Structural Model and Practical Path: Thinking-Based Programming Teaching to Promote Computing Thinking*

Enwei Xu  
Xinjiang Normal University, Xinjiang, China  
Yuanyuan Wang  
Urumqi No. 23 Middle School, Xinjiang, China

Computational thinking is an important ability to solve problems in an intelligent society, and programming teaching is the best carrier for cultivating computational thinking. However, because teachers lack a critical understanding of the connotation of computational thinking, resulting in little effect on the cultivation of computational thinking. Therefore, based on the essential connotation of computational thinking and thinking teaching theory, this paper constructs the “CTAD—TPTM” thinking programming teaching structure model with both “computational characteristics and thinking attributes” of computational thinking and constructs the “A—IPO—D” teaching practice path of concrete implementation, and on this basis, design project-based teaching cases and guidance scaffold. It is expected to provide a new research perspective and practical reference for the design and implementation of thinking programming teaching, so as to promote the development of students’ computing thinking ability as a whole.

*Acknowledgement: This work was supported by the autonomous region’s graduate scientific research and innovation project “Research on In-Depth Learning of High School Information Technology Courses for the Cultivation of Computing Thinking” (Project No.: XJ2022G016), and the independent innovation fund project for doctoral students of the School of Educational Sciences of Xinjiang Normal University “Research on Project-based Teaching of High School Information Technology Courses From the Perspective of Discipline Core Literacy” (Project No.: XINUJKYA2003).  
Enwei Xu, doctoral candidate, School of Educational Sciences, Xinjiang Normal University.  
Yuanyuan Wang, a teacher of Urumqi No. 23 Middle School.

Keywords: computational thinking, thinking programming teaching, structural model, practice path

Introduction

As one of the important abilities to solve problems in intelligent society, computing thinking is the basic literacy that digital citizens should have in the 21st century. Programming teaching is the best carrier for cultivating computational thinking, and it is also the basis for implementing the core literacy training of information technology disciplines (Zhang & Ji, 2018). Some countries or regions in the world have implemented programming teaching into national policies to further cultivate students’ computational thinking. For example, the National Science Foundation (NSF) of the United States launched the “21 century Computing Education in K-12” (CE21) program in 2011 to improve the computing thinking ability of teachers and students in K-12 stage (Grover & Pea, 2017). In the “Citizens Digital Literacy Competence Framework 2.0” released by the European Union (EU) in 2016, EU member states are required to pay attention to the society’s requirements for citizens’ programming skills, and give specific instructions on the use of computational thinking to solve
The State Council of China issued the development plan for a new generation of artificial intelligence in 2017, it is proposed to “set up artificial intelligence related courses in primary and secondary schools and gradually promote programming education”, and in the reply to education proposal in 2020, it is said that “relevant special documents have been formulated to promote and standardize the development of programming education, and it is planned to incorporate programming education into the curriculum system of primary and secondary schools, focusing on the development of students’ core literacy such as computing thinking” (Ministry of Education of China, 2020). In China, classroom teaching is still the main practice field of cultivating computing thinking. How to effectively cultivate computing thinking in programming classroom teaching is an urgent problem to be solved in the curriculum reform based on core literacy.

The Connotation Interpretation and Practical Review of Computational Thinking

Scientific and accurate understanding of the connotation of computing thinking is the premise of clearly understanding the core literacy system of information technology discipline and effectively cultivating computing thinking. Computational thinking was first proposed by Simon Papert of the Massachusetts Institute of Technology in 1980 and described it as “thinking skills trained and cultivated by children in computer learning” (Papert, 1980). Subsequently, Wing redefined it from the perspective of Computer Science in 2006 and believed that “Computational thinking is a series of thinking activities covering computer science, such as problem-solving and system design by using the thinking mode and basic concepts of computer science, thinking, understanding, and solving problems like computer scientists” (Wing, 2006), however, its concept is relatively broad and difficult to operate. In 2011, the International Association for Educational Technology (ISTE) and the Association of Computer Science Teachers (CSTA) jointly proposed an operational definition of computational thinking training in K-12 education, that is, “Computational thinking is a problem-solving thinking process that includes “analyze problems with tools, process data logically, use algorithms to automatically solve problems, and transfer problems in the process of problem-solving” (ISTE & CSTA, 2011). Based on this operational definition, Brennan (2012) constructed a three-dimensional framework of computing thinking including “computing concept, computing practice, and computing sense” based on scratch children’s programming environment, which has become a widely accepted supporting theory and practical guidance for computing thinking teaching. On the basis of summarizing previous studies, Selby and Woollard (2013) extracted the five core elements of computing thinking: “abstraction, decomposition, algorithmic thinking, generalization, and evaluation”, which has not only become the five scientific computing methods for problem-solving, but also become an important evaluation index to test the training effect of computing thinking in teaching practice. China’s new curriculum standards point out that “Computational thinking refers to a series of thinking activities generated by individuals in the process of forming problem solutions by using the thinking methods in the field of computer science” (Ministry of Education of China, 2018).

Although the above definitions of computational thinking are quite different, its connotation focuses on two aspects: One is problem-solving, and the other is thinking activity, that is, computational thinking is not only an ability to solve problems by using computer tools, but also an expression of internal thinking activity in the process of problem-solving, with both the “computational characteristics” of problem-solving and the “thinking attribute” of thinking activity. However, because computational thinking originated in the computer field, teachers’ cognition and understanding only stay at the level of “computation”. In the practice of
programming teaching, they focus on its “calculation” characteristics and ignore the attributes of “thinking”, resulting in the difference between the value significance of computing thinking in the intelligent era and the training effect of teaching practice. It is highlighted in two aspects: Firstly, the content level of classroom teaching focuses on the learning of programming tools rather than the development of thinking ability, although programming education courses, such as graphical programming and Python text programming have been carried out one after another. However, in the actual teaching process, the freshness of programming technology is too pursued, causing students to focus on “try once” technical tools, and unable to achieve a deep understanding of the problem situation and the thinking ability to solve practical problems; the second is that the teacher presupposes the problem-solving path at the level of teaching mode rather than the exploratory meaning construction that points to the thinking. Although several teaching strategies, such as task-driven, problem-based, or project-based are used in teaching, teachers often preset learning paths and task operation steps in order to pursue problem-solving classroom efficiency, and require students to complete learning tasks in sequence according to the operation steps, resulting in programming teaching has become the non-thinking learning of imitating and repeating operations (Gu, 2018).

Facing the practical dilemma that the training effect of computational thinking is very little, how to give consideration to the “computational characteristics of problem-solving and the thinking attributes of thinking activities” of computational thinking in programming classroom teaching and effectively promote the development of students’ computational thinking ability is an urgent problem to be solved based on the curriculum reform of core literacy.

**Theoretical Framework**

**Thinking-Based Teaching Theory**

Thinking teaching focuses on cultivating students’ thinking ability. At present, there are two basic forms of practice: Thinking teaching of independent courses and thinking teaching of integrated courses (Zhao, 2013). The former is to set up a separate thinking course in the school, through specialized teachers to carry out classroom teaching activities to improve students’ thinking ability, its teaching goal is to let students “know how to think”; the latter is to cultivate thinking ability and subject teaching closely integrated, through the teacher’s curriculum design and activity implementation focusing on thinking ability, students can develop their thinking ability while acquiring subject knowledge, its teaching goal is to allow students to “migrate and innovate”. However, the learning of all knowledge involves thinking, and there is no teaching without thinking. Therefore, developing students’ thinking ability in subject teaching is considered to be an effective practical path to cultivate students’ thinking ability (McGuinness, Sheehy, Curry, & Eakin, 2006). Lin and Hu (2010) regarded thinking activities as the core of teacher-student activities in classroom teaching, and put forward four basic principles and four basic links of thinking teaching according to the “three edge model focusing on thinking structure”, That is, the principle of “cognitive conflict, independent construction, self-monitoring, and application transfer” and the corresponding links of “teaching introduction, teaching process, teaching reflection, and application transfer”:

1. The principle of “cognitive conflict” requires teachers to create dilemma situations or problem situations that conflict with students’ existing cognitive experience in the link of “teaching introduction” to stimulate students’ thinking;
2. The principle of “independent construction” includes two levels: individual cognitive construction and social interaction construction. Teachers are required to strengthen the cooperation and interaction between teachers and students in the “teaching process” to promote the development of students’ individual thinking;

3. The principle of “self-monitoring” emphasizes the promotion and improvement of students’ thinking structure by learning reflection. Teachers are required to guide students to summarize knowledge, methods, processes, and experiences in the link of “teaching reflection”, so as to promote students to form a good thinking and cognitive structure;

4. The principle of “application transfer” emphasizes the transfer training of thinking. The transfer of thinking occurs in the process of mastering and applying knowledge and skills. Teachers are required to strengthen students’ application of knowledge and transfer training in different situations in the link of “application transfer”.

Thinking-based teaching theory emphasizes that the core of classroom activities is the cultivation of students’ thinking ability to promote transfer innovation. Its “four basic principles” and “four basic links” provide theoretical guidance and operational guidelines for studying thinking programming teaching. However, it did not point out how to integrate computational thinking with programming teaching and to develop students’ thinking literacy through programming teaching.

The Thinking Literacy of Computational Thinking in the New Curriculum Standard of China

The “General High School Information Technology Curriculum Standards (2017 Edition)” pointed out:

“Computational thinking” refers to a series of thinking activities generated by individuals in the process of forming problem solutions by using the ideological methods in the field of computer science. Students with computational thinking can use the ideological methods in the field of computer science to define problems, abstract characteristics, establish structural models, reasonably organize data, and use various information resources through judgment, analysis, and synthesis reasonable algorithm to form a solution to the problem. Summarize the process and methods of using computers to solve problems, and migrate to other related problem-solving. (Ministry of Education of China, 2018)

“Defining problems, abstracting features, and establishing structural models” implies the cultivation of “problem thinking”. The intelligent society will face various “uncertainty” problems. Is it possible to analyze the problems contained in the situation, define them, and characterize them? To carry out abstract interpretation and establish a structural model of problem characteristics requires students to have the ability to think proactively.

“Reasonably organizing data and using reasonable algorithms to form a solution to problems” contains the cultivation of “critical thinking and algorithmic thinking”, in which “reasonable” is critical thinking, which requires students to dialectically raise questions and make critical choices on the basis of analysis and evaluation to supervise problem-solving to move in the right direction, “algorithm” is algorithmic thinking. It is a series of well-defined logical steps of tasks to be performed, and the sequence of steps is a concrete embodiment of algorithmic thinking.

“Judging, analyzing, and synthesizing various information resources” contains the cultivation of “collaborative thinking and critical thinking”. Learning is achieved in the process of interaction and cooperation between teachers and students. In cooperation, students not only need to critically question, select, transfer, and apply various information resources through critical thinking, but also cooperation itself contains emotional interaction. “Collaborative thinking” contributes to the improvement of community learning emotion.
“Transferring to other related problem-solving” contains the cultivation of “innovative thinking”. Transferring the learned knowledge and ability to solve problems creatively is the embodiment of innovative thinking, the ultimate direction of programming teaching, and the key ability to adapt to the problem-solving of intelligent society.

In summary, the definition of the concept of computational thinking in the “New Curriculum Standard” contains five concrete thinking qualities that point to the development of computational thinking capabilities, including “problem thinking, critical thinking, collaborative thinking, algorithmic thinking, and innovative thinking”.

“CTAD-TPTM” Structural Model: Constructing and Interpreting

Based on the essential connotation of computational thinking and thinking based teaching theory, this study constructs a thinking based programming teaching structure model with “computational characteristics and thinking attributes” pointing to the development of computational thinking ability (the English acronym is “CTAD-TPTM”) (see Figure 1).

Figure 1. Thinking programming teaching structure model pointing to the development of computing thinking ability (CTAD—TPTM).

The teaching structure model embodies three characteristics:

First, there are three dimensions of the model. The X-axis represents the knowledge content of computational thinking, which includes three elements: “computing concept, computing practice, and computing sense”:

1. “Computing concept” refers to the core concepts of programming teaching, such as sequence, variable, data, cycle, and other concepts;
2. “Computing practice” refers to the problem-solving process implemented by using computing concepts;
3. “The sense of computing” refers to the value tendency formed by learners in the process of computing practice.

The y-axis represents the scientific method of computational thinking, including five elements: problem decomposition, abstract representation, algorithm design, scheme evaluation, and generalization transfer. The z-axis represents the thinking quality focused by computational thinking, including five elements: problem thinking, critical thinking, collaborative thinking, algorithm thinking, and innovative thinking. The elements of each dimension are combined with each other to form 75 (3 * 5 * 5) structural units, and each structural unit is the development state of computational thinking ability pointed by thinking programming teaching.

Second, the computing features and thinking attributes of computational thinking are included in the model. “Computational features” refer to the five common computer science methods, such as “problem decomposition, abstract representation, algorithm design, program evaluation, and generalization migration” in the three dimensional framework of “computing concept, computing practice, and computing sense”. “Thinking attribute” refers to the innovative thinking activities focusing on “problem thinking, critical thinking, collaborative thinking, algorithmic thinking, and innovative thinking” in the process of problem-solving.

Third, the model takes the knowledge content of computational thinking as the carrier, and focuses on the accomplishment of thinking accomplishment through the method of computer science. It not only accords with the relationship among knowledge, method, and ability, but also reveals the development path of computational thinking ability in thinking programming teaching: On the one hand, computational thinking ability is an organic whole composed of its knowledge content, calculation method, and thinking goal, and any dimension cannot be ignored in teaching practice; on the other hand, in teaching practice, we need to take the thinking literacy focused by computational thinking as the goal guidance of teaching, guide students to constantly explore and improve the structural units of the model, so as to promote the good development of their computational thinking ability.

![Figure 2. Thinking programming teaching practice path pointing to the development of computational thinking ability (A-IPO-D).](image)

“A-IPO-D” Practice Path: The Embodied Implementation of the “CTAD-TPTM”

Although the “CTAD—TPTM” teaching structure model theoretically interprets the organic unity of the knowledge content, calculation method, and focused thinking literacy of computational thinking, and points out the training direction for the development of computational thinking ability in teaching practice, its abstract theoretical interpretation cannot provide an operable method guide for programming teaching practice. The most common and basic method in programming design is the IPO method (“input-processing-output”), but it is only a general programming method and does not point out how to cultivate computational thinking in
teaching. Therefore, based on the “CTAD—TPTM” teaching structure model and the IPO Method of programming design, this paper constructs a “A—IPO—D” (Analysis situation –Input-Processing-Output-Display Communication) teaching practice path for the specific implementation of “CTAD—TPTM” (see Figure 2).

The “A-IPO-D” practice path includes four teaching links, each of which has corresponding core elements of computational thinking, thinking teaching principles, and focused thinking literacy. The specific implementation points are as follows:

**Analyze the Problem Situation (A): Problem Definition and Abstraction**

Analyzing the problems in the situation is the beginning of programming, and the problem situation is the key to triggering students’ cognitive conflicts and activating thinking. In specific problem situations, students can define, decompose, and refine the problem through the interconnection of new and old knowledge and experience, and be able to abstract and generalize its characteristics. This is the key to problem-solving. This link stimulates students’ cognitive conflicts through the problem situation, and triggers students to think about the nature of the problem, that is, can students use their own language to explain the problem that needs to be solved in the situation? How to abstract the basic characteristics of the problem and the core concept knowledge involved in the subject from the context? Is it possible and how to break down the problem into smaller constituent units? Through this kind of thinking, students will be able to understand the concept of calculation, cultivate students’ problem thinking, and reach the element of computational thinking of “abstraction of problem features and problem decomposition”.

**Input (I): Data Organization and Model Construction**

Input is the beginning of a program, and its methods generally include: file input, network input, user keyboard input, database input, program internal parameter input, etc. The corresponding data types are numeric, list, set, tuple, dictionary, etc. That is, the input mainly revolves around the acquisition and organization of data to construct an organizational model of data. Data is the core concept of the information technology discipline, and its accurate understanding and organization is the key to cultivating computational thinking. In this session, students’ thinking about data is triggered by the abstracted characteristics of the problem, that is, can students clearly state what data is needed for problem-solving? What are the basic types of these data and how to obtain them? Can a data organization model be constructed in the form of a flowchart? How to integrate the knowledge achievements of the members of the collaboration group and think critically about the rationality of data selection and organizational model construction? Through this kind of thinking, students’ understanding of computing practice will be stimulated, and students’ critical thinking and collaborative thinking will be cultivated in collaborative learning, so as to reach the element of computational thinking of “evaluation and generalization”.

**Processing (P) and Output (O): Algorithm Design and Evaluation and Debugging**

The logic of program operation needs to process and output the input data, that is, algorithm design, evaluation, and debugging. The algorithm is the concrete implementation of the data organization model and the concrete realization of computational thinking. The output is the way to show the results of the algorithm design. However, the realization of any program function is not accomplished overnight, and it is necessary to continuously debug and evaluate the problems that arise during the output process. In this session, the organization model of data triggers students to think about algorithms and programming tools, that is, can students find or design algorithm logic to solve the problem? Can they use calculation concepts, such as
sequences, variables, loops, and conditional statements to list the basic steps for solving programming problems? How to choose a suitable programming tool to combine the listed programming steps into a sequence of actions recognized by the computer to form an executable program? Is it possible to debug errors based on the process and results of program execution and evaluate the rationality of the algorithm program to solve the problem? Through this kind of thinking, students will re-understand computing practice, cultivate students’ algorithmic thinking and critical thinking, and reach the element of computational thinking of “algorithmic thinking and evaluation”.

Exhibit and Exchange (D): Collaborative Reflection and Transfer Innovation

Exhibit and communication is a reflection process that allows students to express their understanding and perception of knowledge and methods related to situational problem-solving and programming realization. Two links of collaborative reflection and migration innovation can be designed according to teaching tasks and class hours. “Collaborative reflection” is to evaluate and summarize the problem solutions of situational tasks in the form of collaborative groups, so as to internalize students’ ability to transfer and innovate by using their learned knowledge and skills. “Migration innovation” is task-based problem-solving, including three difficulty levels: “basic task, ability task, and creation task”, so as to reproduce the internalized migration innovation ability. This link leads students to summarize and reflect on the process and methods of programming problem-solving through collaborative reflection and migration innovation, that is, can students explain the core concepts and skills involved in programming problem-solving? Can they reflect on the satisfaction of the programming solution in solving situational problems and how to improve it? Can they use your knowledge and ability to independently solve basic tasks, ability tasks, and creation tasks in similar situations? Through this thinking, students’ understanding of computing concept and re practice of computing practice are triggered, students’ migration and innovative thinking are cultivated, and there integration training of various elements of computing thinking is achieved.

Teaching Case Based on the Practice Path of “A—IPO—D”

Based on the “CTAD-TPTM” teaching structure model and its embodied implementation of the “A-IPO-D” teaching practice path, this research designed a thinking programming teaching case that points to the development of computational thinking ability, it is expected to provide practical reference for teachers’ implementation of embodied programming teaching. The case is selected from the content unit of “calculation and problem-solving” in data and calculation of education science edition. The teaching content is reconstructed on the basis of combing the core knowledge of the unit, selecting “guessing numbers game” as the project theme, integrating the unit knowledge content and carry out project-based teaching based on the teaching practice path of “A-IPO-D” (see Table 1), and designing a learning guidance scaffold to support students to carry out project-based learning (see Table 2).

The project case is presented in a problem situation to stimulate students’ interest in thinking. In the process of project learning, five common computer science methods, such as “problem decomposition, abstract representation, algorithm design, program evaluation, and general migration” are used to traverse the learning content of “computing concept, computing practice, and computing sense”, and focus on students’ thinking literacy of “problem thinking, critical thinking, collaborative thinking, algorithm thinking, and innovative thinking” in the process of problem-solving, and then develop students’ computational thinking ability, so as to develop students’ computational thinking ability.
Table 1
Learning Guidance Scaffold of “Guessing Numbers Game” Project

<table>
<thead>
<tr>
<th>Project theme</th>
<th>Project members</th>
<th>Problem situation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project link</td>
<td>Key points of problem-solving</td>
<td>Measures and results</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyzing problem situations (A): problem definition and decomposition</th>
<th>1. Can you use your own language to explain the problems to be solved in the situation?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. What are the basic characteristics of situational problems? What discipline core knowledge is involved?</td>
</tr>
<tr>
<td></td>
<td>3. Can and how to decompose the problem into smaller constituent units?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input (I): data organization and model construction</th>
<th>1. Can you clearly tell what data is needed in problem-solving?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. What are the basic types of these data and how to obtain them?</td>
</tr>
<tr>
<td></td>
<td>3. Can the data organization model be constructed in the form of flow chart?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing (P): algorithm design</th>
<th>1. Can we find or design the algorithm logic to solve the problem?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. Can you use the calculation concepts, such as sequence, variable, loop, and conditional statement to list the basic operation steps for solving programming problems? Please write the detailed code.</td>
</tr>
<tr>
<td></td>
<td>3. How to select appropriate programming tools to combine the listed programming steps into a behavior sequence recognized by the computer to form an executable program?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output: evaluate debug</th>
<th>1. Where is the processing results output? What is the output data type?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. What errors occurred during operation? How is it corrected?</td>
</tr>
<tr>
<td></td>
<td>3. Whether it can debug errors according to the process and results of program execution and evaluate the rationality of the algorithm program to solve problems.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exhibition and communication (D): collaborative reflection and migration innovation</th>
<th>1. Can you explain the core concepts, skills, and methods involved in programming problem solutions?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. How satisfied is the programming solution with solving situational problems and how to improve it?</td>
</tr>
<tr>
<td></td>
<td>3. Whether they can independently solve the basic tasks, ability tasks, and creation tasks in similar situations by using their mastered knowledge and ability.</td>
</tr>
</tbody>
</table>

Table 2
Teaching Case of “Guessing Numbers Game” Project (A-IPO-D)

<table>
<thead>
<tr>
<th>Project theme</th>
<th>Guess the number game</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem situation</td>
<td>Guessing the numbers is an ancient password-decoding psychological game. It is generally played by two or more people, or it can be played by one person and a computer. The rule of the game is: The person who produces the number thinks a number from 1 to 100, and the person who guesses it must not know it. Every time you guess a number, the person who tells the number must give back the result to judge whether you win or lose. Please try to write a small number guessing game program that can realize human-computer guessing.</td>
</tr>
<tr>
<td>Analyzing problem situations (A): problem definition and decomposition</td>
<td>Lead students to define the specific problem to be solved from the above problem situation: How to use programming tools to write a small number guessing game program that can realize human-computer guessing, which involves calculation concepts such as random numbers, data, judgments, and loops. Then lead the students to decompose it into three sub-problems: (a) Obtaining data: The computer randomly generates an integer number between 1 and 100, and the player inputs a guessed number, and abstractly expresses it as “random-data and guess-data”; (b) Data judgment: The computer automatically prompts “guessed high, guessed low, or guessed right” according to the actual situation; (c) Repeat loop: Repeat guessing until the game is over.</td>
</tr>
<tr>
<td>Input (I): data organization and model construction</td>
<td>Select, analyze, and organize the obtained data through group cooperation according to the sub-problem to build the data organization model, as shown in the following figure:</td>
</tr>
</tbody>
</table>
STRUCTURAL MODEL AND PRACTICAL PATH

Processing and output (PO): algorithm design, evaluation, and debugging

- Lead students to first select programming tools (graphical or Python), then define algorithm structures, such as “data variables, logical comparison of data, and circular structure”, and finally combine them to form programs (graphical and text), and test the results to evaluate whether the situation problem has been solved.

Exhibition and communication (D): collaborative reflection and migration innovation

- Evaluate and summarize the solution of program realization problems in the form of a collaborative group, and constantly ask “whether the realized functions are perfect? Whether other functions (times and time) need to be added?” to train students’ generalization ability and critical thinking. Or then solve the problem based on three types of tasks, and design three types of tasks at different levels to promote students’ personalized development. The tasks include:
  (a) Basic task: price guessing game;
  (b) Ability task: stone scissors paper game;
  (c) Creative task: choose to solve common problems in life.

Summary

Due to the lack of critical understanding of the essential connotation of the concept of computing thinking, teachers pay more attention to its “computing” characteristics and ignore the “thinking” attribute in programming teaching practice, resulting in the difference between the value significance of computing thinking in intelligent society and the training effect of teaching practice. Facing the practical dilemma that the effect of programming teaching on cultivating computational thinking is very little, according to the essential connotation of computational thinking and thinking teaching theory, this paper constructs a “CTAD-TPTM” thinking programming teaching structure model with “the computational characteristics of problem-solving and the thinking attributes of thinking activities” of computational thinking as well as its embodied implementation teaching practice path of “A-IPO-D” and designs a specific project-based teaching cases and guidance scaffold based on this, which can not only provide a new research perspective for academic researchers and teaching practitioners to study the development of computing thinking ability, but also provide practical reference for the design and implementation of thinking programming teaching, so as to promote the development of students’
computing thinking ability as a whole and promote them to become the solver and creator of intelligent social problems.

References


Zhao, G. Q. (2013). A centennial review of thinking teaching research. Modern Distance Education Research, 6, 45.