Exploration of Teaching Reform in Software Engineering Experimental Courses Integrating CAD Software Development Technology

Weiguo Li*, Wenrui Li, Wenjie Xiao
School of Information Engineering, Nanjing Xiaozhuang University, Nanjing, China
**Corresponding Author*

Abstract: **Addressing the problems that the content of software engineering experimental courses is traditionally disconnected from actual industrial scenarios and that students' practical abilities are difficult to improve, this paper proposes a teaching reform plan that deeply integrates CAD software development technology with software engineering experimental courses. The article analyzes the existing problems in current software engineering experimental courses and clarifies the goal of integrating CAD development technology into experimental content to enhance students' system analysis and system design capabilities. Specific teaching reform ideas are proposed, and the experimental course content is designed to be divided into two parts: "Foundation" and "Engineering Practice Advancement". This design can both consolidate theoretical knowledge and enhance engineering practice capabilities. Finally, the effectiveness of the teaching implementation is analyzed.**

Keywords: CAD Software; Software Engineering; Experimental Course; Teaching Reform

1. Introduction

Industrial software, as the core support for intelligent manufacturing and industrial digital transformation, has been deeply embedded in various social production activities such as industrial production and engineering construction, becoming a strategic area for national key core technology research [1]. In 2021, a People's Daily commentator article clearly pointed out the need to gather strength to carry out original and leading scientific and technological research in industrial software and resolutely win the battle for key core technologies [2]. With the rapid development of the industrial software industry, the shortage of industrial software talent in China continues to expand. According to data from the "Key Software Talent Demand Forecast Report," the gap for industrial software talent in China will reach 120,000 by 2025, making it one of the areas with the highest talent urgency [3].

The Software Engineering major serves as the core base for cultivating software talent, yet its experimental courses have long focused on Web application development as the primary practical direction. However, the rapid development and popularization of current AI technology have significantly simplified the process of conventional application development and even replaced some basic application development tasks, leading to a reduction in job demand for programmers in traditional Web application development fields. In contrast, the development of industrial software is entirely different. As the core support for industrial digitalization and intelligent development, the R&D demand for industrial software continues to grow alongside the continuous upgrading of the industrial sector, providing more job opportunities for programmers that possess both technical depth and industrial value. Software engineering experimental course teaching needs to change in response to current changes in talent demand. This paper explores how to integrate CAD development technology into "Software Engineering" experimental courses and reconstruct experimental content to make teaching more aligned with current social job requirements, thereby improving students' engineering practice and social adaptability.

Scholars such as Tu et al. have proposed teaching reforms for professional comprehensive design experiments guided by the cultivation of "Excellent Engineers," emphasizing the precise docking of practical teaching with industry needs [4]. Scholar Tang Yun and others have

also pointed out a software engineering teaching reform path that promotes teaching through practice, providing ideas for curriculum system optimization [5,6]. Against this background, drawing on the experience of constructing a domestic CAD industrial software curriculum system[3], integrating industrial software technology into software engineering experimental course reform—allowing students to learn, practice, and create in real industrial production scenarios—has become a key path for bridging university talent cultivation and industrial demand.

2. Analysis of the Current Status of Software Engineering Experimental Courses

Traditional software engineering experimental courses have many shortcomings in the talent cultivation process and do not meet the talent requirements in the field of new-generation information technology:

2.1 Practical Directions are Disconnected from Core Industrial Needs, Lacking Industrial Scenario System Analysis

Traditional experimental courses mostly focus on Web application development, using mature open-source frameworks to implement upper-level business logic. This type of practice does not interface with real industrial production scenarios such as factory production and engineering design, directly leading to the absence of system analysis training based on industrial needs. Students lack training in modeling system object models, functional models, and behavioral models in industrial scenarios, resulting in a structural gap between talent cultivation and job requirements in the field of new-generation information technology [7].

2.2 The Experimental System Lacks Engineering and Collaborative Training under Industrial Scenarios

Traditional experiments are mostly completed by individuals as small projects, lacking the team collaboration, module division, and full-process product design training required for large-scale industrial R&D. Furthermore, they do not simulate collaborative R&D scenarios across departments and roles in industrial production. Students are unable to experience the complete industrial R&D process from requirements analysis and architectural design to code implementation and testing/delivery. Consequently, engineering thinking and collaborative ability cultivation are insufficient, making it difficult to adapt to job collaboration modes in social production [8].

2.3 Teaching Resources are Disconnected from Real Industrial Production Scenarios

Traditional experimental cases are mostly simulated projects, such as a "Web-based E-commerce Platform," lacking real industrial prototype support. After completing the experiments, students find it difficult to gain a sense of achievement from "applying what they have learned". At the same time, experimental platforms do not interface with the technical standards of industrial enterprises, resulting in students' practical results being unable to adapt to the application needs of the industrial world. This creates a clear separation between education and production practice.

3. Software Engineering Experimental Course Goals and Reform Ideas

3.1 Course Reform Objectives

To respond to the national industrial software talent cultivation strategy, this course reform is based on software engineering theory and integrates CAD software development technology into software engineering experimental content. By focusing on CAD software development technology engineering training, it encourages students to deeply understand the theoretical principles of system analysis and design through application development while improving their engineering practice capabilities. The specific objectives of the course are as follows:

Goal 1: Consolidate the technical foundation and strengthen system analysis modeling capabilities. This program aims to enable students to gain a deeper understanding and application of the core concepts of object-oriented analysis, building upon their proficiency in industrial software technologies such as C++ programming and graphics algorithms. Students will be able to use UML tools to model and analyze geometric entities in CAD software, organize the system's business logic to form functional models, describe interaction processes to establish behavioral models, and achieve a deep integration of theoretical knowledge and industrial practice.

Goal 2: Emphasize practical transformation and enhance system design and development capabilities.

This program immerses students in a simulated industrial software development scenario, allowing them to participate in the complete industrial software development process. Based on software design theory, students complete module partitioning, interface definition, and algorithm design. Utilizing open-source code resources or free plugin development platforms, they transform their designs into runnable software deliverables. Through training, students become able to design system architectures using data flow-oriented design methods; design algorithm flowcharts for each functional module of the system; and measure and evaluate algorithm complexity.

Goal 3: Strengthen independent awareness of domestic industrial software, cultivate team collaboration capabilities, and establish national confidence.

Focusing on the development of domestically produced industrial software, this program aims to help students understand the critical technological bottlenecks hindering the development of domestically produced industrial software during the system analysis and design process. It guides students to combine system design innovation with the need for domestic substitution, fostering a technological awareness of independent innovation and domestic alternatives. Students develop in teams using tools such as GitHub and SVN, enhancing their teamwork and communication skills.

3.2 Teaching Reform Ideas

This course reform draws on the teaching reform concepts for engineering application-oriented software talent cultivation, combined with the technical characteristics of industrial software R&D. It constructs a teaching idea of "Solidifying Theoretical Foundations - Industrial Demand Driven - Layered Foundation and Application Experiments - Collaborative Practice - Ideological Education". This approach closely follows the theme of "strengthening system analysis and design skills through industrial software practice" and decomposes capability cultivation goals into specific experimental contents for each teaching link to ensure that teaching objectives can be successfully achieved through the completion of experimental content. Table 1 below summarizes the teaching reform approach from three aspects: key points of teaching reform, core teaching content, and implementation methods.

Table 1. Ideas for Teaching Reform of Software Engineering Experimental Courses Empowered by Industrial Software Development Technology

Teaching Reform Key Point	Teaching Core Content	Implementation Method
Solidify Theoretical Foundations	Core knowledge of software engineering theory	Theoretical explanation in the Introduction to Software Engineering course
Layered Foundation and Application Experiments	Experiments are divided into two parts: consolidating theoretical foundations and advanced engineering practice development	The basic experiment part covers software engineering theory projects such as system analysis modeling, system design, and algorithm design; the advanced experiments allow students to analyze object models and functional modules in existing CAD systems to intuitively understand the application of software engineering knowledge in commercial software development
Industrial Demand Driven	Implement simple graphical system functions	Using the application of open-source software OpenCasCade in industrial scenarios as a prototype, introduce real enterprise demand scenarios as experimental materials
Collaborative Practice	Team collaboration to complete enterprise scenario software module R&D	Adopt an enterprise-style group division mode, simulate enterprise R&D processes, and introduce Gitee project management tools to enhance version iteration
Ideological Education	Ideological education on independent innovation of domestic industrial software	Integrate "bottleneck" technology research cases of industrial software in national industries to understand the progress of domestic software and strengthen patriotic feelings

Specific implementation key points are explained as follows:

Theoretical Reserves and Case Guidance: Students complete basic courses such as C language, data structures, C++ programming, and graphic algorithms in their first and second years of undergraduate study, consolidating programming skills through summer courses. In the third year, they master core theories of system analysis and design modeling through the "Introduction to Software Engineering" course. Using cases such as "Experimental Teaching

Management System" and "Warehouse Management Subsystem," students master the use of basic tools such as class diagrams, use case diagrams, and state transition diagrams, building a solid theoretical foundation for development practice in experimental courses.

Industrial Requirement Import: Decompose core tasks of requirements analysis. Combining commercial CAD software including open-source software OpenCasCade and Rhino CAD, introduce real cases such as "3D Freeform Surface Modeling System". Guide students to use object-oriented analysis methods to analyze the system's object model, draw class diagrams, clarify relationships between classes, extract core business points, define graphic drawing functions, implement simple graphic editing functions, and write requirement specifications that meet standards.

Layered Experiments: Experiments are divided into two parts: basic experiments and engineering practice advancement experiments. Basic experiments cover system analysis, module design, and key algorithm design to ensure students master core software engineering technologies. Advanced experiments encourage students to tackle technical points in CAD software development, allowing students to "learn by doing" and deepen their understanding and application of knowledge through practice.

Engineering Practice: Conduct software module R&D under industrial demand scenarios in groups, simulating the R&D process of enterprises in real industrial production. Invite enterprise mentors to provide joint guidance, and through testing feedback and problem correction, allow students to improve their engineering capabilities in collaborative practice [9].

Ideological Education: Introduce the progress of independent development of geometric engines for domestic CAD software and the technical difficulties and characteristics of CAX software development enterprises in various industries, achieving the organic integration of course teaching and ideological education [10].

4. Software Engineering Experimental Course Content and Teaching Effectiveness

4.1 Experimental Course Content Design

The software engineering experimental course is combined with the theoretical course and offered in the third year of undergraduate study. The theoretical course accounts for 48 credit hours, while the experimental course accounts for 64 credit hours. Theory is first taught in a classroom setting, followed by experiments in the laboratory.

The experimental content system consists of two parts: "Basic Experiments" and "Engineering Practice Advancement Experiments", as shown in Figure 1. The "Basic Experiment" part serves the mastery and applied understanding of software engineering theory. The "Engineering Practice Advancement Experiment" emphasizes training students' engineering development skills. After completing basic experiments, students further expand their engineering development skills and master basic technologies for CAD software development.

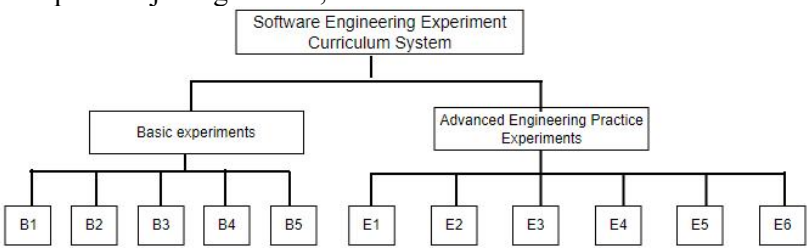

Figure 1. System Structure of the Software Engineering Experiment Course (See Table 2 for Details of the Specific Experiments Represented by the Letter Codes)

Since CAD software development technology is complex and involves a wide range of disciplinary knowledge, this experimental content is designed only around consolidating and improving students' system analysis and design capabilities. By making full use of existing open-source resources and commercial software secondary development platforms, students can utilize computer science knowledge to implement relatively simple CAD software functions within limited experimental hours.

The specific experimental content is arranged as shown in Table 2 below. Modules starting with the letter B belong to the basic experimental section, while modules starting with the letter E belong to the advanced engineering practice section. Modules E1 and E2 emphasize the development of students' system development

skills; Module E3 focuses on developing system analysis capabilities; Module E4 focuses on developing system design capabilities; Module E5 emphasizes ideological and political education; and Module E6 improves comprehensive practical skills. To save teaching costs, the commercial CAD platforms we selected are all free trial versions or open-source systems. For example, the Rhino 6 trial version provides a free plug-in development mode, and OpenCasCade technology is free and open-source software.

Table 2. Software Engineering Experimental Course Experiment Numbers, Names, and Contents

Number	Experiment Content Name	Content Description
B1	Topic Selection	Determine the project topic, preliminary organize the requirement statement, and draw the system business process with informal diagrams
B2	Structured Analysis	Structured analysis method: use data flow diagrams for the functional model, ER diagrams for the data model, and state transition diagrams for the behavior model
B3	Object-Oriented Analysis	Object-oriented analysis method: use use case diagrams for the functional model, class diagrams for the data model, and state transition diagrams for each class
B4	General Design	Design the system structure, divide functional modules, and define interfaces
B5	Detailed Design	Design algorithm flowcharts for key functions, and design the database and user interface
E1	Familiarize with Development Environment	Set up the VS Studio 2022 environment, implement a univariate polynomial root-finding function, and use simple dialogs for input and display
E2	DLL Dynamic Library Component Development	Learn how to use DLL dynamic library components to expand system functions through a simple game architecture case
E3	Study Commercial CAD Software Object Models	Analyze geometric classes in Rhino SDK and OpenCasCade, compare their differences, and deeply understand object-oriented modeling. Design a class diagram for a concise 3D geometric modeling system
E4	Study Commercial CAD Software System Structure	Use OpenCasCade as an example to analyze its dynamic library structure; build a simple graphics program using underlying DLLs to understand system structure design
E5	Ideological Education	Analyze the importance and difficulty of independent development of geometric engines in CAD software; introduce the achievements of domestic software like ZWSOFT and CrownCAD
E6	Simple CAD Function Implementation	Define a specific graphical function and implement it via a plugin on the Rhino platform or directly on the OpenCasCade framework

The experiments listed in Table 2 are designed to closely align with the learning objectives, with each experiment specifically designed to support a particular learning objective. Table 3 below lists the learning objectives and the corresponding experiment numbers. The learning objectives are achieved by completing all the experiments.

Table 3. Teaching Reform Goals and Supporting Experimental Modules

Teaching Goal	Supporting Experiment Content Number
Goal 1: Consolidate technical foundation, strengthen analysis modeling	B1,B2,B3,E3,E6
Goal 2: Emphasize practical transformation, enhance system design	B4,B5,E1,E2,E4,E6
Goal 3: Domestic software awareness, teamwork, national confidence	E5,E6

4.2 Experimental Course Teaching Effect

We implemented this teaching reform plan in the software engineering experimental teaching of the 2023 Computer Science and Technology class (Year 2) at the School of Information Engineering, Nanjing Xiaozhuang University. The class had 46 students, and all of them completed all experimental content. Grades were calculated using a weighted average of each experiment, with an average score of 85, a 100% pass rate, and a 21% excellence rate. Good teaching results were obtained. Some students developed software tools close to practical use,

as shown in the Figure 2 to Figure 5:

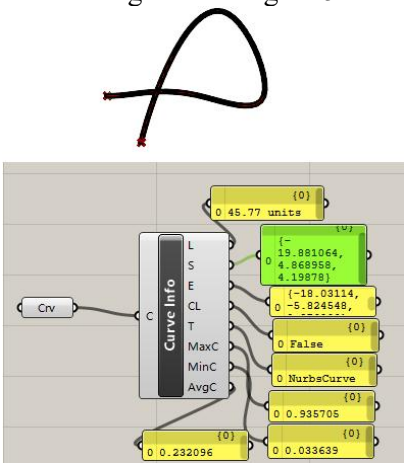

Figure 2. Curve Data Analysis Component

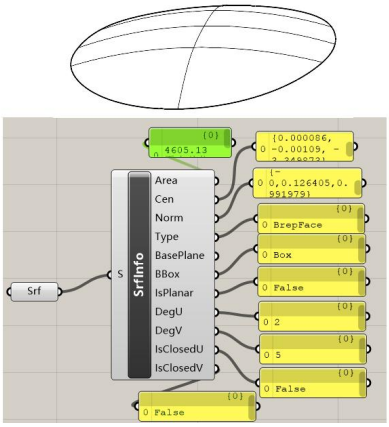

Figure 3. Surface Data Analysis Component

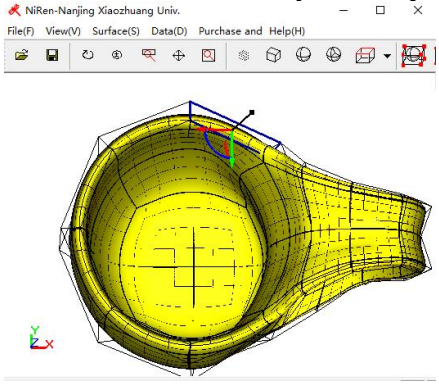

Figure 4. Edit the Coordinate Frame of the Surface

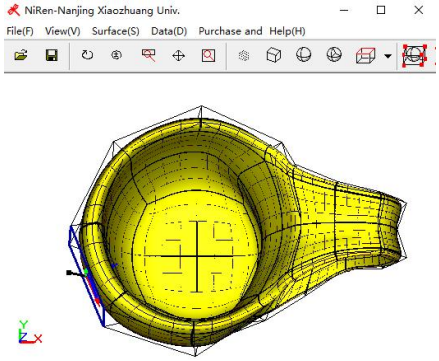

Figure 5. Updated Surface after Editing

Xie's group developed a geometric solid property analysis component in Rhino (Grasshopper) using the C# language. This component analyzes and reports various property data for Breps, freeform surfaces, freeform curves, and lines. Figure 2 shows the curve data analysis component. After the user selects a curve, its various property data are reported. Figure 3 shows the surface data analysis component. After the user selects a surface, its various property data are reported. Liu's group implemented an editing frame function in our self-developed freeform surface modeling software "NiRen". This function allows users to pick up a geometric solid, display an editing frame, and drag the arrow lines of the frame to change the shape of the surface. Figure 4 shows the surface effect before editing, and Figure 5 shows the surface effect after editing. These teaching results demonstrate that students effectively improved their engineering application development capabilities by completing the experiments.

5. Further Thinking on Integrating CAD Development Technology into Software Engineering Experimental Courses

The construction of software engineering experimental courses should always focus on mastering software engineering knowledge. Integrating CAD software development technology is only an auxiliary means to help students understand and apply knowledge. Since CAD software development focuses on specific engineering application backgrounds, this "integrated" teaching method can help students be inspired in the process of engineering practice and deepen their understanding of theoretical knowledge. Therefore, it is an effective teaching method.

Since the knowledge graph of CAD software is very complex and experimental hours are limited, further consideration is needed in the implementation process for students with varying basic abilities, designing experiments of different difficulty levels. At the same time, we should consider creating enterprise cooperation opportunities, contacting domestic enterprises like ZWSOFT to allow students to grow in a real industrial system development environment. Additionally, it is necessary to consider selecting teachers to enter enterprises for CAD software development training to improve teaching capabilities.

References

[1] Guo Gang, Lu Jinping, Dou Junhao, Wang Chong, CHen Yihong, et al. Introduction to the Development of China's Industrial Software Industry Status and Opportunities. Software Guide 2022, (10): 26-30.

[2] Commentator. Resolutely win the battle for key core technologies. People's Daily, 2021-05-31(1).

[3] Yang Qinghong, Xie Mingxuan, Zhuang Weiguang, et al. Construction of experimental course system for domestic CAD industrial software design. Computer Education, 2025, (06): 45-50.

[4] Tu Junling, Lin Ziyuan, Yuan Wuzhi, et al. Reform and Practice of Comprehensive and Designing Experiments of Chemical Engineering for Excellent Engineers Education. The Theory and Practice of Innovation and Entrepreneurship, 2023(4): 56-60.

[5] Tang Yun, Tong Kaiyu. Through Practice to Promote the Teaching, Discussion of Software Engineering Education Reform Program. Experimental Science and Technology, 2011 (Supplement 1): 198-200, 213.

[6] Li Cuixia, Zhang Shuyan. Teaching Reform Practice of Introduction to Engineering Applied Software Talents. Education and Teaching Forum, 2019(8): 121-123.

[7] Liu Wei, Zhao Yu, Ge Guiping. Research on undergraduate information talent cultivation mode from the perspective of engineering education. Industry and Information Technology Education, 2025(5): 24-28.

[8] Zhang Li. Continuously improve the software engineering talent cultivation system. Computer Education, 2025(9): 1-5.

[9] Wang Fei, Li Xinyue, Li Ziyi. Talent demand forecast and suggestions in the key software field. Software and Integrated Circuits, 2025(10): 52-60.

[10] Zhang Lishuo, Xu Yuanyin, Zhang Lu, Wu Hao, Wen Xiaoshuangang. Research on the Promotion of Industrial Software Talents' Practical Ability Enhancement by "Dual-qualification" Teachers. Journal of Xinxiang University, 2025(9): 65-68.

