Journal of Contemporary Educational Research, 2024, Volume 8, Issue 10
http://ojs.bbwpublisher.com/index.php/JCER
ISSN Online: 2208-8474
ISSN Print: 2208-8466

Construction of the Curriculum System of Software Engineering Based on the Agile Development Method

Ziyi Wang*

Hebei University, Baoding 071000, Hebei Province, China

***Corresponding author:** Ziyi Wang, dujiaoshou1972@163.com

Abstract: Under the background of "new engineering" construction, software engineering teaching pays more attention to cultivating students' engineering practice and innovation ability. In view of the inconsistency between development and demand design, team division of labor, difficult measurement of individual contribution, single assessment method, and other problems in traditional practice teaching, this paper proposes that under the guidance of agile development methods, software engineering courses should adopt Scrum framework to organize course project practice, use agile collaboration platform to implement individual work, follow up experiment progress, and ensure effective project advancement. The statistical data of curriculum "diversity" assessment show that there is an obvious improvement effect on students' software engineering ability and quality.

Keywords: Agile development method; Software engineering; Practical teaching; Curriculum system; Experimental project

Online publication: October 23, 2024

1. Introduction

Since its inception, the School of Software has been guided by market demand and aimed at cultivating high-level engineering and practical software talents with international competitiveness [1]. Therefore, training in software engineering practical skills has always been an important goal for software engineering undergraduates. Only through the engineering practice of software systems can students deeply understand the software engineering knowledge they have learned and apply the corresponding knowledge to solve complex system development problems [2]. Software engineering course is a subject with strong theory and practice and has become the core course of computer science and technology, software engineering, and other majors. The main content of the software engineering course includes three aspects: process, method, and tools in software engineering. The teaching of the course aims at cultivating students' software engineering quality and enabling them to possess certain software project management skills, software design ability,

and project implementation ability. On the one hand, most of the content of this course is the summary of previous experience, which has an important guiding role for the implementation of software engineering; on the other hand, students with little experience in engineering projects feel that the concepts are boring and the content is difficult to understand [3]. Therefore, the vast majority of colleges and universities have set up corresponding practical courses, and strive to let students deepen their understanding of the course content through practical practice. With the development of social and economic forms, the construction of new engineering puts more emphasis on the continuous cultivation of practical skills, emphasizing the integration of production and education, the combination of science and education, and the collaborative education of school-enterprise cooperation, so as to cultivate talents with excellent engineering skills who can actively adapt to the development of new technology, industry, and economy. Although school-enterprise cooperation and practical skill training are not novel concepts, how to innovate software engineering practice teaching and adapt it to the development of the new economy is a problem that needs to be considered.

2. Teaching status of software engineering course

The teaching methods for software engineering practice courses are generally divided into two categories: course experiments and practical training or course design. The course often has limited experimental hours and scattered scheduling, allowing only basic practice with software tools. Therefore, institutions with the necessary resources tend to offer extended class hours with concentrated schedules, enabling the use of large project cases for practical teaching. To improve teaching quality, front-line educators have introduced a tutorial system, along with various practical teaching methods such as software maintenance-oriented and multi-agent approaches. In practical training, students are typically guided through the waterfall development model, from feasibility analysis, requirements analysis, and design (both high-level and detailed) to coding and testing stages. While this classical model allows students to experience every phase of software engineering, it also has some drawbacks. First, the progress of each stage heavily depends on the outcome of the previous one, and students, lacking project experience, may make mistakes at various points. These mistakes often go unnoticed until later stages, requiring rework and diminishing student motivation. Second, without a real client, requirements analysis tends to be hypothetical, which limits its practical relevance. Third, the waterfall model generates a significant amount of documentation, increasing workload and slowing project progress, making it difficult for students to complete a full project within the available time. Lastly, within student groups, there is often an uneven distribution of tasks, with too much reliance on a few top-performing students. To address these issues in software engineering practical training, it is essential to improve development efficiency, adopt methods suitable for small teams, and fully engage all students to boost enthusiasm and participation.

3. Reform ideas for software engineering experiment teaching

3.1. Agile software development methods

Agile software development is a methodology that prioritizes people, focuses on delivering products over producing extensive documentation, and is adaptable to change. Unlike the traditional waterfall model, which is more rigid and document-heavy, Agile is considered a lightweight approach. It emphasizes individuals and interactions over processes and tools, functional software over comprehensive documentation, collaboration with customers over formal negotiations, and responding to change over strictly following a predefined plan.

In Agile development, software projects are broken down into multiple sub-projects, with each sub-project delivering tested, integrated, and operational results. Since the formation of the Agile Alliance in 2001, a variety of Agile methodologies have emerged, including notable examples such as Extreme Programming (XP), Scrum, and Feature-Driven Development (FDD). Both domestically and internationally, an increasing number of practical projects have successfully adopted Agile development methods [4].

3.2. Scrum project development process

Scrum is an iterative and incremental Agile software development process in which each development cycle is called a Sprint. During each Sprint, the development team commits to completing a set of tasks, known as the backlog. The workflow of the Scrum method is illustrated in **Figure 1**. The process includes a Sprint planning meeting, daily Scrum meetings, a Sprint review meeting, and a Sprint retrospective meeting, which together form the review and adjustment phases of the Scrum approach [5].

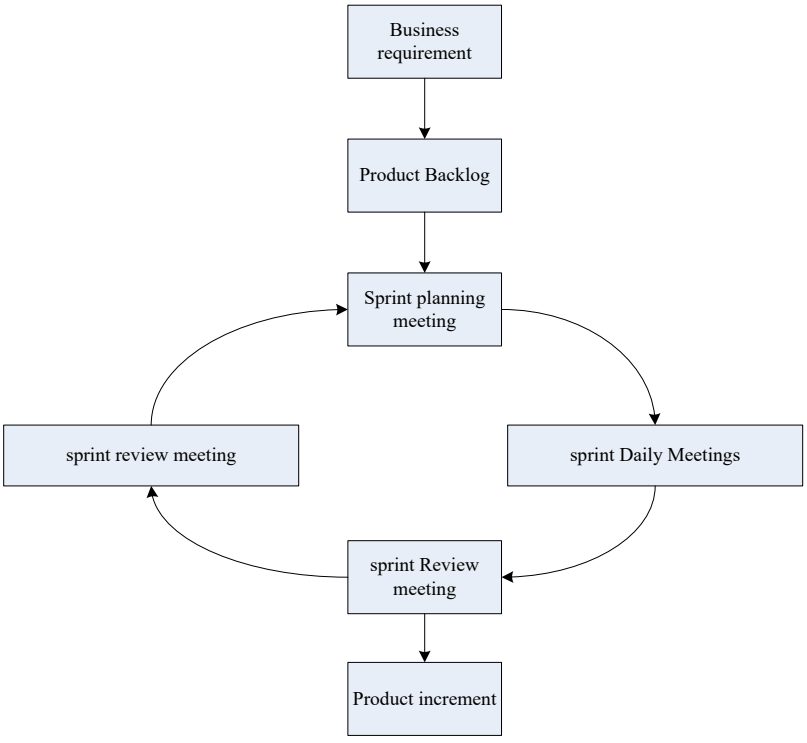

Figure 1. Scrum project development process

(1) Project planning and product backlog creation: Similar to traditional development methods, Scrum requires upfront project planning. However, in Scrum, the development team collaborates to create and prioritize a Product Backlog. This backlog includes an estimated workload for each task. During project execution, the Product Backlog remains a dynamic list of business and technical features to be developed.

(2) Sprint planning meeting: Each Sprint cycle begins with a planning meeting attended by the product owner, ScrumMaster (similar to a project manager), and the development team. In this meeting, the team decides which items from the Product Backlog will be implemented in the upcoming Sprint. After selecting these items, they are refined into the Sprint Backlog. Team members choose tasks based on their expertise and interests, with any remaining coordination handled by the ScrumMaster.

(3) Daily Scrum meetings: The daily Scrum meeting is attended by all team members and is kept brief. The participants remain standing during the meeting, which is why it is often called a "stand-up" meeting. Each team member answers three key questions: What did you accomplish yesterday? What will you do today? Are there any obstacles to completing your tasks? These meetings are concise to avoid time wastage.

(4) Sprint review meeting: At the end of each Sprint, a Sprint Review meeting is held where the team presents the completed work to the product owner and other stakeholders. The team also discusses the next steps based on the results of the Sprint.

4. Construction of software engineering curriculum system based on agile development method

Since Tencent Agile Product Development (TAPD) is an agile development management platform designed for enterprise organizations, it naturally lacks certain features required for teaching management. To successfully integrate TAPD into practical courses and ensure it fully supports the development of various student teams, making it an effective tool for software engineering practice courses, the following reforms need to be implemented.

4.1. Integrating the existing course management collaboration platform

To achieve project management in TAPD, one must first create a project and add members. In a software engineering practice course, each student team is a project team. If the team has many members, adding them manually one by one is a tedious task. In addition, the formation of the practice team is a dynamic process: students submit a team application, and the course teacher reviews and decides to allow the team to be formed or fine-tune the team members, which is done within the course group's existing course management collaboration platform. The introduction of TAPD in the curriculum cannot increase the workload of students, so that they can create project teams in both platforms. Therefore, first of all, it is necessary to integrate the course management collaboration platform and TAPD, and directly import the team information created from the existing course management collaboration platform into TAPD to automatically generate team project organization. Students then log on to TAPD for research and development management. Before the practice starts, the course team must investigate the API provided by TAPD and contact the TAPD research and development team of Tencent to finally realize the integration work.

4.2. Training before using the Agile software development platform

Using the platform to support the development process can improve the efficiency of development, but the mastery of the platform function itself takes time. If students learn the use of TAPD at the beginning of the practice, it will greatly affect the effect of the practice, so students should be trained in the use of the TAPD platform in advance. This work is mainly carried out from three aspects: (1) Introducing TAPD briefly when teaching agile development in the software process management of the pre-course so that students can have a perceptual understanding of TAPD and its support for agile development process; (2) Developing TAPD training courseware suitable for practical needs, and releasing it to students in advance for self-study; (3) Before the practical course begins, the training on "TAPD functions commonly used in practice" is conducted again, and other available functions are briefly introduced. The above three aspects of training will pave the way for the use of TAPD in practice.

4.3. Clarifying the focus of software engineering courses and new requirements for practice

The TAPD covers the whole process of software development, including many functions, however, the software engineering practice teaching is concentrated in two weeks, which has limited time to complete. If all the functions of TAPD are used, students will spend a lot of time learning to use the platform in team practice, which will affect the quality of practice. Therefore, this practice focuses on TAPD's management process of agile requirements.

First of all, demand management is the first step in the software development process, and the quality of demand development affects the software quality in the later period, so the importance of demand management is self-evident. At present, the school has no special courses related to demand management, this practice aims to strengthen students' knowledge and understanding of this link.

Secondly, in real software development, requirement changes happen frequently, and will cause a series of changes in the development process, and the requirement change is an important issue that the team needs to deal with in the actual development of enterprises. In the past practice teaching, the demand was set at the beginning of practice, which was relatively stable, and students lacked the experience to cope with the change of demand. After the introduction of TAPD, the course practice is designed to increase the process of sudden demand changes from customers in the second iteration, so that the student team can experience the impact of demand changes and corresponding demand management.

4.4. Facilitating the software engineering course development process

A training room is set up for students who participate in software engineering training. In the training room, the different teams are separated by partitions, and each team is equipped with a server and six personal computers.

There are three main roles involved in the Scrum process: product owner, ScrumMaster, and development team. Before the practical training in software engineering, students should be allowed to form teams freely. The number of each team should be limited to 5–6 people, and a student with a certain management skill should be recommended as the ScrumMaster. The product owner is a teacher, and teaching assistants, graduate students, or senior students can be invited to participate. They have the right as the customer's representative to make any request for the product.

After the formal start of the practical training, teachers participated in the preliminary project plan of each team and determined the product Backlog, so that students had a clear understanding of the tasks, workload, and priorities of the products. The duration of a Sprint can be flexibly set according to the training duration of the students, but to ensure that all teams follow the same Sprint development cycle; generally, for six weeks of training, every seven days can be set as a Sprint cycle.

At the next Sprint planning meeting, the product owner, ScrumMaster, and the development team are all involved. The teacher explains the priority of each Backlog one by one, and the team selects the Backlog that can be completed within this Sprint cycle, breaks the Backlog into task lists, and estimates the workload for each task. Students in the development team can claim relevant tasks according to their own interests and expertise.

The daily Scrum meeting must take place at the same time and place every business day. Teachers do not participate but can sit in. Since it is a standing meeting, students can disperse to all corners of the training room or even outside to avoid interference.

On a daily basis, all students are assigned tasks and present their results at a review meeting at the end of a Sprint cycle, so students have to work hard for their assigned tasks. At the end of each day, students are required to perform integration and regression tests on all the code that has been checked into the project. This self-management approach significantly boosts efficiency and serves as a constructive method to motivate students without resorting to punishment. Less motivated students, observing the dedication and effort of their teammates, are naturally encouraged to contribute more actively to the team's progress.

4.5. Reforming the assessment method of coursework submission

The assessment of software engineering practice includes both development process assessment and final product assessment. In the past, process assessment relied on offline content such as assignment documents and physical task boards submitted by student teams. The introduction of Tencent's TAPD requires a corresponding reform of course assignment submission and assessment methods. Since this practice focused on TAPD's management process of agile requirements, the previous offline requirements user story and iteration plan documents were completed on TAPD accordingly, so that teachers could directly review the requirements planning of each team on the platform. After the requirements changed in the second iteration, the team also completed the corresponding requirements management process directly on TAPD, and the teacher reviewed the updates of the team online.

5. Implementation effect of software engineering curriculum system reconstruction

In order to assess the effect of the practical reform, a questionnaire survey was conducted after the course to evaluate students' satisfaction with the practical course. In addition to the survey on the practice itself, the 2019 survey also focused on the satisfaction survey on the use of TAPD, including a total of four dimensions. Each question in the questionnaire contains five options (very satisfied, satisfied, average, dissatisfied, very dissatisfied). The feedback statistics of students' satisfaction (including satisfied and very satisfied) are shown in **Table 1**. Based on **Table 1**, students' satisfaction with practice arrangement, review, and guidance is satisfactory, and students also highly recognize that practice methods can improve practical skills and enhance their understanding and mastery of software engineering knowledge.

Table 1. Software engineering practice satisfaction

Survey dimension	Investigation item	Satisfaction (%)
Practical organization	Whether the practical requirements are clear	98
	Whether the practice guidance and review are reasonable	98
	Whether the practice teaching arrangement is reasonable	90
Practical method	Whether the teaching method is helpful in improving students' practical skills	100
	Whether the teaching approach contributes to students' understanding of Agile development	98
Agile development platform support	Whether the managing requirements are helpful	82
	Whether they are satisfied with the introduction of Agile development methods to support the curriculum practice	75
Practical effect	Overall recognition of the curriculum	96

6. Conclusion

In a word, Scrum, a typical representative of Agile software development method, is applied to the practice teaching process of software engineering courses, which can overcome some defects of traditional teaching methods. It not only avoids aborting the project due to frequent rework caused by poor design but also provides practical requirements and guidance for the project as a customer representative in the course. It allows teachers to keep track of each student's workload and assess students more fairly. The agile way of self-management can stimulate students' motivation, allow students to experience the whole process of software engineering implementation, enhance students' project management and development abilities, achieve the training goal, and achieve a good teaching effect.

Disclosure statement

The author declares no conflict of interest.

References

[1] Bolloju N, 2022, Software Engineering Course Restructured to Support Agile Software Development Projects. Proceedings of the 15th Annual ACM India Compute Conference, 5(3): 106–109.

[2] Jali N, Bujang Masli A, Cheah WS, 2017, The Adoption of Agile Software Methodology with Team Software Process (TSPI) Practices in the Software Engineering Undergraduate Course. Journal of IT in Asia, 28(2): 49–52.

[3] Rastogi A, Jain S, 2023, Software Engineering: Agile Software Development. International Journal of Advanced Research in Science, Communication and Technology, 16(2): 455–457.

[4] Martin A, Anslow C, Johnson D, 2017, Teaching Agile Methods to Software Engineering Professionals: 10 Years, 1000 Release Plans, International Conference on Agile Software Development, Springer, Cham, 32(4): 89–90.

[5] Ilyes E, 2021, Curricula and Methods on Teaching Different Aspects of Agile Software Development. Central-European Journal of New Technologies in Research, Education and Practice, 6(1): 74.