Instructional Design of a University Computer Fundamentals Course Oriented Toward the Cultivation of Computational Thinking

Hong Lei*

China Civil Affairs University, Beijing, 102600, China
**Corresponding author: leihong8888@163.com*

Abstract: *Functional transformation of computing thinking as the core of teaching reform of the University Computer Fundamentals Course is from skill training based on tools to cultivating thinking abilities. Based on the conceptual evolution and core elements of computational thinking, analyzing the cognitive position and dual teaching functions of the University Computer Fundamentals Course, revealing the intrinsic correspondence among different knowledge modules with computational thinking elements. Propose a content organization framework by thinking-oriented reorganization of knowledge modules and designing progressive training levels; Establishing a synergistic mapping mechanism for teaching content and thinking cultivation goals. In building up teaching strategies, Constructing a problem-guided teaching process, Providing ways of thinking visualization and cognitive externalization, Setting diverse dimensions of thinking evaluation. Our research provides an instructional design framework that enables the University Computer Fundamentals Course to transform systematically from transmitting knowledge to cultivating thinking.*

Keywords: *computational thinking; University Computer Fundamentals Course; instructional design; organization of teaching content; thinking evaluation*

Introduction

From tool skills training to cultivating computational thinking University Computer Fundamentals Course Computational thinking is one of the fundamental literacy for modern learners, therefore, our course should cultivate thinking ability whilst imparting computer knowledge. There are two disconnects with current teaching practices; firstly, there's no intrinsic connection between organizing knowledge content and the need of thinking training; secondly, teaching activities took knowledge transmission as the main line without guaranteeing students' deep participation in thinking. To solve the above problems, we focus on building up an instructional design framework which centers at cultivating computational thinking including two levels: the organization of teaching content and the design of teaching strategies so as to meet the practical needs of course transformation and driving its transition from a knowledge transmition model to a thinking culturing model.

1. Analysis of the Connotations of Computational Thinking and the Teaching Positioning of the Course

1.1 The Conceptual Evolution and Core Elements of Computational Thinking

The proposal and development of the concept of computational thinking emerged from the constant exploration for the essence of the field of computer science education. After going along with the formation trajectory of this concept, early definitions of computational thinking are obviously oriented towards instrumentality; it is mostly seen as fostering logical reasoning and algorithmic construction ability by programming instruction, concentrating on learning programming syntax and debugging and running codes. With computer science penetrating every field of society extensively, the academic community has realized that restricting the scope of computational thinking into programming skills is a serious underestimation of its value. To meet such change in understanding, the connotation of computational thinking was reexamined and continuously expanded until finally defining it as a general thinking activity which uses the basic concepts and methods of computer science to solve problems, design systems and understand human behavior. This shift in the boundary of the concept reflects a leap

in the academic community's understanding about the essential function of computer education: what we convey through computer courses not just the skill of operating machines, but also a way of thinking to understand and transform the world[1].

In terms of cognitive operations, the main elements in computational thinking include some basic elements such as: abstraction, decomposition, algorithmic construction, systematic thinking etc., focusing respectively on extracting important attributes for certain context and construct computable model; decompose complex problems into independent sub-problems; describe clearly and organize orderly solution steps (algorithmic thinking); collaborating together when solving real problems, forming a whole cognitive chain from identifying problems to implementing solutions.

1.2 Teaching Objectives and Cognitive Positioning of the University Computer Fundamentals Course

The University Computer Fundamentals Course has an unique place within the general education system of higher education; Our teaching goals have changed from operating tools early on to cultivate computational thinking ability. Focusing purely on using office software and basic information processing will not be up-to-date for changing information technology environment but should take on the role of cultivating thinking besides transmitting knowledge. To achieve this goal, we as course designers need to re-examine our rationale for the course's existence and raise our teaching objectives from being able to use (technical) to being good at thinking (cognitive).

Based on the cognitive positioning we see from our analysis of this course, there are two functions of the University Computer Fundamentals Course as a teacher: The surface function is helping students build up a complete cognitive framework of the computer system (hardware composition structure and working principles of computers; hierarchical relationship between system software and application software; representation and storage methods of data inside computers; basic principles and protocol system of network communication), providing the conceptual foundation for understanding the operating mechanisms of computers. And the deep function is guiding at the level of thinking—helping students gradually acquire the methodology characteristic unique to computer science by acquiring the aforementioned knowledge—the choice of an appropriate level of abstraction when dealing with complex problems, transform real world needs into computable form, reduce system complexity via layered design, make efficiency-based trade-offs among multiple possible solutions. They mutually support and symbiosis rather than being sequentially related-one another's acquisition of knowledge provide the materials and carrier for the unfolding of thinking, and the deepening of thinking endow the knowledge with the vitality for transfer and application[2].

1.3 The Intrinsic Connection Between Computational Thinking and the Teaching Content of the Course

For the teaching content of the University Computer Fundamentals Course covering knowledge modules including computer system cognition, data representation and processing, algorithms and program logic, and network communication, it is evident that they have a clear cognitive correspondence between each other and their respective core elements of computational thinking: The stored-program principle and the hierarchy of hardware has the training value of systematic thinking in the system cognition module; We can find direct training materials (encoding and data structures) for abstract thinking in the data representation module. Algorithms and program logic module has the closest relationship with the elements of computational thinking, where process control corresponds to the thinking mode of algorithmic construction and modular design corresponds to the thinking mode of decomposition and combination.

The concept of protocol layering in the network communication module reflects a way of decomposition for complicated communication tasks by different functional layers which handle them separately respectively; Error detection and fault-tolerant mechanisms reflect identifying and responding to uncertain factors. All these are specific knowledge content but at its essence it embodies the principle of computational thinking in designing distributed systems. Our main work on instructional design is to reveal and highlight the implicit thinking training resources hidden inside such knowledge contents so that when students learn specific concepts they will go through the process of computing thinking cognition, thus achieve an organic union between knowledge transmission and thinking cultivation.

2. Teaching Content Organization Framework Oriented Toward Computational Thinking

2.1 Strategies for Thinking-Oriented Reorganization of Knowledge Modules

Mostly, the content arrangement in traditional University Computer Fundamentals Courses conforms with the logic of the knowledge system of the computer discipline; arrange them section by section including hardware, software, networks, and applications. It's a convenient way to present the whole knowledge system while hiding to some extent the thinking training value hidden behind the knowledge. To organize teaching content towards cultivating computational thinking requires restructuring based on the existing knowledge modules using thinking as the main line. The key idea of reorganization is taking the core cognitive operations of computational thinking (abstraction, decomposition, algorithm design, and systematization), as the main line passing through all knowledge modules, breaking down the arrangement inertia of parallel listing by disciplinary branches[3].

Specifically, we conduct a thinking-oriented reorganization from two aspects: First, reconnecting knowledge points according to the natural process of problem solving (e.g. integration of abstract method, data organization, process control, optimization evaluation, etc., that were scattered in different chapters), forming several problem-centered module units; Second, annotate the thinking elements contained in every knowledge module and clarify what kind of computing thinking training is focused for each teaching unit to carry, change thinking cultivation from implicit infiltration to explicit design. The reorganized knowledge modules are no longer merely division of disciplinary knowledge anymore, but functional units with computational thinking development orientation. Each one has its own specific thinking training as the core goal, accompanied by corresponding knowledge content.

2.2 Design of Progressive Levels for Thinking Training

Computational Thinking as a high-level cognitive ability conforms to the general rules of cognitive psychology on forming and developing, it can't achieve overnight through intensive short-term training, organizing teaching contents must fully conform to the characteristics of thinking development at each stage building up a training sequence that is logically related inside. The inherent upper limit of learners' working memory according to Cognitive Load Theory, if the total amount of cognitive processing needed for a teaching task exceeds what learners can afford then the efficiency of learning will drop significantly, even leading to cognitive stagnation. When cultivating computational thinking initially, students haven't formed a basic thinking framework and operational schema; when facing highly complex thinking tasks, they encounter difficulty in establishing effective processing pathways because of information overload. Reasonable level design starts from task forms with lower cognitive load so that students gradually establish the basic conceptual framework and operational schema of computational thinking under low risk situation, and only once they have stabilized internalized these foundational schemas, we increase the complexity of tasks and deepen thinking[4].

Based on the cognitive gradient of the core elements of computational thinking, we classified our training levels as follows: the basic cognition stage, the method mastery stage, and the comprehensive application stage: In the basic cognition stage, it stresses establishing the basic concepts of computational thinking; here, students will have a basic operation mode of abstraction and decomposition with concrete and intuitive tasks. The method mastery stage focuses on the orderly training of algorithmic design and systematic thinking methods, which requires that students can solve the process from identifying the problem to designing solutions relatively independently. In the comprehensive application stage, students are confronted by ill-structured and complex problem situations, where they need to independently choose and combine different computational thinking strategies to solve problems. There exists not just an increase in difficulty levels between these three stages, but there is also a qualitative change in thinking operations from partial to holistic and from single to composite.

2.3 Synergistic Mapping Between Teaching Content and Thinking Cultivation Objectives

The extent to which we actually cultivate computational thinking versus just carrying around slogans is determined by whether our teaching content and thinking cultivation objectives have been properly mapped. Synergistic mapping means establishing an appropriate structure correspondence relationship between knowledge content and thinking elements such that each kind of teaching content has their attribution for thinking training; each thinking cultivation objective has a knowledge carrier upon which they can rely. Such correspondence should not be artificially attached but instead be built

according to the intrinsic compatibility between the epistemological characteristics of the knowledge content itself with the thinking elements.

The establishment of synergistic mapping relationship should accomplish two basic tasks: First, classify and code the knowledge content covered by the course from the perspective of thinking dimension (i.e., identifying the core kinds of thinking operation involved in the actual cognitive process of every knowledge unit) such as content like binary encoding and data compression, which has natural relation with abstract thinking and optimization thinking; sorting algorithms and search algorithms, which has close relation with algorithmic construction and efficiency evaluation. Secondly, decompose the thinking objectives at different levels expected to be achieved by the course into operable specific indicators, and match its position according to a three-dimensional comparison table of objectives, content, and thinking, thus presenting clearly the thinking cultivation function carried by each teaching unit and the distribution density of each thinking objective at different teaching stages. Synergistic mapping relationship provides a clear logical start point for further design of teaching strategy, making the cultivation of computational thinking having a traceable and evaluable carrier support when implementing teaching[5].

3. Instructional Design Strategies Based on Thinking Development

3.1 Construction of a Problem-Guided Teaching Process

How our arranging of teaching process affects the level of students' cognition participation and the quality of their thinking activity in the classroom. Most of the time we arrange the logical sequence of knowledge points mostly as the main line of teaching for traditional teaching processes. The speed of going in the class relies upon how fast the teacher explains; students receive passively. This type of process arrangement helps us quickly disseminate knowledge but it is hard to guarantee that students' thinking can keep up synchronously. In order to cultivate computational thinking with instructional design, we should change the construction logic of the teaching process from transmitting knowledge into using problem solving based on problems as driving force to advance the teaching process so that students will go through a whole thinking process guided by problems.

Teaching process guided by problems has three links: problem design, problem presentation, and problem advancement. Problems are designed according to two principles: computability and moderate openness. Computability means choose problems which can be solved step-by-step (which will carry out the training effect for algorithmic construction and systematic thinking), moderate openness means there is multiple entry angles/solution paths to avoid being routine training due to only one path. When presenting problems, it cannot be presented in a conclusion-first way but we should start with describing situations so that students can find the key information in the problems themselves and define the solution objectives. Staging sub-tasks is included in advancing problems. Every sub-task's completion apply some elements of computational thinking; cover all the cognitive chain from identifying problems/abstract modeling to designing solutions/result verification.

3.2 Design of Thinking Visualization and Cognitive Externalization

The operation process of computational thinking (as an implicit cognitive activity) is hard for outsiders to understand clearly, and learners have no awareness about their thinking processes. Due to its implicit nature, there are two difficulties when teaching: Firstly, teachers find it hard to know if the thinking activities of students unfold well or not without any real-time information base for teaching adjustment; Secondly, students find it hard to timely detect deviation on their thinking path which hinders them develop their metacognitive monitoring ability. In order to solve these problems, we need to deliberately apply a strategy of thinking visualization and cognitive externalization[6] in our instructional design.

The main idea of Thinking visualization is to make visible the thinking operations previously hidden in our minds. This includes but are not limited to: Ask students to write out the step-by-step process when solving problems with flowcharts or pseudocode; Graphically show how to establish an abstract model; Display the hierarchy of problem decomposition through mind maps. Besides cognitive externalization also needs visibility both on the result of thinking as well as the decision making base and judgment criteria during the thinking process. The operational means include setting up necessary explanation sessions, ask students to verbal or written explain the reason why they choose this solution while excluding other alternative paths after finishing a certain sub-task. By designing visualization and

externalization, we convert student's thinking processes from private experience to public texts which can be used as actionable objects for classroom discussions, peer assessments and teacher guidance.

3.3 Setting of Diversified Thinking Evaluation Dimensions

Evaluation methods have a great washback effect on teaching: if the evaluation system emphasizes the ability to memorize and reproduce knowledge points no matter how many times we emphasize the goal of cultivating thinking in our teaching, then it will inevitably lean towards exam orientation; establishing a system matching evaluation dimensions with the objective of computational thinking cultivation is an important guarantee to ensure that the philosophy of instructional design can be carried through. A single knowledge assessment or a single skill test cannot reflect the level of thinking development of students, so there should be diversification for setting up evaluation dimension according to multiple attributes of computational thinking.

The horizontal setup of thinking evaluation: We evaluate each single core element of computational thinking independently like abstract modeling ability (whether students can extract effective models from specific context), decomposition ability (whether students can find out the structural nature of problems and reasonably divide modules), algorithmic construction ability (whether students can design solution steps with determinacy and efficiency) and systematic ability (whether students can comprehensively consider the interactive relationship among modules). In the vertical direction it means the difference between different thinking qualities at a hierarchy; i.e., distinguish the levels of performance on the same thinking element: the imitation level, the transfer level and the creation level. The former indicates the completeness of students' thinking structure while latter indicates the depth of thinking development. By combining them, there will be an evaluation matrix that covers various modes of computational thinking operations and multiple cognitive levels. Each evaluation unit has their own performance indicators and observable behavior evidences which provide an operational basis for thinking evaluation and give more comprehensive diagnostic information for teaching decisions.

Conclusion

In order to cultivate computational thinking, we proposed a systematic instruction design framework for the University Computer Fundamentals Course at three levels: connotation analysis, content organization, and strategy design. In regard to connotation analysis, in order to make clear the composition of the core elements of computational thinking and the dual teaching functions of this course, we find that there is an intrinsic connection between teaching content and thinking cultivation; in regard to content organization, we put forward a thinking-oriented reorganization strategy for knowledge modules, which integrates course content organized with disciplinary logic into functional teaching units oriented towards thinking development, designs a training sequence consisting of three progressively increasing levels (basic cognition, method mastery, and comprehensive application), establishes a structural correspondence between teaching content and thinking objectives via a synergistic mapping mechanism; in regard to strategy design, we build up a teaching process centered upon problem solving, introduce thinking visualization and cognitive externalization methods to solve the dilemma brought about by the implicitness of computational thinking, set diversifying evaluation dimensions including multiple thinking operations and multiple cognitive levels.

These three levels constitute a complete design chain from theory cognition, passing through content architecture, ending up with implementation strategies. These three levels are logical progression among each other and give us a systematic solution for the instructional design of the University Computer Fundamentals Course. Future research we can pursue these directions such as: adaptive adjustment strategies for our design framework in various teaching contexts; Developing teaching resources and constructing platforms supporting thinking cultivation; Designing and verifying the reliability and validity of quantitative tools for thinking evaluation dimensions; Analyze differential needs of student groups with different disciplinary backgrounds on their computational thinking training paths.

References

[1] Fang Xingxing, Wang Huan, and Yu Jing. "Exploration of the Instructional Design of the Introduction to the University Computer Course Oriented Toward Computational Thinking." Computer

Knowledge and Technology 20.5(2024):135-137.
[2] Nie Ru, and Li Zhengwei. "Research on the Reform and Practice of University Computer Fundamentals Teaching Aimed at Cultivating Computational Thinking." Computer Knowledge and Technology 20.3(2024):142-144.
[3] Ma Pengwei. "Design of Teaching Thinking in Computer Fundamentals Courses." Electronic Technology 52.9(2023):236-237.
[4] Zhou Gang, et al. "Teaching of the University Computer Fundamentals Course to Promote the Implementation of Computational Thinking Cultivation Through Python." Computer Education .2(2022):79-82.
[5] Ning Kuangfeng, and Peng Hao. "Instructional Design of Computational Thinking Based on the University Computer Fundamentals Course." Industrial and Science Tribune 21.14(2022):184-185.
[6] Ju Yarong, et al. "Instructional Design of Computer Systems Oriented Toward the Cultivation of Computational Thinking." Computer Education .4(2021):93-97.