Exploration on Theoretical and Practical Projects of Software Architecture Course

Li Zhang
School of Computer Science and Engineering
Beihang University
Beijing, China
lily@buaa.edu.cn

Yanxu Li
School of Computer Science and Engineering
Beihang University
Beijing, China
liyanxu2018@buaa.edu.cn

Ning Ge*
School of Software
Beihang University
Beijing, China
gening@buaa.edu.cn

***Abstract*—It has been widely recognized that software architecture is one of the hard courses to teach in software engineering. First, it is usually difficult to update the knowledge written in textbooks with the rapidly developed practical progress. Accordingly, we need to teach students how to follow the state-of-the-art research works within both academic and industrial context. Secondly, due to the fact that students are lacking practical experience on the design and development of large-scale systems, it is difficult for them to deeply understand the theory and rationality of software architecture. Accordingly, we need to teach students how to cope with practical problems based on the learnt theoretical work. This paper presents our exploration on designing theoretical and practical projects of software architecture course for postgraduate students during the last ten years. We aim to improve students' ability to investigate cutting-edge achievements and concerns in academia and industry, to understand, analyze, and design industrial software architecture based on open-source systems. Furthermore, we expect and encourage students' potential to become software architects. This paper details our project setup and discusses the feedbacks, and shows the benefits of our projects.**

Keywords—software architecture course, theoretical project, practical project, research ability, open-source systems

I. Introduction

Software architecture plays a key role as a bridge between requirements and implementation [1]. It models requirements to meet user needs and provides a guidance for code implementation. Therefore, software architecture is regarded as one of the core courses in computer science and software engineering. There has been increasingly more attention on the teaching method to cultivate students' ability to analyze and design software architecture, and to excavate their potential to become future software architects.

Teaching software architecture course faces two challenges. First, it is usually infeasible to update the knowledge written in textbooks with the rapidly developed practical progress. Accordingly, it is difficult for students to follow the state-of-the-art problems and solutions within both academic and industrial context. Secondly, due to the fact that students are lacking practical experience on the design and development of large-scale systems in different domains, it is difficult for them to deeply understand the theoretical principles and rationality of software architecture, and to cope with practical problems based on the learnt theoretical principles. Consequently, it is a tough teaching task to build students' ability to analyze and design software architecture. To overcome the above obstacles encountered in the course, we have explored to design appropriate student theoretical and practical projects since the course established in 2010 and achieved some positive results.

The objective of the designed projects is to improve students' ability to investigate cutting-edge achievements and concerns in academia and industry, to understand, analyze, and design industrial software architecture based on open-source systems. Furthermore, we expect and encourage students' potential to become software architects. This paper will detail the setup of the designed projects in our course, evaluates students' feedbacks, reports the effectiveness, and discusses possible future improvements. Therefore, we have designed a framework for conducting the theoretical and practical projects. The global design is depicted in Fig. 1, including the objectives, the project setups, the methods for evaluating the objectives, the guiding principles, and the implementation principles of our projects. We will explain and discuss these details in the following chapters.

The paper is organized as follows. Section II presents the objectives and design principles of our framework. Section III introduces the theoretical projects for investigating the state-of-the-art research topics in software architecture. Section IV presents the practical project for software architecture analysis based on open-source systems. Section V gives some concluding remarks and discusses future improvement.

II. Objectives and Design Principles of Projects

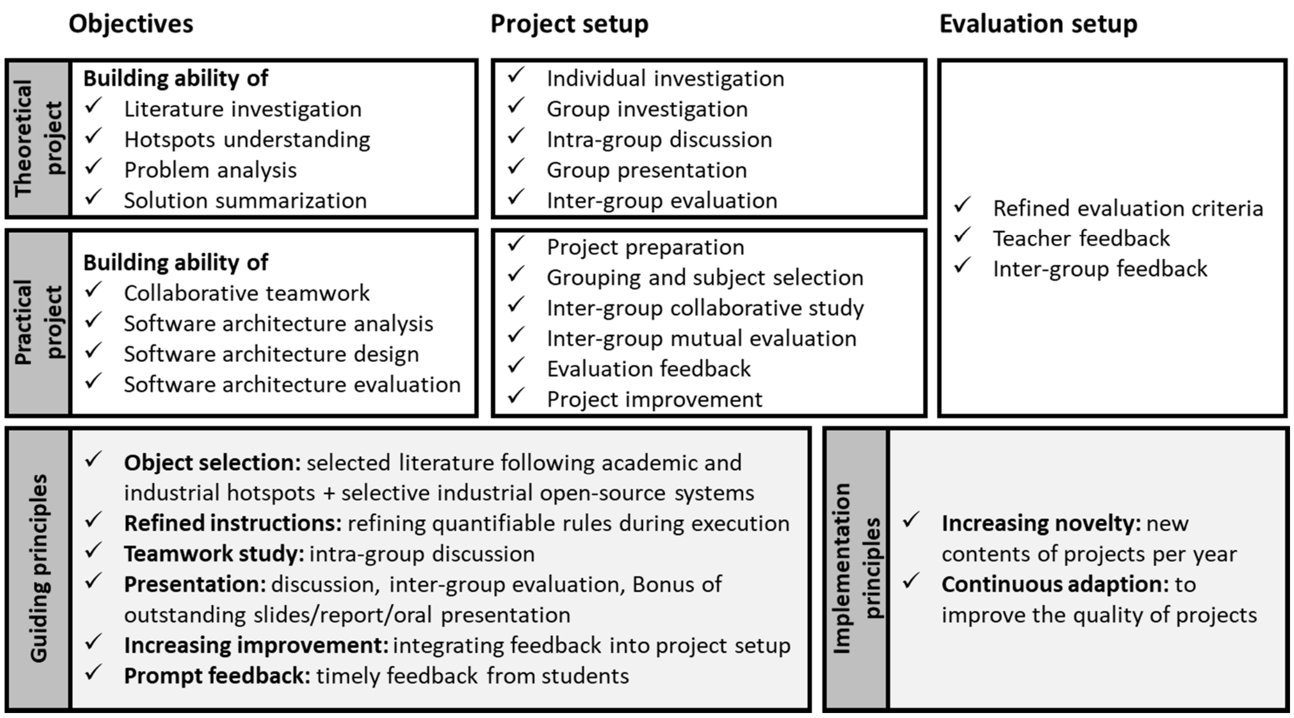
We have designed theoretical and practical projects separately in the course. In this chapter, we will illustrate the objectives, the guiding principles, the implementation principles of our projects framework given in Fig. 1. Then, briefly overview the setup of two projects.

A. *Objectives of projects*

We aim to improve students' ability to analyze and design software architecture on complex systems through theoretical and practical course projects. We have defined detailed objectives for the theoretical and practical projects separately.

For the theoretical project, we expect that students find the advanced achievements and concerns of software architecture in academia and industry through autonomous learning. The project is aimed at helping students to build the ability to investigate literatures, analyze problems, summarize existing solutions, and understand hotspots in the discipline. Students will be accustomed to investigating the published articles and technical reports, instead of only focusing on text books, to follow up cutting-edge research and application works.

* Ning Ge is the corresponding author. This work was supported by the National Natural Science Foundation of China Grant No. 61732019 and No. 61690202.



Objectives | **Project setup** | **Evaluation setup**

Theoretical project

Building ability of
- ✓ Literature investigation
- ✓ Hotspots understanding
- ✓ Problem analysis
- ✓ Solution summarization

- ✓ Individual investigation
- ✓ Group investigation
- ✓ Intra-group discussion
- ✓ Group presentation
- ✓ Inter-group evaluation

Practical project

Building ability of
- ✓ Collaborative teamwork
- ✓ Software architecture analysis
- ✓ Software architecture design
- ✓ Software architecture evaluation

- ✓ Project preparation
- ✓ Grouping and subject selection
- ✓ Inter-group collaborative study
- ✓ Inter-group mutual evaluation
- ✓ Evaluation feedback
- ✓ Project improvement

- ✓ Refined evaluation criteria
- ✓ Teacher feedback
- ✓ Inter-group feedback

Guiding principles
- ✓ **Object selection:** selected literature following academic and industrial hotspots + selective industrial open-source systems
- ✓ **Refined instructions:** refining quantifiable rules during execution
- ✓ **Teamwork study:** intra-group discussion
- ✓ **Presentation:** discussion, inter-group evaluation, Bonus of outstanding slides/report/oral presentation
- ✓ **Increasing improvement:** integrating feedback into project setup
- ✓ **Prompt feedback:** timely feedback from students

Implementation principles
- ✓ **Increasing novelty:** new contents of projects per year
- ✓ **Continuous adaption:** to improve the quality of projects

Fig. 1. Objectives and global design of course projects

For the practical project, we expect that students practice more on the industrial software systems, which are complex and reveal different views from stakeholders. The project is aimed at helping students to build the ability to work collaboratively, to analyze, design, and evaluate industrial software architecture. Through the exercise in the project, students will be able to analyze user requirements, design methods to bridge the gap between requirements and software implementation. More importantly, students will understand the design decisions and trade-off between different design alternatives.

B. *Guiding principles*

We have designed seven principles to guide the project setup and the evaluation setup based on the defined objectives. The details are illustrated as follows:

1) Object selection

The literatures in theoretical project and target systems in practical projects are selected according to two principles: the quality of literature/system and the students' interests.

In order to help students master the literature searching capability, we first demonstrate how to conduct a search of high quality literatures using academic search engine and university library resource. The teacher pre-defined a list of searching keywords. Some students are confused in front of large amount of searched papers. We provide some filter criteria such as the citation number, the topics of conference, the impact factor of journal, etc. It is important to guide students in evaluating papers and finding highly related ones.

For the target systems involved in the practical project, we have selected widely known industrial systems that represent different architecture styles. If students are interested in other systems, we encourage students to follow their interests. In order to guarantee the quality of self-selected systems, we guide their selection by defined clear criteria like the scale of systems (LOC, number of class/methods/functions), the source of systems (open-source), the maturity of systems (time of maintenance), the availability of documents, etc.

2) Conducting rules

In our projects, most questions are related to literature understanding and system analysis. The answers of these questions are usually open. We thus encourage students to freely explore and keep their divergent opinion. Nevertheless, in order to guarantee that students can stay in scope of projects and achieve the training goals, we have refined quantifiable instructions to conduct their work and supervise the project results.

3) Teamwork study

Most of the project works are designed for teamwork study. We define this principle for two reasons. First, students usually have different views on the same article or software system. We encourage them to discuss and exchange viewpoints. Second, the analysis and design of a software architecture involve many stakeholders, they have different concerns about the system, and software architecture embodies multiple design decisions. Our project simulates real-world system design process. It helps students understand the complexity of architecture design. Team members can discuss, collaborate, and reach consensus on the conflicts. Moreover, students can learn how to cooperate with others and improve the decision-making ability. We have noticed that the team size can impact the quality of projects, thus carefully limited number of students in a team.

4) Presentation and inter-group evaluation

Each group will give an oral presentation in front of the class to report the project results and their gains. This design is to let students practice the expression and communication capability. Other groups will evaluate their works by asking questions and proposing suggestions for future improvement. Students are encourage to deliver suggestions in a clear, respectful, constructive manner. This practice can enhance students' ability of problem analysis and critical thinking.

Incentive mechanisms are carefully designed to encourage students' participation. For instance, some students can get bonus points for their outstanding slides, reports, and oral presentation.

5) Increasing improvement

Students can incrementally improve their project results based on comments from teachers and classmates. We expect that students benefit from the project rather than achieving a project score. Student can evaluate their performance and improve the ability of reviewing others' work.

6) Prompt feedbacks from students

We collect students' feedbacks on the project setup and their project results to analyze the weaknesses and difficulties in the project, then optimize the design of projects.

C. Implementation principles

In terms of implementation, our design of course projects follows two principles. One is that new contents are yearly added in the projects to avoid duplication. The other is that the quality of projects is continuously improved to adapt up-to-date results in the domain of software architecture.

D. Project setup

According to the objectives and the design principles, we elaborately design two course projects: (1) theoretical project: Investigating state-of-the-art research topics (2) practical project: software architecture analysis based on open-source systems.

1) Theoretical project: investigating state-of-the-art research topics

It is usually difficult for the updating speed of knowledge written in textbooks to match the speed of conference and journal publications. This project is design to let students learn how to conduct literature search and get insight and foresight from highly related and high-quality publications.

2) Software architecture analysis based on open-source systems

Analyzing software architecture is one of the key abilities expected in the course, because students are lack of industrial experience and domain knowledge. Practicing analysis ability is a heavy task in our course and we have experimented on different projects since ten year. One important question is *what kind of systems should be chosen as target in the project.*

We used to select some small-scale software systems for students, but the results are not as good as expected. In fact, it is difficult to balance between the complexity and the feasibility. Many industrial software systems are large-scale and too difficult to analyze, while the toy systems are too small or simple to analyze. Since last year, we decided to take the challenge and let student analyze some large-scale open-source systems. Open source systems are popular and mature. We rely on the open-source systems for their well-designed architecture, the development styles, the maintenance by professional practitioners, as well as the code and document availability.

III. Investigating State-of-the-art Research Topics

Software architecture is an important research field of software engineering, it includes multiple research directions, such as software architecture design and analysis [2], [3], evolution analysis [4], consistency checking [5], [6] and recovery [8], [9], etc. Many researchers, teams and organizations are committed to research in these directions, and make state of art contributions.

To help students understand the state-of-the-art topics in software architecture, we have designed four project tasks and make several observations during the project improvement. We discuss our results as follows.

A. Task-1: Investigating assigned papers

We assign some high-quality academic papers in the scope of software architecture to students. Each student read a paper and answers a set of questions, including target problem, solution, novelty and contributions, experimental results, lessons learned, shortcomings, then formed a reading report in slides.

Most students learnt how to read an academic paper and can understand the papers. However, we found some shortcomings of this task from the students' feedback:

- Some students were not familiar with none of the given papers, they encountered some obstacles during reading and did not well understand the paper.
- Teachers need to check plagiarism, because a paper was selected by several students.
- Only a few students can share their reading report in class due to limited class hours.

To overcome these shortcomings in Task-1, we re-designed Task-2 in following year.

B. Task-2: Investigating self-selected papers

Students read papers in pairs and cooperated to form reading notes to avoid individual biases. The papers were self-selected from well-known journals and conferences. Each group were required to select different papers.

We find that many students have selected high-quality papers and output better reading report than those in Task-1. Besides, students learn how to balance different opinions with their teammates. Moreover, students have formed good habits of searching and reading papers, and obtained wider scope of knowledge compared to Task-1.

However, it is a pity that students cannot research in industry. Since then, we decided to design a project guiding them to learn the frontiers in both academia and industry.

C. Task-3: Recommending research teams or organizations in software architecture

Task-3 covered research works in both academia and industry. Students are asked to search for research groups and industrial companies in the field of software architecture. Then, they ranked the top ten teams or organizations and explained the ranking reasons in the report.

Some group recommend professional research groups (e.g. the SoftArch Laboratory at University of South California) and organizations (e.g. ArchSummit). The ranking reasons are various but reasonable, like popularity and contributions. All group presented their results in class and commented on other groups' report.

A bonus of this project task is that the recommended teams or organizations are regarded as the benchmark or direction for searching the latest techniques and concerns in other project tasks.

D. *Task-4: Investigating representative achievements of research teams or organizations*

After conduction Task-(1-3) for years, we summarized the experience and designed Task-4, as a combination of Task-(1-3). Task-4 is the current theoretical project in our course. This project is aimed at investigating hotspot research directions in software architecture, the excellent teams or organizations in academia or industry, and systematically sort out their frontiers including papers and software systems. We recommend some alternatives based on Task-3, while students can also investigate other topics in the scope of software architecture.

To develop students' research and learning ability, we specified some requirements which students should abide. First, each group should form an overview of the research direction, team or organization. Second, each achievement should be summarized in the report. For papers, they report solution, validation and experimental results. For software systems, they report function description, usage scenarios and application cases. Both papers and software systems should include target problems, novelty and contributions, lessons learnt and shortcomings.

As results, most groups followed with frontiers and concerns in academia and industry, and have shown deep understanding on the papers and software systems, not just summarize the achievements as notes by PowerPoint documents. For example, several groups discuss some topics of ArchSummit in the past five years, such as big data architecture, cloud computing architecture, microservice architecture and financial software architecture. Indeed, the project results are far from our preconceived effect.

E. *Summary*

With several years of practice, we continue to improve the learning state-of-art project and get significant feedbacks. In terms of students, some of them tell us that it is important to learn the latest techniques and concerns in industry for their future occupation. Furthermore, some students say that many theories are uneasy to understand, but with the help of reading related papers, they can grasp the important theories. Others say that they practice the methods learned in the papers and the software systems as tools to their own projects. Most of all, students improve their ability to research, analyze materials and learn the state-of-art. In terms of teachers, we learn more academic achievements from student assignments. Both students and teacher benefit from the project.

IV. Software Architecture Analysis Based on Open-Source Systems

A. *Task-5: Software architecture design*

Our initial objective is to develop students' skills of system design on a workable software system, so we set a project for designing a crowdsourcing software system based on swarm intelligence widely used in industry. Swarm Intelligence (SI) techniques are decentralized and self-organizing by defining and providing a viable solution for multi-objective optimization of (expensive) black-box functions [10].

However, the project failed to achieve a good results. On the one hand, most students had no experience in designing and developing large-scale systems and were lacking of domain knowledge of crowdsourcing systems. They had to spend much time on learning the basic knowledge, which results in less time to design and implement. And students cannot be engaged in design systems with different domains. On the other hand, the project is not suitable for the course with more than a hundred students.

Therefore, we turned to design a practical project for software architecture analysis. At first, we allow students to select systems they are interested in, including open-source and private systems. With several practices, we observed that open-source systems are more suitable for the project, because students are more familiar with these systems.

B. *Task-6: Software architecture analysis based on Open-source systems*

Students preferred to open source systems because of their good development styles, developing and maintaining by professional practitioners, as well as available codes. And a large number of students use open-source systems in their research work. After evaluation, we think open-source systems are the better choice than private systems for the project of software architecture analysis in the course.

Students formed a group and chose a target system to analyze. Each group analyzed not only the software code, but also software documents, In the process, students focused on the implementation of both functional and non-functional requirements, analyzed the basis and reasoned of designing such architecture.

We found many excellent project results in this task. They have several commonalities:

- explaining the requirements in the application scenarios
- analyzing software architecture styles or design patterns conforming to functional and non-functional requirements
- summarizing the advantages and disadvantages in such software architecture

These results show that the objectives of our projects are achieved, and the students have mastered the method to analyze the architecture of a large-scale software system.

C. *Task-7: Software architecture recovery*

Software architecture recovery is a frontier in academia and industry. It is a reverse engineering approach that aims at recovering viable architectural views of a software application [7], which plays an important role in software architecture consistency checking [6] and evolution analysis [4].

In this task, students not only need to analyze the requirements and design decisions as Task-6, but also need to recover the components of the software architecture and assign entities to corresponding components.

It is a good task for students to learn the frontiers and exercise their skills of software architecture analysis and design. We also can get inspirations for the future work of software architecture recovery from the task results.

D. *Summary*

Based on student feedbacks, we abandon the project of software architecture design for its inefficiency, and devote to researching how to cultivate students' ability based on open source systems. We illustrate our current project on software architecture analysis based on open-source systems, with Task-6 and Task-7. Both can help students apply the theoretical knowledge to the application on real systems, such as architectural styles, design patterns and decision. This project improved their ability of analyzing and designing a software architecture and relevant experiences in large-scale of systems as well.

V. Conclusion

Software architecture course is hard to teach because knowledge written in textbook is more oriented to theory. It is for students without much more design and implementation experience on industrial systems to learn practical knowledge from the textbooks. According to our 10-years teaching experience, designing good projects will help students to understand the research and application problems in the field of software architecture. After defining the objectives of the course projects, we designed a framework for the projects of software architecture course, including the guiding principles, implementation principles, project setup and evaluation setup for both theoretical and practical projects.

In this paper, we presented the evolving process of our theoretical and practical projects. One is for investigating the state-of-the-art research topics. The other is for analyzing software architecture based on open-source systems. Both students and teachers benefit from these course projects. For students, they do not only learn the cutting-edge research and concerns in software architecture, but also apply the learnt methods to their research works and have developed the ability of autonomous learning. They also improve their ability of software architecture analysis and design. For teachers, we have got a lot of helpful feedbacks from students to optimize the projects.

An interesting fact is that it seems the course projects are getting more and more difficult every year, but more and more students take software architecture course in our school. We expand the course capacity from 120 to 150 in 2019, and from 140 to 160 in 2020.

In our on-going work, we will design more projects oriented to industrial problems, and invite some software architects with design experience on large-scale systems to give talks on software architecture analysis and design. Moreover, we can cooperate with enterprises to design software architecture as projects, stakeholders will give more valuable guidance to students.

References

[1] D. Garlan, “Software architecture: a roadmap,” in 22nd International Conference on on Software Engineering, Future of Software Engineering Track, ICSE 2000, Limerick Ireland, June 4-11, 2000, A. Finkelstein, Ed. ACM, 2000, pp. 91–101.

[2] O. Sievi-Korte, I. Richardson, and S. Beecham, “Software architecture design in global software development: An empirical study,” J. Syst. Softw., vol. 158, 2019.

[3] Y. Cai, L. Xiao, R. Kazman, R. Mo, and Q. Feng, “Design rule spaces: A new model for representing and analyzing software architecture,” IEEE Trans. Software Eng., vol. 45, no. 7, pp. 657–682, 2019.

[4] D. M. Le, P. Behnamghader, J. Garcia, D. Link, A. Shahbazian, and N. Medvidovic, “An empirical study of architectural change in open-source software systems,” in 12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015, M. D. Penta, M. Pinzger, and R. Robbes, Eds. IEEE Computer Society, 2015, pp. 235–245.

[5] J. Buckley, S. Mooney, J. Rosik, and N. Ali, “JITTAC: a just-in-time tool for architectural consistency,” in 35th International Conference on Software Engineering, ICSE ’13, San Francisco, CA, USA, May 18-26, 2013, D. Notkin, B. H. C. Cheng, and K. Pohl, Eds. IEEE Computer Society, 2013, pp. 1291–1294.

[6] N. Ali, S. Baker, R. O’Crowley, S. Herold, and J. Buckley, “Architecture consistency: State of the practice, challenges and requirements,” Empirical Software Engineering, vol. 23, no. 1, pp. 224–258, 2018.

[7] S. Ducasse and D. Pollet, “Software architecture reconstruction: A process-oriented taxonomy,” IEEE Trans. Software Eng., vol. 35, no. 4, pp. 573–591, 2009.

[8] J. Garcia, I. Krka, C. Mattmann, and N. Medvidovic, “Obtaining ground-truth software architectures,” in 35th International Conference on Software Engineering, ICSE ’13, San Francisco, CA, USA, May 18-26, 2013, D. Notkin, B. H. C. Cheng, and K. Pohl, Eds. IEEE Computer Society, 2013, pp. 901–910.

[9] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidovic, and R. Kroeger, “Measuring the impact of code dependencies on software architecture recovery techniques,” IEEE Trans. Software Eng., vol. 44, no. 2, pp. 159–181, 2018.

[10] L. Peska, T. M. Tashu, and T. Horváth, “Swarm intelligence techniques in recommender systems - A review of recent research,” Swarm Evol. Comput., vol. 48, pp. 201–219, 2019.