ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202406_7(6).0008

Teaching Design of "Architecture Design" Based on Case Teaching

Yadong Gong[1, a]

[1]School of Computer Science and Software, Zhaoqing University, Zhaoqing 526061, P.R.China

[a]371126472@qq.com

Abstract

Based on the existing problems in the traditional teaching procedure of "Introduction to Software Engineering" course, we take the "architecture design" section as an example, and adopt the case teaching method to enhance student learning. In the teaching procedure, the attention of students is attracted by the case discussion or questions. Then, further thinking is triggered by group-based activities. In the end, The teacher conclude the results of the case discussion or the questions. Based on the results of teaching practice, the teaching goal, such as the ability to learn and apply knowledge, has been achieved.

Keywords

Introduction to Software Engineering, architecture design, case teaching.

1. Introduction

In the current educational context, students cultivated by universities need to possess the following abilities: a certain level of engineering practice capability and innovative practice capability. This requires students not only to acquire the corresponding knowledge and skills in daily teaching activities but also to develop higher-level abilities such as problem identification and resolution, a certain level of innovation capability, teamwork ability, and analytical practice skills[1,2].

"Introduction to Software Engineering" is a core course for the Software Engineering major, which helps students understand the processes, job responsibilities and positions in each process of software development. It is an important supporting course for computer science students to determine their career planning. Due to the strong theoretical nature, scattered knowledge points, and weak practicality of the software engineering course, the curriculum reform of software engineering is imminent. Without changing the theoretical nature of this course, it is necessary to organically integrate practical content[3].

Through the investigation of the teaching methods of "Introduction to Software Engineering" courses in domestic universities, the following two aspects of deficiencies in teaching methods were found[4,5]:

Theoretical knowledge is detached from actual enterprise practices. The "Software Engineering" course is positioned as a theoretical course in most universities. However, essentially, the "Software Engineering" course is one that extracts theories from practice; theory comes from practice, and practice supports theory. The textbooks used by most domestic university teachers for the "Software Engineering" course are classic, with a solid knowledge framework and good reference value. However, some young teachers, due to their lack of work experience in enterprises, tend to teach by rote in the teaching process, focusing only on the teaching of theoretical knowledge and lacking specific practical projects to support the theoretical knowledge. As a result, students have a superficial understanding of the related knowledge of software engineering, and the guidance for students to engage in related fields after graduation is insufficient.

The subjectivity of students is not reflected. The teaching method of the "Software Engineering" course mainly adopts the PPT lecture method. Teachers impart knowledge according to the course outline and textbooks, and students participate in teaching activities following the teacher's pace. This shows that students passively receive knowledge, with little teacher-student interaction in the classroom, and the subjectivity of students is not brought into play. Students learn according to the teacher's wishes without a clear goal, and their motivation to learn is not strong. Moreover, since the software engineering course does not have a corresponding practical course and mainly follows a teaching method focused on theoretical teaching, it lacks matching practical courses. This fails to reflect students' creativity, cannot test the degree of students' learning, and cannot stimulate students' motivation to learn, which can easily lead to a situation where students have good exam results but poor application ability. Such teaching outcomes are contrary to the teaching objectives.

In order to enhance students' interest in the course and improve their abilities to learn and apply knowledge, we have adopted a case-based teaching approach. In this paper, we take the "architecture design" section as an example to demonstrate the teaching design of the course.

2. Teaching Goals

Teaching objectives include the following four aspects:

To understand the tasks of architecture design;

To master the process of architecture design;

To grasp the design principles and heuristic rules of architecture design;

To master the data-flow oriented design methods.

3. Teaching Strategies

This is a theoretical course on software architecture design, covering five aspects: the architecture design process, design principles, heuristic rules, graphical tools for describing software structure, and data-flow oriented design methods. To stimulate students' interest in learning and facilitate their understanding and application of knowledge, we plan to adopt the following teaching methods:

Before introducing the content of software architecture design, we first introduce architecture design through the various stages of a software development case implementation, along with the main tasks of that phase. Then, based on the requirements of the case project, we ask students to attempt designs and analyze the impact of different design outcomes on software quality and maintainability, thereby highlighting the significance of learning in this chapter. At the same time, by utilizing these cases, we discuss the practical importance of software designers to help students improve their professional skills actively.

We use the lecture method to help students understand the concepts involved in design principles, including abstraction, gradual refinement, information hiding, module independence, coupling and cohesion, and then, demonstrate the application of these concepts with examples.

We use a poorly designed software structure as an introduction, guiding students through discussion to spark their interest in learning about software structure design. We also explain each heuristic rule in architecture design with examples, ensuring students have a correct and clear understanding of the content of heuristic rules.

We first introduce graphical tools for describing software structure and then, based on specific examples, explain the steps and methods for mapping software data flow diagrams to the software structure, enabling students to understand and apply this knowledge.

We set up a comprehensive application segment. We ask students to use the knowledge and methods learned in this chapter to design a corresponding software structure for a brief software requirements specification and to present its structure diagram using graphical tools, thereby mastering the method of architecture design.

4. Teaching Procedure Design

Step 1:

- Teaching goal: Understanding the Significance and Tasks of Architecture Design;
- Teacher Activity:
  - ✓ Introduce the various stages of a sample software development project to lead into the architecture design phase;
  - ✓ Explain the two main tasks of the architecture design phase: determining the general solution for the software and designing the software structure;
  - ✓ Illustrate the significance of the architecture design content by discussing the impact of different design outcomes on software quality and maintainability.

Step 2:

- Teaching Goal: Mastering the Design Principles in the Architecture Design Process;
- Teacher Activity
  - ✓ Interpret each rule of the design principles;
  - ✓ Use examples to explain the specific application of design principles;
  - ✓ Pose some architecture design problems for students to solve, to test their understanding of the knowledge points.

Step 3:

- Teaching Goal: Mastering the Heuristic Rules in the Architecture Design Process;
- Teacher Activity:
  - ✓ Provide a poorly designed software structure for students to discuss;
  - ✓ Introduce the concept of heuristic rules in architecture design based on the discussion results;
  - ✓ Explain the meaning of each heuristic rule with examples.

Step 4:

- Teaching Goal: Mastering the Data-Flow Oriented Software Design Method;
- Teacher Activity:
  - ✓ Introduce graphical tools for describing software structure diagrams;
  - ✓ Explain the two types of data flow diagrams: transformation flow and transaction flow;
  - ✓ Introduce the basic methods for mapping data flow diagrams to software structure diagrams;
  - ✓ Demonstrate the method and steps for mapping transformation flow to software structure diagrams with examples;
  - ✓ Require students to work in groups to map a transaction flow example to a software structure diagram, and then, review the design results.

Step 5:

- Teaching Goal: Mastering the Comprehensive Application of Knowledge;
- Teacher Activity:
  - ✓ Summarize all the knowledge points involved in architecture design;
  - ✓ Require students to work in groups to complete the architecture design of a case project;

- ✓ Give comments on the project results completed by the students.

5. Conclusion

This paper firstly introduces a case teaching method based on the existing problems in the teaching process of the course " Introduction to Software Engineering ". Then, we present the teaching objectives, teaching strategies and teaching process design of the above course by taking the "architecture design" section as an example. After the teaching practice in the class, it has been demonstrated that our teaching model can enhance the students' interest in learning new knowledge, and also greatly improve their abilities to learn and apply new knowledge.

Acknowledgements

This work was supported by the 2023 Educational Teaching Research and Reform Project of the Online Open Course Consortium of Universities in the Guangdong-Hong Kong-Macao Greater Bay Area under Grant WGKM2023162.

References

[1] Tong Yujun, Yi Huawei, Chen Xin, et al. Research and Practice of Ideological and Political Education in "Software Engineering" Course [J]. Journal of Liaoning University of Technology (Social Science Edition), 2024, 26(01): 112-114.

[2] Shen Liwei, Peng Xin. Exploration of Software Engineering Teaching Method Centered on Practical Ability [J]. Software Guide, 2023, 22(12): 1-6.

[3] Jiang Ying, Wang Hongbin, Ding Jiaman, et al. Exploration of Software Engineering Course Teaching to Promote Ability Enhancement with Practice Orientation [J]. Software Guide, 2023, 22(12): 25-29.

[4] Xiao Bo. Exploration of Software Engineering Course Teaching Based on CDIO and Task-driven [J]. Computer Era, 2023(12): 209-212.

[5] Mao Xinjun, Sun Yanchun, Chu Hua, et al. The Construction Philosophy and Achievements of "101 Plan" Software Engineering Course [J]. Computer Education, 2023(11): 29-33.