Teaching Reform and Practice of Software Architecture for Postgraduates in Universities

Le Wei(✉) and Qiuyun Zhao

School of Software Engineering, Chengdu University of Information Technology, Chengdu 610225, China
weile@cuit.edu.cn

Abstract. Software architecture is a core course for postgraduates of the first-level discipline of software engineering, with a high degree of abstraction, complex content and strong practicality. Aiming at the current problems of outdated course content, disconnection between theory and practice, and difficulty in achieving course objectives in teaching, as required by academic/professional degree graduate core curriculum guidelines, the course content is optimized from five aspects: software architecture overview, software architecture style, software architecture description and documentation, software architecture design and software architecture evaluation, designing teaching cases of course, and giving case-driven teaching implementation method.

Keywords: Software Architecture · Software Architecture Design · Teaching Case Design · Case-driven teaching

1 Introduction

With the rise of software definition and the deep convergence of software and industry, the scale of the software system is expanding, and the complexity is also increasing rapidly. This has led to increasing difficulty in developing software products that meet the functional and non-functional requirements of users, which is reflected in the difficulty in ensuring the quality of software products, the continuous lengthening of software development cycles, the high cost of software development, and the persistence of software crises. Software architecture, also known as software architecture, is a prefabricated and reconfigurable software framework structure, which provides a high-level abstraction of the structure, behavior and properties of the software system. As an early design decision of software system, software architecture plays an important role in the whole software life cycle. For example, in the system design stage, software architecture is the basis for system design decomposition, implementation and verification; in the project implementation phase, software architecture is the basis for division of work, personnel arrangement, organization and coordination, and performance management; in the project testing phase, the software architecture is the basis for performance testing and review; in the maintenance and upgrade phase, the software architecture is the basic model for software system modification, expansion and upgrade. Therefore, the training of software architects with the ability to build the core architecture of the system, clarify technical details, and clear major difficulties [1] has become an important goal of the professional talents training in software engineering in China. In the short term, the training of software architects can help software enterprises improve the quality of software products, reduce development costs, promote the timely delivery of projects, and improve the rationality of project management. Academic education provides the necessary and systematic knowledge system for software architects, and is the main way to train software architects. Therefore, it is of great significance to carry out in-depth research on the training methods and approaches of software architects in academic education. It is beneficial to cultivate talents needed by the industry, improve the software design ability, promote high-end software research and development, and promote the development of the software industry.

Postgraduate education plays an important role in cultivating high-level specialized talents with innovative ability, entrepreneurial ability and practical ability. Therefore, it is important to cultivate advanced engineering and technical talents with applied software architecture design ability to adapt to the development needs of software industry. In 2020, the Office of the Academic Degrees Committee of the State Council organized public publication Core Course Guide for Academic Degree Postgraduates (Trial) and Core Course Guide for Professional Degree Postgraduates (Trial), clearly pointing out that the software architecture course should be offered by the master students of software engineering, and preliminarily clarifying the teaching objectives and contents of the course. Currently, domestic universities have carried out relevant research on how to improve the teaching quality of software architecture courses and have gained a series of experiences. Tongji University [2] gives software architecture curriculum construction plan, from three aspects: content, course practice project, and teaching effect evaluation; Zhejiang Normal University [3] practices the teaching method of flipped classroom in the course of software architecture; Nanjing Audit University [4] practices cross-case practice teaching method in software architecture course; Xidian University [5] has carried out the teaching exploration and practice of software architecture course based on mutual assistance; Shandong Agricultural University [6] applies the theory of learning transfer to the teaching of software architecture; University of Science and Technology of China [7] proposes a methodology for teaching software architecture based on agile software development. National University of Defense Technology [8] has proposed and implemented a software architecture curriculum teaching programs based on case-based teaching, flipping of classrooms and open source software.

Taken together, many universities have carried out a series of research and practice in the teaching of software architecture courses, and have achieved some results. However, duo to the software architecture course has the characteristics of high degree of abstraction, complex content, strong practicality, as well as the short duration of the course, the problem that the course teaching cannot meet the requirements of the industry for software architects always exists, and shows a trend of gradually increasing the gap. Hence, this paper aims at the training of master's students majoring in software engineering in ordinary universities, combined with the core course guide of academic (professional) degree graduates, to conduct research on the teaching content and teaching methods of software architecture courses, and explores effective ways to cultivate software architects who meet the needs of the industry.

2 Reform of Course Content

Software architecture involves software engineering, network, database, operating system and many other knowledge, and the course content is complex; the course also lacks mature and classical teaching materials and teaching cases. Considering the few credit hours allocated to the course at the postgraduate level, universities must combine their own professional talent cultivation objectives and positions, referring to the core curriculum guidelines for academic (professional) degree postgraduates, and reasonably design the course content under the premise of cultivating qualified software architects. The purpose of the software architecture course is to introduce the principles and methods of software architecture connotation, software architecture modeling, software architecture design methods, analysis and evaluation, and through case studies, students are cultivated with the ability to design and analyze the architecture of software systems of a certain scale, so that the students can make the best architecture design decisions for realistic concrete systems, thus adequately develop the students' ability to think abstractly, analyze and design global-oriented systems, apply knowledge to solve practical problems, and think independently and innovatively[9]. As a consequence, the software architecture course of our university mainly covers five aspects: software architecture overview, software architecture style, software architecture description and archiving, software architecture design and software architecture evaluation.

2.1 Software Architecture Overview

This part starting with the software crisis, the definition of software architecture, development history, research content, role, research significance, research status, development direction and responsibilities of software architects are introduced, so that students can have a preliminary understanding of software architecture, understanding that software architecture has an important impact on the success or failure of software projects, and understanding the knowledge required for software architects.

2.2 Software Architecture Style

Software architecture style describes the customary pattern of how families of software systems are organized in a particular domain, reflecting the structural and semantic characteristics common to many systems in the domain, and guiding how modules and subsystems can be effectively organized into a complete system, so software architecture style is one of the key elements of the course. This part includes classic architecture style and mainstream architecture style, where classic architecture styles include data flow style (batch processing sequence; pipeline/filter), call/return style (main program/subprogram, object-oriented style, hierarchical structure), independent component style (process communication, event system), virtual machine style (interpreter, rule-based system) and repository style (database system, hypertext system, blackboard system). Mainstream architecture styles include network-based software architecture style, platform/plug-in architecture style, client/server style, service-oriented architecture style, cloud architecture style, big data architecture style and microservice architecture style (Serverless, ServiceMesh).

2.3 Software Architecture Description and Archiving

The main function of software architecture description is to use graphics, text, mathematical expressions, etc. to describe the main characteristics of software, mainly including graphical visualization methods and ADL methods. This part first introduces the software architecture description method, then introduces the software architecture description language ADL, the use of UML to describe the software architecture, and the use of 4 + 1 model to describe the software architecture, and finally introduces the method of writing software architecture documents.

2.4 Software Architecture Design

Software architecture design is the key to software development, so this part is also one of the main contents of the course. This part first introduces the characteristics of software bad design and the basic principles of software design; then focus on object-oriented design methods, object-oriented design principles and common design patterns(Abstract Factory, Factory Method, Singleton, Adapter, Composite, Façade, Proxy, Command, Iterator, Observe, Strategy) will be explained; subsequently, based on a brief introduction of artifact-driven, use-case-driven, pattern-driven, and domain-driven approaches, key quality attribute-driven architecture design approach (ADD, Attribute Driven Design) will be explained emphatically, and the key quality attributes are summarized as Availability, Modifiability, Performance, Security, Testability, and Usability, with descriptions of key quality attribute scenarios and related strategies for improving key quality attributes, and analysis of software architecture design steps driven by key quality attributes with cases.

2.5 Software Architecture Evaluation

Evaluation of the software architecture is the key to determining whether the chosen architecture for the software system is appropriate and to ensure that a successful software product can be developed successfully according to the chosen architecture. This part first introduces the main approaches to software architecture evaluation, and then focuses on the specific implementation steps of the ATAM evaluation methodology.

3 Reform of Case-Driven Course Teaching Methods

The basic concepts, principles and methods of software architecture are the extraction and sublimation of common features of software systems in many different domains [10]. Comparing with the software engineering courses that students are usually exposed to, the content is more abstract and difficult for students to understand and comprehend. Meanwhile, software architecture is a course with strong practicality. So, it is difficult for students who is in the absence of practical software project design experience to understand abstract principles and methods, and far from the application of theoretical knowledge to complete the architecture design of a large complex software system. Case-driven course teaching is an open and interactive new teaching method, which incorporates students into case scenarios by simulating and reproducing some scenarios in life, so that the connotation and value of the learned knowledge points [4] will be deeply appreciated by students. As a consequence, the introduction of the case study method into the teaching of software architecture will promote students' mastery of theoretical knowledge on the one hand, and help students to clarify how theory is integrated into practice through cases on the other hand, so as to improve students' engineering practice ability.

3.1 Design of Teaching Cases

The teaching cases are designed according to the principle of "moderate difficulty, inclusive theory and hierarchical progression". Moderate difficulty means that the designed cases should meet the actual knowledge level of most students, because master students come from different undergraduate universities, and the curriculum system at the undergraduate level is not the same, so it is necessary to design targeted teaching cases on the basis of researching students' mastery of knowledge, and cases that are too difficult or too easy will not achieve the expected effect. Inclusive theory means that the teaching case should be able to cover the main theoretical knowledge involved in the course, because the course explains the general knowledge of software architecture, so it is unrealistic to design a case that covers all the knowledge points, but if the case covers too few knowledge points, it cannot play the role of combining the theory and practice of the case. Hierarchical progression means that the designed cases should cover the three levels of "point-line-surface", including cases covering single knowledge points, cases covering multiple knowledge points, and comprehensive cases covering most of the knowledge points. In particular, it should be pointed out that through case is a comprehensive case, which can be decomposed to achieve the integration of "point-line-surface". According to the design principles, a series of teaching cases had been designed, such as for each software design pattern, relevant design cases were designed, including the use of system operation logger cases suitable for factory method mode, skin library cases suitable for abstract factory mode, toy car control software cases suitable for adapter mode, antivirus software cases suitable for combination mode, etc.; a case of a communication business system supporting coal mine safety production and a case of a garage door control system were designed for the architecture design method driven by key quality attributes. The case of "key word in context (KWIC)" was designed according to the classic architecture style; MapReduce cases were designed for big data architecture styles; the Alibaba Cloud cases were designed for the cloud computing architecture style; industrial APP cases were designed for the microservice style; online shopping system was designed for client/server style; academic Management System and Library Management System cases were designed for software architecture description.

3.2 The Practice of Case-Driven Teaching Method

Based on the designed cases, the case-driven teaching method had been adopted in the software architecture lectures as follows:

1. Grouping of students
   The whole class will form a group of 3–5 students on a voluntary basis, and each group will elect a leader. The group will be divided into different roles such as requirements analyst, architect, module developer, document writer and architecture evaluator according to the practice of software engineering projects.
2. Case-driven theory teaching session
   In the theoretical classes, teachers combined case studies to explain the relevant knowledge. The lectures focus on the guidance of students, and through questions, group discussions, flipped classroom methods to increase the degree of student participation in teaching, inspiring students to think, closing the gap between the classroom and the real project. Outside of class, similar cases would be issued to students as practice tasks, and students would be asked to work in groups to complete the analysis of the cases, thus deepening their mastery of the theoretical knowledge.
3. Case-driven practical teaching session
   Considering that our software architecture course only has 32 h, there are no dedicated practical hours in the course. However, in response to the fact that students had already completed their undergraduate level courses, we put the practical sessions outside of class and require students to complete a software architecture design in groups, including tasks such as requirements analysis, architecture style selection, top-level architecture design, enhancement strategies for key quality attributes, class/package/service design and database design, writing relevant documents, describing them according to the 4 + 1 view or UML, evaluating each other by cross-evaluation between groups, and presenting a defense, which is counted in the examination. The project is completed by students themselves. Teachers only need to control the overall direction and provide necessary guidance for answering questions, but will not involve details. Through the introduction of cases in the practice link, enthusiasm and initiative of students in learning can be brought into full play and their engineering practice ability could be improved.

4 Conclusion

Software architecture design is an important software design decision made by software architects from a macro and global perspective in software life cycle activities. Software architecture is a course on the design and analysis of the overall high-level structure of large and complex software systems. In accordance with the requirements of the core curriculum guidelines for academic/professional degree graduate students and the actual situation of Chengdu University of Information Technology, the teaching contents of the software architecture course were optimized, the design principles of teaching cases and typical cases were given, and the case-driven course teaching method was practiced. Through the reform of the course, the abstract thinking ability, engineering practice ability and innovation ability of students were cultivated, and a favorable technical foundation was laid for students to work in software enterprises and engage in scientific research in the field of software engineering.

Acknowledgement. This work was support by the 2021 Graduate education reform project by Chengdu University of Information Technology (CUITGOKP202106).

References

1. Y. Sun, S. G. Chen, Z. G. Wang, et al. The course construction of "Software System Architecture" based on real project cases, Journal of Jinling Institute of Technology (Social Science), vol. 32, pp. 58-62, 2018.
2. Y. Shen, L. Zhang. Teaching practice of Software architecture for graduate courses, Computer Education, pp. 140-144, 2017.
3. Z. G. Ding. The practice of flipped classroom in Software Architecture course teaching, Computer Education, pp. 68–71, 2017.
4. J. J. Huang, P. W. Li. The application of penetrating case and practical teaching method in Software Architecture course, Software Guide, vol. 19, pp. 249-251, 2020.
5. Y. S. Lin, Q. S. Li, L. Bao, et al. Exploration and practice of teaching method of Software Architecture based on mutual learning and application, Software Guide, vol. 21, pp. 11-13, 2022.
6. Q. Zhou, Z. J. Wang. The application of learning transfer theory in Software architecture teaching, Computer Education, pp. 175–178, 2019.
7. J. Ding. Research on Software Architecture Teaching based on Agile Development, Computer Education, pp. 59–62, 2018.
8. T. Li, Y. J. Wen, W. W. Liu, et al. Software Architecture course teaching reform planning and implementation, Computer Education, pp. 19-21, 2015.
9. National Professional Degree Graduate Education Steering Committee. Core Curriculum Guide for Professional Degree PostGraduates (I) (Trial), Beijing: Higher Education Press, 2020, pp. 419-421.
10. T. K. Li. Teaching content design of applied undergraduate Software Architecture course, Computer Education, pp. 120–123, 2018.