International Journal of Engineering and Applied Sciences (IJEAS)
ISSN: 2394-3661, Volume 7, Issue 10, October 2020

The Application of Progressive Case Teaching Method in the Course of Software Modeling and Practice

Pengtao Jia，Zi Wen

Abstract— **Aiming at the current teaching situation of software modeling and practice course, combined with the learning objectives of the course, the progressive case teaching method is proposed and implemented. According to the characteristics of the gradual deepening of the course content,the progressive case teaching method selects the cases that are easy to understand and interested by students, and adopts the hierarchical structure to design the cases and decompose the tasks. The latter sub-task of the case solves the problems existing in the former sub-task, and evolves or reconstructs the former sub-task. In the process of guiding students to solve problems step by step, the knowledge points are gradually introduced, so that students can learn from the shallower to the deeper, and establish computational thinking. After the teaching method was implemented in our school, good teaching results have been achieved and the software design ability of students has been generally improved.**

Index Terms—**Progressive case teaching; Software modeling; Design pattern**

The case teaching method is a case-based teaching method, which aims to guide learners to think, analyze, discuss and communicate according to the content of the case, so as to improve students' ability to analyze and solve problems. "Software Modeling and Practice" is an elective course in the training program of computer science talents in our university, which aims to improve students' software design ability. The content of the course is complex and various, the concepts are abstract and difficult to understand, and the engineering practice is strong. Many students are lack of interest in this course, and even feel tired of learning. Therefore, according to the needs of teaching objectives and course contents, it is of great significance to introduce case-based teaching, introduce students into specific situations of cases, guide students to understand and master theoretical knowledge, and cultivate students' practical ability, which is very important to stimulate and cultivate students' innovative thinking and achieve professional talent training objectives.

Pengtao Jia received the Ph.D. degree from the Department of Computer Science, Northwestern Polytechnical University, Xian, China, in 2008.She is currently a Professor with the Department of Computer Science, Xi’an University of Science and Technology, China. Her current research interests include Machine learning, data mining,visualization of coal mine safety.

Zi Wen is currently pursuing the master’s degree with the Department of Computer Science, Xi’an University of Science and Technology, China.

I. EXISTING PROBLEMS IN TEACHING

With the increasing social demand, the course of "Software Modeling and Practice" is becoming more and more popular in various colleges and universities. In our school, the training program for computer professionals also takes it as a degree elective course,aiming to improve students' software design ability. After investigation, it is found that the teaching of "Software Modeling and Practice" has yet to be improved.The main problems are as follows:

First, the content of teaching is relatively one-sided, with more theories and less practice. Most of the existing course content is to refine the content of software modeling chapters in the software engineering course. The content of theoretical guidance is more than the content of practical application, which can not effectively cultivate the practical application skills of software design talents. Second, case teaching does not really dominate the classroom. Compared with foreign related majors, the proportion of case teaching in the course teaching is small, and the number of lessons is still at the level of theoretical teaching, which is far from meeting the needs of computer professional personnel training.Third,the case library has fewer resources and the quality is worrying.At present,there are few case libraries suitable for the course of "Software Modeling and Practice".

Most of the case libraries that have been built are those for software engineering and software testing courses. Although the contents of software modeling are involved, the emphasis of case contents is different from the course name and training objectives, and there is a lack of practical content related to software modeling. Fourth, the existing cases cover few and scattered knowledge points. At the moment, the commonly used cases are only scattered knowledge points design cases. After learning, students can not integrate all the knowledge points, and can not form engineering ideas. Students have no way to start facing enterprise level development.

Therefore, in order to solve the above problems, the authors put forward a progressive case teaching method for the course of "Software Modeling and Practice", gradually and level by level to cultivate students' software modeling and comprehensive design capabilities.

II. DESIGN OF PROGRESSIVE CASE

A. Research on Progressive Case Teaching Methods

The case teaching method was first proposed in Harvard Law School in 1870 and was quickly promoted. It has been widely used in teaching. The progressive case teaching model refers to the fact that teachers design a spiral problem chain based on the characteristics of the course knowledge points, and build a problem step for students. In the process of guiding students to solve problems level by level, the knowledge points are introduced progressively so that students can learn from the shallower to the deeper, which helps to reduce the difficulty of learning and improve the learning effect.

The content of the "Software Modeling and Practice" course includes process-oriented design methods, object-oriented design methods, object-oriented design principles, software modeling, software design patterns, etc., involving the concept of program design structure, functions, classes and objects, inheritance, polymorphism, 7 object-oriented design principles, 5 types of creation design patterns, 7 types of structural design patterns, 11 types of behavioral design patterns and other knowledge points. These knowledge points have the characteristics of spiral and gradual evolution, which is very suitable for the progressive case teaching mode.

The design idea of the progressive teaching case of the "Software Modeling and Practice" course is: select the case that students are interested in and suitable for the design of progressive learning, and decompose it into multiple sub-tasks from the shallower to the deeper. The latter sub-task of the case solves the problems existing in the former sub-task, and evolves or reconstructs the former sub-task.Each sub-task covers the corresponding knowledge points, and as the learning deepens, until all the knowledge points are covered. Since one case cannot exhaust all knowledge points, a case library can be built, and multiple cases cross cover all knowledge points.

B. Design of the Case

This paper takes the calculator case in the case library as an example to show the design process of the progressive course case. The design ideas of this case are: the design method is changed from process-oriented method to object-oriented method; the course contents cover program structure, object-oriented core characteristics, software design mode, simple factory mode, factory mode, etc. The software platform is converted from the console to the visualization platform. The calculator case has conventional calculation functions such as four arithmetic operations, derivative calculations, trigonometric functions, squares, and square roots. The list of sub-tasks is shown in Table 1.

Task 1 implements the function of a single four arithmetic calculator with the minimum requirements, and the knowledge points involved are the sequence structure and branch structure in the program structure.Task 2 implements a four arithmetic calculator function that can be calculated multiple times,and the knowledge points involved are the loop structure in the program structure. Task 3 introduces object-oriented thinking to enable students to understand the concepts of abstraction and encapsulation. Task 4 uses inheritance to reconstruct task 3 to enable students to understand what software is good for reusability. Task 5 uses polymorphism and simple factory patterns to reconstruct task 4, so that students can have a preliminary understanding of software design patterns and understand what is easy to extend software. Task 6 analyzes the advantages and disadvantages of the simple factory model, and uses the factory model to reconstruct task 5 for its shortcomings that do not conform to the principles of open and closed software design. Task 7 guides students to modify the system interface from the console mode to the visual operation mode, and introduce the singleton mode, so that students understand the conversion of input and output modes under different platforms and the application mode of singleton mode.

Table 1 Sub-task division of scientific calculator case

Sub-task name	**Course content**	**Training objectives**
Task 1: Realize a simple four arithmetic calculator with a single operation.	sequence and branch structure in program structure	Let students implement a four-arithmetic calculator that can be executed once in a process-oriented way, so as to master the sequence and branch structure.
Task 2: Realize a simple four arithmetic calculator with multiple operations.	loop structure, functions, software development specifications in program structure	Let students realize the four arithmetic calculators which can be executed many times in a process oriented way, so as to master the loop structure.
Task 3: Reconstruct task 2 using object-oriented thinking.	encapsulation, software modeling language, object-oriented design principles	Introduce object-oriented thinking, let students understand the concepts of abstraction and encapsulation, and learn to build classes and objects.
Task 4: Refactor task 3 using inheritance.	inheritance	Analyze the similarities and differences of computing objects, abstract the base class and the derived class. Use inheritance to reconstruct task 3 to enable students to understand what software is good for reusability.
Task 5: Refactor task 4 using the simple factory pattern.	polymorphism, simple factory model	Use polymorphism and simple factory model to reconstruct task 4, students can understand what software is easy to extend.
Task 6: Refactor task 5 using the factory pattern.	factory mode, interface	Reconstruct task 5 with the factory model to make up for the shortcomings of the simple factory model that does not conform to the open and closed principle, and make the software easier to expand.
Task 7: Use the visualization platform to reconstruct task 6, expand the calculation function, and realize a scientific calculator.	singleton mode, visualization	Using singleton mode and using visualization platform to reconstruct tasks 6, let students understand the conversion of input and output modes and the reusability of software under different platforms.

III. TEACHING PRACTICE OF PROGRESSIVE CASE

Although the calculator case is simple in function, it involves a lot of teaching contents and is permeable. In order to cultivate students' software design ability, the author divides the teaching process of "Software Modeling and Practice" into three stages.

In the first stage, students' programming ability and code specification are mainly cultivated. This stage explains and implements task 1 and task 2. The teacher teaches the related knowledge points involved in the two tasks, and guides the students to complete task 1 and task 2. This stage is based on basic knowledge, step by step, so that students are familiar with and understand the program structure and related grammar knowledge, and develop good code specifications.

The second stage is to train students to solve problems. This stage explains and implements task 3 and task 4. Teachers first instill in students the idea of object-oriented, layered design, and let students establish concepts such as abstraction, encapsulation, and inheritance. Then design the program hierarchically to separate business logic and interface logic. Next, we abstract task 3, design a calculation class and implement it. The computer logic is further layered, the computing base classes and derived classes are designed using the idea of inheritance. This stage equips students with the ability of object-oriented analysis, object-oriented design and object-oriented realization.

The third stage is to train students to build software that is easy to reuse, expand, and maintain. In this stage, explain and implement task 5, task 6, and task 7. Through the analysis of the shortcomings of the former task, the software is reconstructed with design patterns to achieve the purpose of easy reuse, easy expansion and easy maintenance of the software. This stage allows students to experience the differences in software quality caused by different design patterns to solve the same problem, and cultivate students' interest and development ability.

The progressive case is implemented in three stages from the shallower to the deeper, which allows students to establish software design thinking, thereby paving the way for students to engage in software design work in the future.

IV. CONCLUSION

The progressive case teaching method in the process of teaching reform experiment on 324 students of computer science and technology major from grade 16 to grade 19, the classroom atmosphere is active and the students are interested in it. 90% of the students can independently complete the software design tasks assigned by the teacher and design own unique software. At the scene of the software design defense, the atmosphere is very active. Students are very proud because of they can independently realize the software which is easy to reuse, easy to expand and easy to maintain. It can be found that the progressive case teaching method has achieved good teaching effects, achieved the training purpose and requirements, and has a great promotion value.

ACKNOWLEDGMENT

This work was funded by the Elite Talent Project of Xi'an University of science and technology.

REFERENCES

[1] Pang Guangyao. The application of progressive single-case teaching method in Java curriculum[J]. *Asia Pacific Education*, 2015(07): 83-84.

[2] Liu Dabin, Xiong Mingli, He Mei. A teaching case study of "Principles of Machinery" and "Mechanical Design" based on the flipped teaching model[J]. Mechanical Design, 2018, 35(S2): 110-113.

[3] J. Chen, Y. Wang and Y. Yang, "The Application of Case-based Task Driven Teaching in Teaching of Computer Studies."2019:341-344.

[4] L. Caixia, G. Yangyang, S. Wangqin and S. Yan, "The Application of Case Teaching Combined with Task-Driven Teaching Method in Obstetric Nursing Teaching," 2016 8th International Conference on Information Technology in Medicine and Education (ITME),2016:790-794.

[5] S.Tao,"Construction and Implementation of Modular Situational Teaching Model in Case Teaching Method on TCM Teaching in Medical Colleges and Universities," 2015 7th International Conference on Information Technology in Medicine and Education (ITME),2015:408-412.