ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

Design and Implementation of Experimental Teaching in Software Engineering Economics

Jie Huang[1,2,*], Jing Zhang[1,2], Nuo Chen[2]

[1]National Pilot Center of Experimental Teaching in Computer and Information Technology, Tongji University, Shanghai, 201804, China
[2]School of Software Engineering, Tongji University, Shanghai, 201804, China
huangjie@tongji.edu.cn
**Corresponding author*

Abstract: *In order to meet the requirements for cultivating emerging engineering talents and enhance the teaching quality in the field of software engineering, this paper addresses the challenges faced in teaching and learning of software engineering economics by introducing a reform in the course's teaching methodology, focusing on experimental teaching. Based on the Institute of Electronics and Electrical Engineers (IEEE) Software Engineering Body of Knowledge (SWEBOK) and engineering education practical teaching requirements, the problem-driven experimental teaching method is introduced, and a series of "virtual + real" experimental projects are designed and arranged; relying on industry-university cooperative education resources, course experiments are constructed on Ali cloud, and is free for students to use. Practice has shown that the experimental teaching reform of software engineering economics has effectively promoted the combination of "teaching" and "learning" in the course, enabling students to exercise and improve the economic thinking and innovation ability of software engineering in practice.*

Keywords: *Software engineering, Software engineering economics, Experimental teaching design, Cloud platform, Teaching reform*

1. Introduction

Software engineering originated from the engineering practices of pioneering scholars who applied engineering concepts and methods to reform the design and development of software systems [1-2]. As an engineering activity, software engineers often need to make decisions during the process of software design, development, and maintenance. For example, software requirement decisions agreed upon by both software providers and clients, decisions regarding technical debt and software refactoring, as well as task allocation decisions under limited resource constraints. Introducing principles and methods from economics to address engineering economic decision-making problems became inevitable, leading to the emergence of software engineering economics as a core area of study, primarily focused on analyzing the cost-effectiveness of software engineering activities. In the SWEBOK third edition published by IEEE in 2014, and the SWEBOK fourth edition (draft) in 2023, software engineering economics is recognized as an independent knowledge area within SWEBOK and has become an essential component of software engineering education [3-4].

The challenges in teaching and learning software engineering economics arise from the interdisciplinary nature of the subject matter. On the knowledge supply side of teaching, most domestic universities lack faculty members with both interdisciplinary knowledge and practical experience in software engineering, economics, and management discipline [5]. On the knowledge demand side of student learning, many undergraduate students in software engineering programs lack the prerequisite knowledge required for this course, such as basic concepts of market, economics, engineering management, and business regulations.

The traditional teaching model, which primarily relies on theoretical lectures, not only fails to engage students effectively but also leads to low teaching and learning efficiency for both teachers and students. This mismatch results in an inadequate alignment between teaching and learning. Additionally, software engineering economics, as an emerging course in software engineering discipline, has a curriculum that not only embodies the characteristics of multidisciplinarity and interdisciplinary fusion but also evolves rapidly due to the iterative and evolving nature of knowledge derived from the information industry's

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

practices. Relying solely on textbooks and limited in-class hours for theoretical teaching makes it challenging to elucidate the course's key concepts to university students who lack work experience and fails to provide students with the latest industry best achievements for practices. This mismatch between talent cultivation in this field [6] and the needs of the industry necessitates a reform in teaching methods and approaches to address these issues [7].

Drawing inspiration from the "Emerging Engineering Education" philosophy of "student-centered, outcome-oriented, continuous improvement" [8], and considering the course content and the economic decision-making processes in software engineering, a series of progressively structured, modular in-class experimental projects are designed to complement theoretical instruction. Through active participation in these experiments, students gain firsthand experience and gradually develop an understanding of economic thinking in software engineering and practical industry methods. Additionally, in collaboration with Alibaba Cloud through the university-industry cooperation in education, a blended software engineering economics virtual simulation experiment teaching platform is established and deployed on Alibaba Cloud, and students can use it for free [9].

2. "Blended" Experimental Teaching Plan

Undergraduate engineering education emphasizes the cultivation of students' abilities to solve complex engineering problems [10-11]. The content of software engineering economics encompasses interdisciplinary knowledge elements from software engineering, computer science and technology, management, and economics. Software project requirements involve multiple stakeholders, and their interests may conflict, requiring the establishment of appropriate abstract models and various methods to resolve conflicts or solve problems. The solutions are highly comprehensive and involve multiple interrelated subproblems. The software engineering economics experimental teaching plan adopts a "blended" approach. The term "real" refers to experimental projects originating from the real course projects that students choose independently, such as national innovation projects, national competitions, or research and development projects supervised by professors. These projects typically possess real business requirements, real functions, real software activities, engineering management aspects, and source code. The term "virtual" refers to elements within the experimental project requirements related to stakeholder conflicts of interest, software evolution, software market dynamics (such as product supply and demand relationships, pricing strategies, and profit analysis), and regulations. These kind of elements require students to make assumptions or make predictions during the experiments.

The foundation for implementing "blended" experimental teaching is the construction of a "serial, modular, problem-oriented" series of experimental projects (Figure 1). These experimental projects begin with basic validation experiments that even students with no prior knowledge can undertake. They then progress to intermediate design experiments and advanced comprehensive experiments. All of these experiments require students to establish appropriate abstract technical-economic models to address the issues faced in the experimental projects. In particular, students are expected to demonstrate creativity in problem-solving during the comprehensive experiments. The modular experimental projects range from simple to intermediate to advanced, from individual problems to complex integrated problems. This progressive approach not only lowers the entry barrier for the course but also allows motivated students to have a comprehensive practical experience in this course's main exercises.

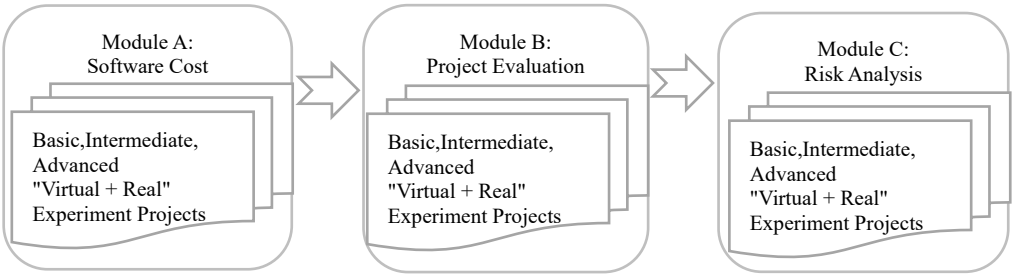

Figure 1: Example Structure of Modular "Blended" Series of Experimental Projects

The Software Engineering Economics virtual simulation experiment teaching system cloud platform allows teachers and students to access the experimental environment through a web browser and the internet. Students can conduct experiments at any time and place without restrictions. Additionally, teachers can make real-time adjustments to add, remove, or update experimental projects or content based on teaching and student feedback during the course. The virtual simulation experiment teaching cloud

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

platform is now an integral part of both offline and online MOOC components, forming a three-dimensional teaching environment. The "blended," "virtual replacing real," and "mutually complementary" experimental teaching system meets the requirements for teaching a large volume of knowledge within limited class hours.

The design of "blended" experimental projects emphasizes modularity, with three types of experiments within the same knowledge module: basic, intermediate, and advanced experiments, corresponding to validation, design, and comprehensive experiments, respectively. Basic experiments are compulsory for all students. Intermediate experiments require all students to complete at least one experiment within a specific knowledge module, while advanced experiments are optional and offered to students who have the capacity and time to complete them (Table 1).

Table 1: Representative Experimental Projects in the Course

Knowledge Module	Experimental Project Name	Nature of Experiment	Difficulty Level
Supply and Demand	Supply and Demand Experiment of Carbon Emission Rights	Validation	Basic
Software Size Measurement	Software Size Measurement Experiment - IFPUG, ISO Standard	Design	Intermediate
	Software Size Measurement Experiment - NESMA, ISO Standard	Design	Intermediate
	Software Size Measurement Experiment - COSMIC, ISO Standard	Design	Intermediate
	Software Size Measurement Experiment - MARK II, ISO Standard	Design	Intermediate
Software Cost	Software Cost Estimation Experiment - GB/T36964-2018 Software Engineering Software Development Cost Measurement National Standard	Design	Intermediate
	Software Testing Cost Estimation Experiment - GB/T 32911-2016 Software Testing Cost Measurement National Standard	Design	Intermediate
Economic Indicators	Basic Software Project Economic Indicator Calculation Experiment - Net Present Value (NPV) & Internal Rate of Return (IRR) Indicators	Validation	Basic
	Basic Software Project Economic Indicator Calculation Experiment - Dynamic Investment Payback Period Indicator	Validation	Basic
Software Project Evaluation	Software Project Financial Evaluation Experiment - Single Scenario Evaluation Model	Comprehensive	Intermediate
	Software Project Financial Evaluation Experiment - Multi-Scenario Evaluation Model	Design	Advanced
Software Economic Lifecycle	Software Economic Lifecycle Calculation Experiment	Design	Intermediate
Engineering Management and Control	Software Process Monitoring and Control - Earned Value Analysis (EVA)	Design	Advanced
Risk Analysis and Assessment	Risk Analysis and Assessment Experiment - Break-even Method	Design	Basic
	Risk Analysis and Assessment Experiment - Monte Carlo Method	Design	Intermediate
	Risk Analysis and Assessment Experiment - Sensitivity Analysis Method	Design	Intermediate
	Risk Analysis and Assessment Experiment - Decision Tree Method	Design	Intermediate
Uncertainty Analysis and Assessment	Uncertainty Analysis and Assessment Experiments	Comprehensive	Advanced

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

3. Virtual Simulation Experiment Teaching Cloud Platform

The Software Engineering Economics Virtual Simulation Experiment Teaching System (referred to as the system) is deployed on the Alibaba Cloud platform. The platform is designed with an emphasis on modularity, scalability, and openness, representing a collaborative achievement in cooperative education between the university, industry, and enterprise. This system comprises five subsystems: experiment management, performance management, user management, announcement management, and log recording. It is capable of providing blended virtual and real experiment teaching services for representative course experiments. Students can access the platform website using web browsers on tablets, laptops, or desktop computers to participate in experiment projects.

This system seamlessly integrates data interfaces and records the entire experiment process. It offers self-service access to experiment manuals, announcements, and a Q&A module, empowering students to conduct experiments independently. The system assists in course-assisted teaching by guiding students through the experiment process, from principles to operations. It provides a graphical interface, allows browsing or downloading of experiment manuals, offers interactive experiment reports, and visualizes experiment results, aiding students in learning and reviewing relevant course content. Experiment reports are graded electronically on the system, resulting in paperless work. Graded experiment scores are entered into the performance management module, making it more convenient for both teachers and students to access and analyze their grades.

This system adopts a hierarchical architecture (Figure 2) with the goal of providing a web-based virtual experiment environment for users to conduct various experiments and simulations. Users can interact with the backend through the interface of the system and obtain experiment results and feedback.

The interface of the system boundary is built as a Single Page Application (SPA) using the Vue framework, providing user interfaces, experiment controls, and result displays. The backend of the system, built on the Spring Boot framework, processes frontend requests, executes experiment logic, and interacts with the database, among other functions.

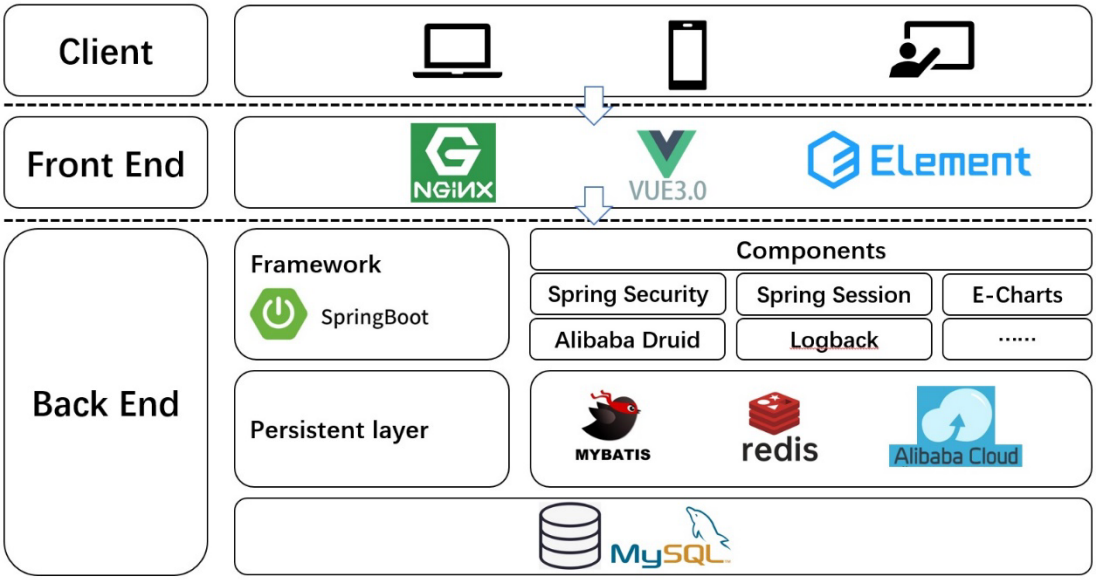

Figure 2: Architecture of the Virtual Simulation Experiment System

The system's frontend components include page components, router components and HTTP request components. Page components include pages for the experiment list, experiment details, experiment control, result display, and more, implemented using Vue components. Router components manage navigation and routing between different pages using Vue Router. HTTP request components use Vue's HTTP library or third-party libraries to communicate with the backend API.

The system's backend components consist of controllers, services, data access layer and security layer. Controllers receive frontend requests, process request parameters, call the relevant services, and manage input and output data. Services layer implements specific experiment logic, data processing, and business rules, handling requests from the controllers. Data Access Layer interacts with the database, performing data read and write operations. Security Layer manages user authentication and authorization, ensuring

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

the security of experiment data and user information. It can also integrate with third-party services such as authentication or storage services as needed.

Dependency relationship between the frontend and backend of the system. Frontend depends on backend: The frontend makes HTTP requests to call APIs provided by the backend to retrieve experiment data, send control commands, and more. Backend depends on frontend: The backend provides APIs for the frontend to call, receiving and processing frontend requests.

System deployment architecture. Frontend code can be deployed on a static resource server or CDN to allow users to access it through web browsers. Backend code is deployed on an application server, which receives and processes frontend requests via the HTTP protocol.

4. Blended Virtual and Real Experiment Demonstration

Software size measurement is the foundation of software cost estimation and is one of the key concepts in this course. In the online course, we explain the prevailing software size measurement method used in the industry, which is the Function Point method. For offline experiments, students are free to choose either national standards (GB) or international standards (ISO) for Function Points as their experiment projects in the software measurement module. It should be emphasized that the software projects are selected by student groups themselves to ensure the authenticity of software requirements, functionalities, and code. In the following, we will demonstrate the experiment process using the "Software Size Measurement Experiment - IFPUG ISO Standard" case.

The first step is to identify Data Function Points and Transaction Function Points. Data Functions are categorized into Internal Logical Files (ILF) and External Interface Files (EIF). Transaction Functions refer to operations such as External Inputs (EI), External Outputs (EO), and External Queries (EQ), including activities like data retrieval and output. Student groups identify these five metrics based on the real software system architecture and the detailed design documents of their respective course projects.

The second step is to measure the Internal Logical Files (ILF). This involves counting the ILF quantity in the course project's application program, which are data blocks or control information that can be confirmed to be maintained within the application program and logically related.

The third step is to measure the External Interface Files (EIF). This step entails counting the EIF quantity in the course project's application program, which are data blocks or control information that can be confirmed to be referenced by the tested application program but maintained within other application programs.

The fourth step involves calculating the complexity of ILF and EIF. Student groups apply the IFPUG method for measuring functional size, based on national standards [12], to identify the complexity of ILF and EIF in the second step and the third step. This is done by referencing the detailed design documents of the course project and assigning complexity values (simple, average, or complex) according to the parameters in Table 2.

Table 2: ILF and EIF Data Complexity Identification Table

Record Element Type (RET)	Data Element Type (DET)		
	1 ~ 19	20 ~ 50	>50
1	Simple	Simple	Average
2 ~ 4	Simple	Average	Complex
>5	Average	Complex	Complex

The fifth step is to measure External Inputs (EI). This involves measuring the basic processes in the course project's application program that handle data or control information from outside the system boundary. Count the number of EI in the experiment case.

The sixth step is to measure External Outputs (EO). This involves measuring the basic processes in the course project's application program that provide data or control information to outside the system boundary. Count the number of EO in the experiment case.

The seventh step is to measure External Inquiries (EQ). This involves measuring the basic processes in the course project's application program that provide data or control information queries to outside the system boundary. Count the number of EQ in the experiment case.

The eighth step involves calculating the complexity of EI, EO, and EQ. Using the IFPUG method,

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

identify the complexity of EI, EO, and EQ in steps five to seven, referencing the detailed design documents of the course project. Assign complexity values (simple, average, or complex) according to the parameters in Tables 3 and 4.

Table 3: EI Complexity Determination Table

Number of Referenced File Types (FTR)	Data Element Types (DET)		
	1 ~ 4	5 ~ 15	>15
0 ~ 1	Simple	Simple	Average
2	Simple	Average	Complex
>2	Average	Complex	Complex

Table 4: EO and EQ Complexity Determination Table

Number of Referenced File Types (FTR)	Data Element Types (DET)		
	1 ~ 5	6 ~ 19	>19
0 ~ 1	Simple	Simple	Average
2 ~ 3	Simple	Average	Complex
>3	Average	Complex	Complex

The ninth step is to calculate Unadjusted Function Points (UFP). According to the mapping between component complexity levels and function point numbers as defined in the IFPUG method, and with reference to the detailed design documents of the course project, calculate the Unadjusted Function Point count (UFP). An example calculation is detailed in Table 5.

Table 5: Sample Calculation of Unadjusted Function Points

Component	Complexity									Unadjusted Function Point CountC+F+I
	Simple			Average			Complex			
	Count	Weight	Points	Count	Weight	Points	Count	Weight	Points	
	A	B	C = A*B	D	E	F = D*E	G	H	I = G*H	
EI	3	3	9	-	4	-	-	6	-	9
EO	2	4	8	-	5	-	-	7	-	8
EQ	3	3	9	-	4	-	-	6	-	9
ILF	2	7	14	1	10	10	-	15	-	24
EIF	1	5	5	1	7	7	-	10	-	12
Total Unadjusted Function Point Count										62

The tenth step is to calculate the Adjusted Function Points. Taking into account the non-functional factors of the application software measured by the IFPUG method, the relevant system characteristic adjustment factors are selected and summed to obtain the Value Adjustment Factor (VAF) for the experiment case. The IFPUG method's function point calculation formula is then used to calculate the Function Point value for the experiment case. This value represents the software size of a specific course project designed and developed by a certain student team measured by IFPUG ISO standard.

In this experiment, teachers should explain to students that the concept of function points in software size measurement is different from the concept of function points in software requirements. The two are not comparable in terms of numerical values.

5. Conclusion

The reform of software engineering economics experiment teaching has fundamentally solved the problem of the mismatch between "teaching" and "learning" that has plagued both teachers and students. It has accumulated experience for the construction of the emerging engineering education of software engineering and engineering education practice. The "combined virtual plus real experiments" teaching model based on the course project selected freely by students, which not only stimulates students' motivation and interest in independent learning and exploration but also cultivates their hands-on abilities. The problem-oriented experimental project design provides students with the innovative ability to apply

ISSN 2522-6398 Vol. 6, Issue 25: 107-113, DOI: 10.25236/FER.2023.062519

what they have learned in practice. The experiment teaching cloud platform provides the possibility of flexible allocation and sustainable improvement of modular experimental projects according to teaching needs. The modular experimental project design meets the needs of students with different learning objectives. The reform of software engineering economics experiment teaching provides a new idea and an example for experimental teaching in software engineering discipline.

Acknowledgments

This work was supported by the following funding projects: "Construction and Teaching Reform of the Hybrid Course Software Engineering Management and Economics", the first batch of industry-university collaboration and co-education projects in 2020, approved by Higher Education Division of the Ministry of Education, 2021, No. 202002001028. "Virtual Simulation Laboratory Course in Software Engineering Management and Economics" , the 17th phase of special fund projects for experiment teaching reform at Tongji University, 2022, No. 7.

References

[1] ISO/IEC/IEEE 24765:2017(E), ISO/IEC/IEEE International Standard: Systems and Software Engineering — Vocabulary [EB/OL]. [August 31, 2023]. https://standards.ieee.org/findstds/standard/24765-2017.html.

[2] Steve Tockey. Return on Software: Maximizing the Return on Your Software Investment [M]. New York: Addison-Wesley, 2004.

[3] IEEE. Software Engineering Body of Knowledge (SWEBOK) [EB/OL]. [August 31, 2023]. https://www.computer.org/education/bodies-of-knowledge/software-engineering.

[4] IEEE. SWEBOK Evolution [EB/OL]. August 31, 2023. https://www.computer.org/volunteering/boards-and-committees/professional-educational-activities/software-engineering-committee/swebok-evolution.

[5] Song Fagen, Liu Jia. Application and Analysis of Practice in the Teaching of "Software Engineering Economics" [J]. Software, 2020, 41(07): 288-291[CHN].

[6] Yu Tianzuo, Jiang Jianwei, Ren Rui, et al. Continuous Quality Improvement Based on Engineering Education Accreditation Standards — A Case Study of the National Exemplary Software Engineering Major at Z University [J]. Tsinghua University Education Research, 2015, 36(06): 104-111[CHN].

[7] Huang Jie, Tang Jianfeng, Yan Haizhou. Reform and Practice of Experimental Teaching in Software Engineering Economics [J]. Higher Education Journal, 2020(21): 135-137[CHN].

[8] Ministry of Education, Ministry of Industry and Information Technology, Chinese Academy of Engineering. Opinions on Accelerating the Construction and Development of the New Engineering Education and Implementing the Excellent Engineer Education and Training Program 2.0 [EB/OL]. September 17, 2018. https://www.gov.cn/zhengce/zhengceku/2018-12/31/content_5443530.htm[CHN].

[9] Higher Education Division of the Ministry of Education. Notice on Publishing the List of Projects Approved for Collaborative Education between Industry and Universities in 2020. Letter from the Higher Education Division [2021] No. 3 [EB/OL]. March 4, 2021. http://www.moe.gov.cn/s78/A08/tongzhi/202103/t20210324_522389.html [CHN].

[10] Jiang Zongli. Undergraduate Engineering Education: Focusing on Cultivating Students' Ability to Solve Complex Engineering Problems [J]. Chinese University Teaching, 2016(11): 27-30[CHN].

[11] Lin Jian. Construction of New Engineering Majors with Interdisciplinary Fusion [J]. Research on Higher Engineering Education, 2018(01): 1-22[CHN].

[12] GB/T 42449-2023. Systems and Software Engineering — Functional Size Measurement — IFPUG Method [S]. Beijing: State Administration for Market Regulation, Standardization Administration of China, 2023[CHN].