Advances in Social Science, Education and Humanities Research, volume 352

International Conference on Advanced Education, Management and Humanities (AEMH 2019)

Teaching Reform and Practice of Software Architecture Design Course under the Background of Engineering Education

Weigang Li

School of Software, Northwestern Polytechnical University, China

Keywords: Software architecture, Engineering education, Flipped classroom, Teaching reform.

Abstract. The increasing scale and complexity of software architecture in the Internet era is a great challenge for the teaching of software architecture design course. The course content and its teaching mode must be reformed to match the technology progress and the new software development practices. We report our teaching reform based on the case-based teaching, flipping classroom and project practice in this paper. The practice process and effect of teaching reform in senior undergraduates of software engineering specialty are expounded.

Introduction

In the Internet era, with the continuous expansion of software scale, the importance of software architecture design in the development of large and medium-sized software projects is becoming more and more prominent. Software architecture design course has become an important professional course in software engineering disciplines [1]. Its purpose is to improve students' ability to understand the problems and plan the solutions at the conceptual level, and to transform the learner from programmer to architect.

In the software architecture point of view, it separates the overall structure of the system (including components and their connections) from the internal details of each component, and emphasizes the external view of components, that is, the interfaces provided and needed by components and the connections with other components. Therefore, architecture design is a high-level design, which needs to focus on the quality attributes of the system, including performance, security, maintainability, etc., while meeting the functional requirements. It is the starting point for detailed design and implementation. At present, the demand for software architects in industry is very urgent and high. This puts forward a new requirement for the teaching of software architecture design course, that is, close to the actual software development practice, so that students can truly grasp and flexibly use the principles and methods of software architecture design.

Course Characteristics and Difficulties in Teaching

The software architecture design course has characteristics as below [2]:

1. Knowledge covers a wide range. The activities in each stage of software architecture design involve a lot of knowledge learned in other courses, such as network, database, operating system, distributed system, software engineering, etc. It is a comprehensive consideration and application of these knowledge. So it is important for the student to learn these courses well.

2. The degree of abstraction of curriculum knowledge is high. Software architecture design focuses on exploring the macro level of software, which is very far from the specific coding that students are very familiar with. If it relies mainly on classroom teaching, it is difficult for students to understand and master the design methodology.

3. The balance between theory and practice. Theoretical knowledge is the general knowledge that can be applied to all software architecture design. Practice requires a compromise between theory and real system as real as possible, so as to combine general knowledge with specific software projects and achieve better learning results.

Software architecture design course of Northwestern Polytechnical University is offered to junior undergraduates majoring in software engineering. Although the students have systematically studied the basic courses and most of the specialized courses of software engineering, the experiences of software research and development is limited to the homework or training projects of some specialized courses, without the training of medium-sized and large-scale software development, and lack of project experiences under the real scene.

In recent years, we have found that using all the classroom time to teach abstract theory of software architecture design makes students feel bored. It is prone to turning this course into a course that needs to memorize a lot of principles but not being applied in practice. It is difficult to reflect and understand the important position of software architecture design in software development. In addition, after investigating most of the textbooks, we find that the cases in the textbooks are small in scale and only suitable for teaching, but not for students to practice. Finally, the class hours are limited, only 40 theoretical hours and 16 experimental hours. How to make good use of the class hours and longer teaching cycle and arrange the classroom and practical content has always been one of our key considerations.

In view of the characteristics and problems of the course, we propose to improve the course teaching from the aspects of classroom teaching content, teaching form and practical links. Specifically, the teaching contents are mainly summarized after the students doing, the teaching form is mainly case-based teaching, and in the practical links the design and analysis of real software projects from enterprises are focused. We combine the three as far as possible. In a more real environment, the students experience and use software architecture design theory and technology in specific projects, so as to improve the teaching effect.

Teaching Form Reformation and Practice

The teaching plan is as follows: 20 lessons for basic knowledge of software architecture design; 8 lessons for discussion of 4 design cases; 4 lessons taught by enterprise experts; 16 lessons for students to practice the whole design process of 4 cases discussed in the classroom, and 8 lessons for students to practice the architecture design of a real project from the enterprise and report the results of practice.

The above contents are arranged in sequence, and the experimental links are synchronized with the teaching of the course. In the experiment teaching, we adopt the way that students design first and then teachers teach. The real project experiments and the teaching of invited enterprise experts do not affect each other.

Combination of Case-based Teaching and Flipping Classroom

In the course of teaching, teachers can first teach the basic knowledge of software architecture design, including definition, view, software quality attributes, software architecture design method and process. The theoretical teaching in this stage is indispensable, and it is the basis of subsequent courses and practice. At this stage, more theoretical knowledge should not be introduced, because the knowledge introduced is enough for students to start preliminary design practice. While the introduction of more knowledge, such as software architecture design patterns, will increase the burden of students rather than helpful to practice.

When introducing the design method and process of software architecture, teachers can choose the ATM system implemented by Client/Server architecture style as a case study [3]. In the classroom, the teachers will introduce in detail how to get the process and method of the final architecture design from the analysis of software requirements, and at the same time they will insert a lot of in-class exercises. For example, for all the use cases of ATM system, a case is selected as a classroom demonstration. After introducing the design of static model, dynamic model and state machine for the use case in detail, students are required to complete the design of other use cases of ATM system in class in time, laying a good foundation for the follow-up practices.

In the follow-up teaching, teachers can mainly conduct case-based teaching in the form of flipped classroom. The teaching cases in this stage are mainly collected from various textbooks. The purpose is to gradually enable students to gradually adapt to the design task of software architecture design.

The cases we selected include online sales system based on service-oriented architecture style, emergency monitoring system based on component software architecture design pattern, automatic driving system based on real-time software architecture design, etc. In this teaching link, for each case, teachers can adopt the following teaching mode.

1. Instead of introducing the theoretical knowledge related to the case in class, each case will be assigned to the students in the form of homework in order. Each student has one week to analyze the case and complete the design tasks required by the homework. The new knowledge involved in the homework needs the students to study by themselves.
2. Students are required to hand in their homework before the next lesson. In the next lesson, the case of the last arrangement is explained in detail and introduced to the students in the form of reference design.
3. At each design stage of the case, the common problems in students' homework are pointed out and students are guided to compare their homework with reference design to find out the shortcomings.
4. In the course of case teaching, the knowledge of software architecture design pattern is gradually introduced.

This teaching mode will be carried out for three rounds, and the selected cases will be taught and practiced separately. Through this teaching link, students can enter into the practice of larger projects in the early stage of the course, without waiting for all the theoretical knowledge to be introduced. Secondly, every design stage and result of each case has been thoroughly considered and practiced in the homework, so they can immediately grasp the key points of design in the classroom and find the shortcomings of their own design. Finally, the teacher introduces various software architecture design patterns with specific cases to materialize abstract knowledge, so that students can understand the meaning of architecture style through learning several architectural styles.

Practice with Real Software

After students have mastered and practiced the design method and process of software architecture, and have seen and understood some common software architecture design patterns, teachers can guide students to practice with real large-scale software. We chose the actual software project developed by the software enterprise which cooperates closely with our university after dealing with it as the main project of curriculum practice. Firstly, the students are divided into three to four groups. Each group is required to select a software from seven projects as the object of practice. The common characteristics of these projects are large scale, with many users and many documents, which are convenient for students to study and analyze. In practice, students will experience the following stages.

1. Understanding the project. For the selected project, the students must read the document firstly to understand the software requirements, application background, architecture overview and other knowledge.
2. Identifying stakeholders. The students must analyze of the selected project carefully, imagine as an architect to develop such a project. He or she must ask him or herself some questions such as which stakeholders will be involved in the project, what are their respective interests, etc.
3. Identifying quality attributes. According to the quality attributes and their measurements, the main and potential quality attributes requirements of the selected items are analyzed, and the quality attributes are modeled by the methods introduced in class.
4. Architectural description. According to the "4 + 1" view, the architecture view of the selected software is described and depicted.
5. Architectural evaluation. According to the extracted architecture design and quality attributes, the students in the group play different stakeholders and practice software architecture design evaluation activities.
6. Source code analysis. Read and analyze the source code of the selected project at the module level, draw the software structure diagram at the module level, and map with the view of software architecture design.

These practices last for five weeks. The instructors and counselors follow the activities of each group throughout the course. Through this link, students can use real large-scale software projects to experience activities related to software architecture design, including requirements analysis, architecture design and evaluation. Through reading source code and comparing with architecture design, students can experience how to reflect the decision-making in the architecture design practices. Through the modeling of quality attributes and architecture evaluation activities, students can observe how to experience the compromise in architecture design.

Joint School-Enterprise Teaching

While using real software for practice, we invite front-line architects of cooperative enterprises to give lectures. The experts are from various industries with rich commercial projects experiences, such as hospital information management system, health information management system, online photo sharing system, intelligent building monitoring system, etc. These projects have practical application background, users, stakeholders, quality attributes, etc. They are no longer imaginary cases in textbooks. Experts will introduce the whole software development process from requirements to architecture design and software release with each specific case. They will take students to think and design together and arrange appropriate in-class exercises in class. In line with theoretical knowledge, experts focus on the specific considerations and trade-offs in the architecture design stage and their reasons. At the same time, experts will also talk about their understanding of software architecture design and some insights into software architecture design from their own point of view, such as software architecture design is not a design, but the result of repeated design iterations.

Through this course, students will be able to see the work of front-line architects and their role in software teams. Front-line architects teach software architecture design in combination with specific projects in plain and concrete language, which is easier for students to accept and understand. Contrasting with the theoretical knowledge learnt in class, students can discover a deeper meaning of theoretical knowledge and find the application of theoretical knowledge in specific projects.

We think that the most influential thing for students is the specific application method of general principles in software engineering. Each expert has his own unique analysis and design method. Although it looks different, the embodiments of design principles such as software engineering and software architecture design are consistent, which makes students realize how flexibly the general principles learned in class are embodied in the design.

Reform Effect

Since 2016, we have implemented this teaching reform program in the course of "Software Architecture Design" for software engineering specialty. The number of students in the three sessions is 76, 81 and 51 respectively. Classroom teaching is carried out in strict accordance with the scheduled hours. Teachers and counselors follow up and guide the whole course, and find problems in students' learning at any time. The whole course is completed within 10 weeks. Questionnaire survey shows that compared with the previous teaching students have a deeper understanding of software architecture design and its related technologies and methods, which is highlighted in the following aspects.

1. By arranging experiments, students can start to practice as soon as possible while learning theory. The application of software architecture design method and technology is closely followed by theoretical learning, and the abstract knowledge is embodied.

2. In case selection, from textbook cases to actual open source projects, step by step, students can immediately use the knowledge they have learned in various experiments, and there will be no situation that they cannot start. Experiments on practical open source projects can test students' comprehensive application ability and enhance their confidence.

3. In the way of teaching, do it first and then talk about it. It can bring more challenges to students than simple preview and reading materials. Students have more questions in class and pay more attention during class. They can find their shortcomings and improve them in time.

4. The teaching of front-line architects is very popular with students. Students realize how to use abstract principle knowledge in practice: follow the principle, and operate flexibly in practice.

Summary

We have explored and practiced the teaching of software architecture design course. The designed teaching content, teaching form and arrangement of experiments are very effective to improve the teaching effect. But in practice, we also find that in order to do this teaching plan, it puts forward high requirements for teachers and students. Every year, we need to redesign teaching cases and invite front-line architects from software companies. The workload is very large. For students, it takes a lot of time to persist in completing all the experiments. If they persist, they will get some results, but there is plagiarism in the actual teaching. In the future, in view of the new problems, we will continue to explore the teaching methodology of the software architecture design course, and seek better training model to achieve better training effect.

Acknowledgement

This research was financially supported by the Teaching Reform Project and High Level Course Construction Project of Northwestern Polytechnical University.

References

[1] T.K. Li, Teaching Content Design of Software Architecture Course in Engineering Collage, Computer Education. 6 (2018) 120-123.

[2] H. Li, Y.J. Wen, W.W. Liu, W. Dong, Y. Luo, Planning and Implementation of Software Architecture Course Teaching Reform, Computer Education. 6 (2015) 19-21.

[3] H. Gomaa, Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures, Cambridge: Cambridge University Press, 2011.