Creative Education, 2015, 6, 211-214
Published Online February 2015 in SciRes. http://www.scirp.org/journal/ce
http://dx.doi.org/10.4236/ce.2015.62019

The Study on Case-Driven Methodology to Teach Software Engineering in Graduate Education

Juntao Gao, Wei Chen, Lingling Guo, Xiaozhe Yin, Zhibao Wang, Hongbo Zhou

School of Computer & Information Technology, Northeast Petroleum University, Daqing, China
Email: gjt@nepu.edu.cn

Received 22 January 2015; accepted 11 February 2015; published 13 February 2015

Abstract

This paper discusses the reform of teaching contents, teaching model, teaching method of software engineering courses, and assesses the effects of a series of reform measures and programs. It formed a case-driven methodology to teach software engineering. The teaching practice undertaken by Northeast Petroleum University has shown that the methodology provides graduate students with the opportunities to experience the realistic software engineering problems and environments, and then effectively raise their interest in learning.

Keywords

Software Engineering, Graduate Education, Case-Driven

1. Introduction

Software engineering teaching plays important roles in computer education of universities. It involves basic principles, methodologies, techniques and tools used in development of complex software systems. The teaching effect has important influence on the future work of students engaged in software development. However, it is not easy for graduate students to understand the knowledge of software engineering. The reason is twofold. First, most of the methods in software engineering are proposed to solve the problem of developing software system in large scale and they seem nonsense to the little prototype system developed in experiments. Second, graduate students have few opportunities to experience realistic case of complex software development. Therefore, they have no idea of the problem in complex software development.

In order to improve the effect of software engineering teaching, the case-driven methodology to teach software engineering is proposed in this paper. Case-driven teaching is a flexible model which exploits the basic

How to cite this paper: Gao, J. T., Chen, W., Guo, L. L., Yin, X. Z., Wang, Z. B., & Zhou, H. B. (2015). The Study on Case-Driven Methodology to Teach Software Engineering in Graduate Education. *Creative Education, 6,* 211-214.
http://dx.doi.org/10.4236/ce.2015.62019

capacity for students to learn from cases and the basic desire of teachers to tell cases that are indicative of their experiences (Li, 2010; Guo et al., 2010). The case-driven methodology to teach software engineering makes the students participate in a case that the students found interesting and where the telling of a case would be appreciated. Through participating in some typical cases, the students are demanded to apply principles, methods and tools to solve realistic problems (Yang & Liu, 2009; Teiniker et al., 2011). The teaching practice undertaken by Northeast Petroleum University has shown that the methodology can effectively enhance their interest in learning software engineering.

This paper includes the following contents: the next section introduces the principle to select and design cases. The third section specifies the process to teach software engineering in graduate education. The fourth section analyzes the teaching effects. At last, the conclusions are drawn.

2. The Principles of Designing Cases

Integrating theory and practice is emphasized in case-driven teaching, where the students are made to learn knowledge on their own initiative and trained to cultivate the ability to solve realistic problems. This is consistent to the goal of software engineering teaching. In addition, the quality of the cases has important impact on the interest of students. Therefore, the following five principles should be conformed during design of teaching cases.

Diversity Principle

There are widely different types of software systems, the software system development process used by the software development process and methods are different according to different types of software systems, so the software engineering teaching cases should not include only a single type system, but cover a variety of types of software. For example, during the phases of requirement analysis and system design, the cases of developing the management information system, real-time system, network software development should be included at least.

Interesting Principle

Interesting is critical to make the students to learn knowledge on their own initiative, so it is very important to select some interesting cases, such as the greedy snake game development, memos tool development, online teaching management system development, etc.

Complexity Principle

The teaching cases should have certain scale and complexity and need three to five students participates in the analysis and design. Otherwise, it is hard to let the student understand the effects of the principles, methods of software engineering.

Variability Principle

It is better to predefine some points of variability in one teaching cases, in order to simulate the realistic development processes which make developer have to deal with continuous change of user requirements. In this way, the students may understand the importance the scalability of software architecture when dealing with changes of requirements.

Completeness Principle

It is better to make the contents of the case cove the complete lifecycle of software systems, a variety of methods and technologies are integrated, to raise students' comprehensive ability to solve problems. However, because of the limit of time, it should not take a very long time to let the students to write programs, one case should cover at least requirements analysis, architectural design, detailed design, and another case is used to practice testing.

3. The Teaching Processes

Effective teaching is based on proper teaching processes, which can be divided into steps.

1) Preparation

Before the formal discussion begins, the materials of cases are hand out to each student. It is guarantee that the students have enough time to learn the cases and retrieve necessary literatures. In this way, the students can initially form their own opinions and solutions to the problems in the cases. This stage is indispensable in the whole teaching process. If the students do not fully understand the contents of the cases, the teaching effect is going to be affected.

2) Construction of Organization

Construct some groups collaborate to solve the problems. The size of each group should not be too big, usually within three to five personal advisable. The students are allowed to select their partner freely.

3) Requirements Analysis

In the teacher's guidance, the group members are inspired to discuss freely, think actively, and make the case to discuss closely around the key requirements. In the process of the discussion the group members must be active to speak, fully showed his opinions to the problems of the cases. Other group member should pay attention to the opinions. Through comparing their team opinions, try to make comprehensive decisions. The result of this stage is requirements model and specification. Finally, the requirement specifications are checked among groups in order to simulate the process of requirements checking.

4) Software Design

According to the requirements specification of their own team, apply the technology of design to draw a software solution to the requirements, then Select another requirements specification from other team, proceed to design. The result of this stage is software design model. Finally the requirement specifications are checked among groups in order to simulate the process of requirements checking.

5) Coding

Because of the limit of course time, it is impossible to implement all of the modules designed in the last stage. Therefore, some typical modules selected to practice coding.

6) Testing

The purpose of this stage is to simulate the test procedure. First, arbitrarily select the codes from other team. Then design the test cases, write test script and implement them. The result of this stage is the report of software testing.

7) Summarization

After the completion of the case, the teacher first make the summarization according to the performance of each group, analyze some good advice and unique insights and puts forward the deficiencies and problem analysis thorough and comprehensive degree comments, so as to improve the quality of the case discussion, and write case analysis report. Through the written report, students' abilities to presentation are raised.

4. Teaching Effects

In order to assess the effect of the case-driven teaching, sixty students from four specialties (computer science and technology, electronic science and technology, information management, education technology) are divided into two groups. First group was taught using the methodology described in this paper, the other group was taught using traditional methodology. The contents of course are the same. The comparison of teaching effects is depicted as **Table 1**. The ratio of excellence and good of first group reaches 50%, while the ration of excellence and good of second group reaches 21%. In addition, the opinions of the first group are more comprehensive than the second group. In the experiments, the first group is more active and few solutions are similar.

5. Conclusion

The teaching effect of software engineering has important influence on the future work of students engaged in software developments. It takes a long way to raise the quality of software engineering teaching. The practice of case-driven methodology to teach software engineering reaches good effect. In order to satisfy the requirements of technology advance, we are going to continue to explore the new teaching methods and enrich the base of teaching cases, try to cultivate more and more creative talents.

Table 1. The comparison of teaching effect between two groups.

Ratios	First Group	Second Group
Excellence	15.16%	5.22%
Good	28.53%	14.11%
Medium	36.56%	53.3%
Pass	16.5%	20.55%
Fail	3.25%	6.82%

Acknowledgements

This work is sponsored by the Engineering Professional Degree Graduate Education Projects of China (2014-JY-040).

References

Guo, L. L., Man, Y., & Yu, F. (2010). Exploration and Practice on Case-Driven Research-Based Teaching Model. *2nd International Conference on Education Technology and Computer*, 1209-1211.

Li, W. (2010). To Explore Case-Driven teaching Mode for Mechanical CG Course. *2nd International Workshop on Education Technology and Computer Science*, 284-287.

Teiniker, E., Paar, S., & Lind, R. (2011). A Practical Software Engineering Course with Distributed Teams. *14th International Conference on Interactive Collaborative Learning*, 195-201. http://dx.doi.org/10.1109/ICL.2011.6059575

Yang, C., & Liu, Y. (2009). Teaching Reform and Practice on the Software Engineering Course. *International Conference on Information Science and Engineering*, 3470-3473.

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.