaire, which includes three types of

[1] Fund project: Research on Evaluation Tools of Computational Thinking for Core Literacy (S20231081Z)

questions: algorithm, logical reasoning, and system and society, aiming to investigate and analyze the development of computational thinking among junior high school students.

3.2 Research process

In this paper, a total of 44 questionnaires were surveyed, 40 questionnaires were actually collected, and 40 questionnaires were valid, with an effective rate of 91%. The results of these 40 valid questionnaires were counted and analyzed. (See the appendix for the specific content of the questionnaire)

4. THE DEVELOPMENT STATUS OF COMPUTATIONAL THINKING AMONG JUNIOR HIGH SCHOOL

4.1. A comparative description of the average scores of the three types of questions

According to the statistical results, the average scores of the three types of content questions were obtained, among which the average score of the algorithm was 1.625, the average score of the logical reasoning questions was 2.342, and the average score of the system and society questions was 2.283. Through the comparative analysis of the average scores of these three types of questions, it can be seen that the average scores of students in logical reasoning questions are the highest, which indicates that students have clear thinking and are good at using their brains. The average score on the algorithm is the lowest, indicating that the students are lacking in the actual practice process.

4.2. Statistical description of the class scores of junior high school students

According to the scores of the seventh class of the first grade of junior high school, statistics were made, as shown in Table 1 below:

Table 1. Statistical chart of scores

Score	**36points**
Class average	23.625 points
Top score in class	34 points
Lowest score in the class	10 points

It can be seen that the range difference is 34-10=21, which indicates that the development of students' computational thinking is quite different.

4.3. A comparative analysis and description of the number of people who answered correctly and the number of people who answered incorrectly in the three types of content questions

According to the statistical analysis of students' scores, the number of complete correct answers and the number of complete incorrect answers for three types of knowledge were sorted out, as shown in Table 2 below:

Table 2. Statistics on the scores of the three types of questions

	Number of people who answered correctly	**Number of people who answered incorrectly**
Algorithmic classes	0	2
Logic	19	1
Systems & Society	16	2

According to the above data analysis, the proportion of the number of people who answered correctly to the total number of students in the class was calculated, among which the number of people who answered the algorithm questions correctly was 0%, the number of people who answered the system and social questions correctly accounted for 46% of the total number of students, and the logical reasoning questions that answered the most in the class accounted for 54%. This shows that most students have strong logical reasoning ability, but lack a lot of algorithms, which makes the development of computational thinking unbalanced.

4.4. A description of the student's computational thinking development

4.4.1. A description of the development of students' computational thinking in terms of algorithms

According to the statistics of students' scores, the scores of each student's algorithm questions were sorted out. In view of this, the average scores of students in algorithm questions and the comparison of students' extreme scores were further counted. Among them, the full score of the algorithm question is 18 points, of which the average score of the students in the class is 9.75 points, the highest score of the class is 16 points, the lowest score of the class is 3 points, and the range difference is 16-3=13, which further shows that the development of students' computational thinking is very different in the algorithm.

4.4.2. A description of the student's computational thinking development in terms of logical reasoning

Table 3. Statistics of the scores of logical reasoning questions

Score	9**points**
Class average	7.025 points
Top score in class	9 points
Lowest score in the class	0 points

According to the data analysis, the full score of logical reasoning questions is 9 points, the average score of students is 7.025 points, the highest score of the class is 9 points, the lowest score of the class is 0 points, and the extreme difference is 9-0=0, which is a large difference.

4.4.3. A description of the systematic and social aspects of the development of students' computational thinking

According to the statistics of students' scores, the scores of each student's system and social questions are sorted out and counted as follows:

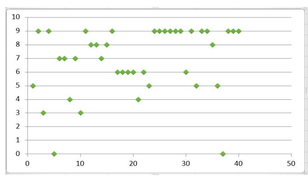


Figure 3. Scores of system and social questions

According to the above figure, most of the students' scores in the system and social questions are concentrated in 6 points or more, and the degree of dispersion is relatively high.To sum up, firstly, the dispersion of students in the three aspects of algorithm, logical reasoning and system society is too high, especially in the algorithm. Second, there is a big difference between the highest score in the class and the lowest score in the class; Finally, students in logical reasoning and systematic social questions scored highly, but many students scored very low. This indicates that there are large individual differences in the development of students' computational thinking.

4.5. A description of the differences in the development of students' computational thinking

4.5.1. A description of the differences in the development of each student's computational thinking

According to the scores of the first grade students, the radar analysis chart can be made as shown in Figure 4:

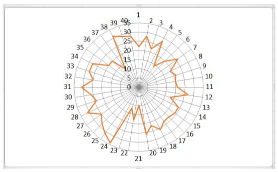


Figure 4. Radar chart of scores

The numbers 1-40 in the figure represent 40 students, and it can be seen from the graph analysis that the development of students' computational thinking is uneven.

4.5.2. A description of the differences in the development of computational thinking among students in three aspects

In this study, the overall computational thinking development of students is analyzed from three aspects: algorithm, logical reasoning, and system and society, and the average scores of students in the three aspects are calculated, which can be made as shown in Figure 5.

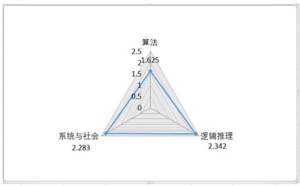


Figure 5. Radar analysis diagram

The development of students' computational thinking is better than that of algorithms in terms of systems and societies and logical reasoning, and the balance between the two is more balanced, and the lack of algorithms is greater.

5. EXISTING PROBLEM

5.1. The overall computational thinking development of junior high school students is lacking

According to the survey, 46% of the students in the class were below the class average; Especially in the field of algorithms, no students answered this type of question completely correctly, and the average score of the algorithm class was only 54% of the total score, which did not reach the passing level.

5.2. There are large individual differences in the development of computational thinking among junior high school students

This study analyzes the development of students' computational thinking from three aspects: algorithm, logical reasoning, and system and society, and finds that students' scores in these three aspects are highly discrete and unfocused. There is a large difference between the highest and lowest scores of students in the class; This indicates that there are large individual differences in the development of computational thinking among middle school students.

.

5.3. There is an imbalance in the development of computational thinking among junior high school students

The analysis showed that the average score of middle school students in algorithm questions was very low, while the average score of logical reasoning and system and society was relatively high. This shows that the students' ability to solve algorithm questions is very weak, and they are more inclined to logical reasoning questions and system and social questions.

6. STRATEGIES FOR CULTIVATING COMPUTATIONAL THINKING AMONG JUNIOR HIGH SCHOOL STUDENTS

Based on the above conclusions, this study concludes that the development of computational thinking among middle school students is insufficient. This requires teachers to focus on cultivating students' computational thinking skills in teaching.

6.1. Upgrade the teaching mode to cultivate students' computational thinking

In the teaching of information technology in junior high schools, students are trained to think computationally by adopting innovative teaching models. As teachers, we should explore new teaching models, avoid the phenomenon of going through the motions and being empty, and comprehensively use flipped classrooms, micro-classes and other methods to achieve comprehensive teaching effects and truly attract students to the classroom. In the process of learning information technology in junior high school, many students lack interest in learning and learning participation, and the learning is not targeted and effective, which needs to be further optimized and improved[3].

6.2. The cultivation of students' computational thinking should respect the individual differences of students

From the perspective of current education, teachers should focus on improving the effectiveness of computational thinking classrooms. At the same time, it actively guides and changes the bad learning habits of junior high school students in the teaching process, so that students can effectively participate in the actual teaching, and then improve students' attention to information technology classrooms [4] At the same time, innovative teaching methods are introduced to guide students to fully relax in the classroom, create a classroom teaching atmosphere suitable for students' learning, and improve the participation and experience of junior high school students in computational thinking education with the help of innovation.

7. REFERENCES

Zou Wei. Computational thinking of information technology for junior high school students[J].Computer Knowledge and Technology,2019,15(13):149-150.)

Zhou Chun. Current situation and cultivation strategy of computational thinking of junior high school students from the perspective of core literacy: Based on a survey of 8981 sixth-grade students in M District, Shanghai[J].Shanghai Educational Research,2022,(06):42-47.)

Wang Yunhui. Research on the design of junior high school programming gamification learning process for the cultivation of computational thinking[D].Northeast Normal University,2021.

Xiao Nan. Research on the creation of information technology problem situation in junior high school based on computational thinking training[D].Shandong Normal University,2021

亚太计算机教育应用学会

北京師範大學

BEIJING NORMAL UNIVERSITY