Asia Pacific Science Press

Journal of Educational Theory and Practice
ISSN(O):3005-9313
ISSN(P):3005-9305
Vol. 3 No.1 (2026)

Research on Experimental Teaching Reform of Vue.js Course Based on CDIO and Knowledge Graphs

Jie Wu[1]*, Yan Liang[2]

1.School of Computer and Software Engineering, University of Science and Technology Liaoning, Anshan, 114051, China

2.School of Applied Technology, University of Science and Technology Liaoning, Anshan, 114051, China

Corresponding author: Jie Wu, wujie@ustl.edu.cn

Abstract: With the rapid advancement of information technology and the increasing emphasis on engineering education certification, traditional experimental teaching models exhibit significant limitations in fostering students' abilities to address complex engineering problems. Using the Vue.js front-end development course as a case study, this paper proposes an experimental teaching reform model based on the integration of CDIO and knowledge graphs. The proposed model first constructs a structured, visualized, and interconnected Vue.js knowledge graph to organize course content. Guided by the CDIO (Conceive, Design, Implement, Operate) engineering education framework, the model then restructures the experimental teaching process around authentic projects, dividing it into four stages aligned with six experimental modules. This approach achieves a deeper integration of theoretical instruction and engineering practice. Teaching practice demonstrates that the reform model significantly enhances students' systematic knowledge construction, practical engineering skills, and capacity for innovation and collaboration, thereby offering an effective pathway for preparing learners to meet the evolving demands of the front-end technology ecosystem and the software industry.

Keywords: CDIO; Knowledge Graph; Experimental Teaching Reform; Vue.js; Engineering Education

Published: Feb 24, 2026

DOI: https://doi.org/10.62177/jetp.v3i1.1066

1.Introduction

In the era of the digital economy and intelligent industrial transformation, web front-end development technology has emerged as an essential core competency in the field of software engineering. As a progressive JavaScript framework renowned for its accessibility, Vue.js has achieved widespread industrial adoption due to its strengths in reactive data binding and component-based architecture, rendering it a critical skill for front-end engineers[1]. Nevertheless, contemporary university-level Vue.js experimental teaching often suffers from challenges such as fragmented content delivery, insufficient realism in project simulations, and a misalignment between skill development and industry needs. These issues hinder compliance with engineering education standards, such as those outlined in the Chinese Engineering Education Professional Accreditation, particularly in fostering students' ability to address complex engineering problems.

The CDIO (Conceive-Design-Implement-Operate) engineering education model advocates a project-based learning framework that cultivates students' integrated competencies across the complete project lifecycle. This pedagogical approach has achieved widespread adoption and demonstrated significant efficacy in global engineering education[2]. Meanwhile,

knowledge graph technology provides sophisticated capabilities for knowledge organization, semantic reasoning, and visualization, thereby addressing the pervasive challenge of "knowledge silos" and facilitating the construction of cohesive knowledge systems[3]. Drawing on these foundations, this paper examines the pedagogical application of the prevailing front-end framework Vue.js by integrating the CDIO engineering education philosophy with knowledge graph technology. It introduces an integrated experimental teaching framework that combines the CDIO engineering philosophy with knowledge graph technology. This dual-driven approach, which integrates CDIO with Knowledge Graphs, is designed to synergistically enhance both knowledge acquisition and engineering competency. The proposed framework aims to provide a replicable and scalable model for the reform of front-end development curricula.

2.Research Background and Current Situation Analysis

2.1 Challenges in Vue.js Experimental Teaching

Current experimental instruction in Vue.js predominantly encounters the following issues: (1) Fragmented knowledge points with weak correlations: Teaching is often linearly structured around isolated modules (e.g., syntax, directives, components), hindering students' ability to comprehend the integrated application logic of these elements in real-world projects. (2) Insufficient project authenticity: Experimental cases are frequently oversimplified or fragmented, lacking complete business workflows and authentic application scenarios, which creates a disconnect from industry practices. (3) Narrow focus in skill cultivation: An overemphasis on syntax and tool usage often sidelines the development of essential engineering competencies such as system design, team collaboration, and innovative iteration. (4) Outdated assessment methods: Evaluation remains predominantly based on final code output and experimental reports, failing to effectively measure process-oriented abilities and comprehensive engineering skills.

2.2 Current Applications of CDIO and Knowledge Graph in Education

The CDIO (Conceive-Design-Implement-Operate) engineering education model has been successfully implemented across multiple engineering disciplines. Its “learning by doing” and project-driven philosophy have been proven effective in enhancing students’ practical engineering capabilities. For instance, a review of CDIO reforms in Chinese universities noted that this model addresses the classic disconnect between theoretical knowledge and practical application by integrating knowledge acquisition with competency development, yielding significant outcomes in fields like mechanical and electronic engineering[2]. Extending this approach, recent studies have applied the OBE-CDIO framework to programming courses to cultivate comprehensive practical skills through project-driven methods[4], and integrated C language programming into automated system projects to provide an “early engineering experience”[5]. Notably, graph theory has been employed to optimize the structure of CDIO computing courses, ensuring coherent knowledge point coverage[6]. While these studies underscore the broad applicability of CDIO in engineering education, most focus on traditional disciplines, with limited in-depth exploration of its adaptation to modern technology stacks like Vue.js.

Concurrently, knowledge graph technology, representing a deep integration of artificial intelligence and education, demonstrates significant potential for structuring curricular knowledge systems, recommending personalized learning paths, and organizing teaching resources intelligently. Its educational applications are diverse: for example, clustering combined with knowledge graphs has been used to identify at-risk students in online environments by modeling entity relationships[7]; cosine similarity-based knowledge graphs enhanced with contextual signals have improved the precision of personalized content recommendations[8]; and multi-level knowledge graph generation methods have been shown to boost learning outcomes in online settings[9]. Furthermore, knowledge graphs facilitate interdisciplinary learning by linking formal education with resources from platforms like social media[10]. Despite these advancements, most applications remain confined to specific scenarios such as risk prediction or content recommendation, with a notable lack of systematic integration with established engineering pedagogical frameworks like CDIO.

Given the demonstrated pedagogical strengths of the CDIO framework and the structural advantages of knowledge graphs, significant potential exists for their integration in technical course design. A key research gap lies in tailoring this integration for specific technology stacks, such as Vue.js. Specifically, it remains unclear how to seamlessly embed knowledge graphs to dynamically interact with the distinct stages of the CDIO cycle (Conceive, Design, Implement, Operate), thereby achieving

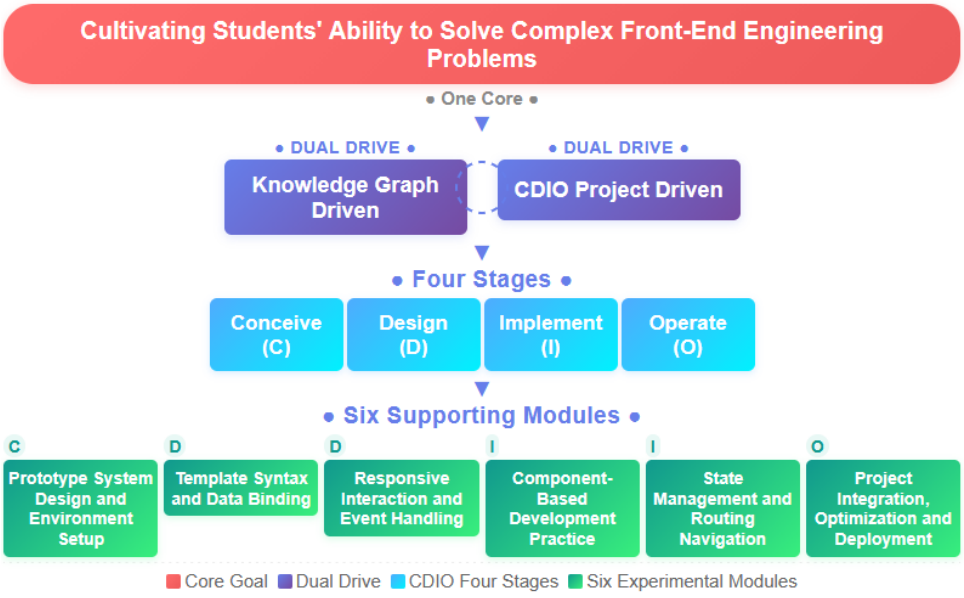
a synergistic enhancement of both knowledge assimilation and engineering skill development. To address this gap, this paper proposes an integrated instructional model that combines CDIO with knowledge graphs. This model aims to leverage the structured representation of knowledge graphs to systematically support and enhance the entire project-driven CDIO process.

3.Teaching Reform Model Design

3.1 Overall Framework Design

This paper proposes an integrated teaching model structured around a core objective, dual drivers, a four-stage process, and six supporting modules (as illustrated in Figure 1). The core objective is to cultivate students' ability to solve complex front-end engineering problems. The model is dual-driven by a knowledge graph, which structures the knowledge system, and by CDIO projects, which anchor engineering practice. The four-stage process integrates the complete CDIO cycle—Conceive, Design, Implement, and Operate—throughout the experimental teaching. This process is enacted through six progressive experimental modules, each constituting a micro-cycle of the CDIO stages and designed to incrementally build competency with Vue.js core knowledge points.

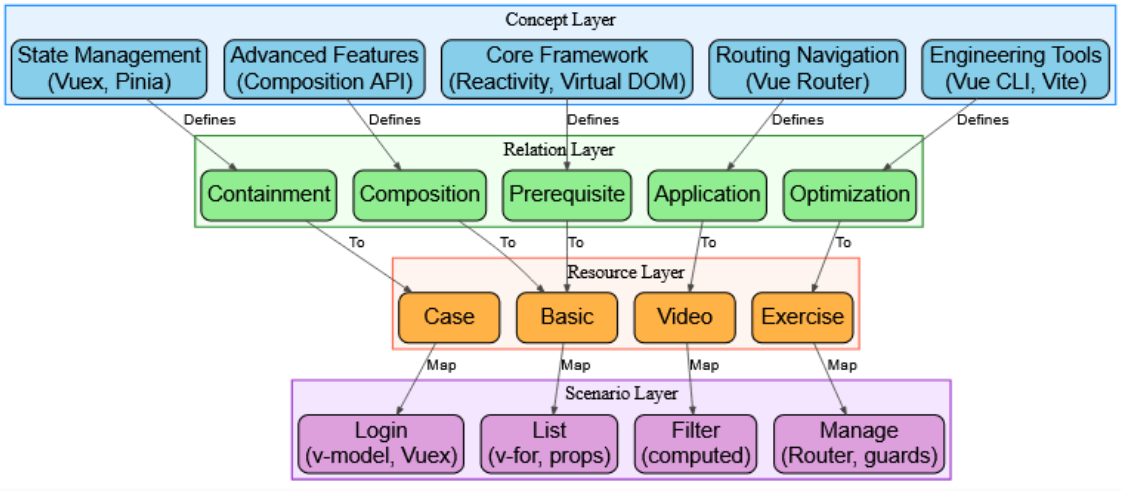
Figure 1: Overall Framework of the CDIO and Knowledge Graph Integrated Teaching Model



3.2 Construction of the Vue.js Course Knowledge Graph

The Vue.js course knowledge graph is constructed with a four-layer architecture—Concept, Relationship, Resource, and Project Scenario (Figure 2)—to achieve semantic association and visual presentation of knowledge points while ensuring deep integration with engineering practice.

Figure2: Four-Layer Architecture of the Vue.js Knowledge Graph



Concept Layer: This core layer defines the domain concept entities for the Vue.js course, categorized into Core Framework, State Management, Routing Navigation, Engineering Tools, and Advanced Features, encompassing 86 core entities.

Relationship Layer: This layer defines semantic associations between concepts through five formal relationship types: Prerequisite (isPrerequisiteOf), Containment (contains), Application (appliesTo), Optimization (optimizesFor), and Composition (composesTo).

Resource Layer: This layer attaches multimodal teaching resources (Basic, Case, Video, Exercise) to each concept node via semantic matching, supporting dynamic updates.

Project Scenario Layer: To bridge theory and practice, this layer maps abstract concepts to concrete functional modules within a core project case—the "Book Query and Borrowing System."

3.3 CDIO-Based Experimental System Design

Centered on the "Book Query and Borrowing System," the experimental curriculum is decomposed into six progressive modules, each aligning with a primary CDIO stage to form a complete project iteration cycle (see Table 1).

No.	Experimental Project	Core Knowledge Points & Tasks	Primary CDIO Stage
1	Prototype Design & Environment Setup	Requirement analysis, tech stack selection, Vue CLI setup, prototype design.	Conceive (C)
2	Template Syntax & Data Binding	Interpolation, directives (v-bind, v-if, v-for), MVVM pattern, static data binding.	Design (D)
3	Reactive Interaction & Event Handling	Methods, computed, watch, v-model for forms, event modifiers.	Design (D)
4	Component-Based Development Practice	Parent-child communication (props/events), slots, reusable components, lifecycle hooks.	Implement (I)
5	State Management & Routing Navigation	Vuex (state, mutations, modules), Vue Router (routes, guards), state-route coordination.	Implement (I)
6	Integration, Optimization & Deployment	API integration (Axios), UI library, performance optimization, build & deployment, documentation.	Operate (O)

Module 1: Conceive. This initial module focuses on the Conceive (C) stage, aiming to cultivate students' analytical understanding of complex engineering problems. Tasks include: (1) analyzing the core functional requirements of the book system (e.g., user authentication, book browsing, borrowing management); (2) selecting a technology stack (Vue.js, Vue Router, Vuex) informed by comparative analysis within the knowledge graph; (3) setting up the development environment using Vue CLI; and (4) designing interactive prototypes with tools like Figma. The goal is to establish a systematic pre-coding understanding of the project, adhering to the software engineering tenet that early design prevents later defects.

Module 2: Design (Foundations). Corresponding to the Design (D) stage, this module centers on Vue.js's template syntax and reactive data binding. Students learn to: (1) use interpolation and core directives (v-bind, v-if, v-for); (2) comprehend the Model-View-ViewModel (MVVM) architecture to drive view updates via data changes; (3) implement dynamic data binding, such as rendering book lists from API-fetched JSON; and (4) apply dynamic CSS binding (:class, :style). A key pedagogical challenge is facilitating the transition from imperative DOM manipulation to a reactive, data-driven paradigm, a process supported by the semantic concept associations in the knowledge graph.

Module 3: Design (Advanced). This module deepens the Design (D) focus by exploring Vue.js's reactive system and event handling. Students: (1) define and invoke component methods; (2) utilize computed properties for derived data and watch for side effects; (3) implement form handling with v-model; and (4) apply event modifiers (.stop, .prevent). A comparative exercise highlights the performance advantage of computed properties over methods, a rationale underpinned by the optimizesFor relationship in the knowledge graph.

Module 4: Implement (Components). Entering the Implement (I) stage, this module emphasizes Vue.js's component-based architecture. Key activities include: (1) creating reusable functional components (e.g., book cards); (2) mastering communication patterns (props/events, event bus/Vuex, provide/inject); (3) applying slots for content distribution; and (4)

understanding component lifecycle hooks. Assessment focuses on reusability and maintainability, guided by principles like the Single Responsibility Principle (SRP) and Open-Closed Principle (OCP). The knowledge graph's composition relationship illustrates how simple components integrate into complex systems.

Module 5: Implement (Architecture). Continuing the Implement (I) stage, this module addresses architecture for larger applications. Students: (1) implement centralized state management with Vuex (state, mutations, actions, getters); (2) organize state using namespaced modules; (3) configure routing with Vue Router; (4) set up navigation guards for access control; and (5) manage state preservation across routes. The knowledge graph clarifies the applicability of Vuex (e.g., via appliesTo relationships) to navigate the inherent complexity of state management.

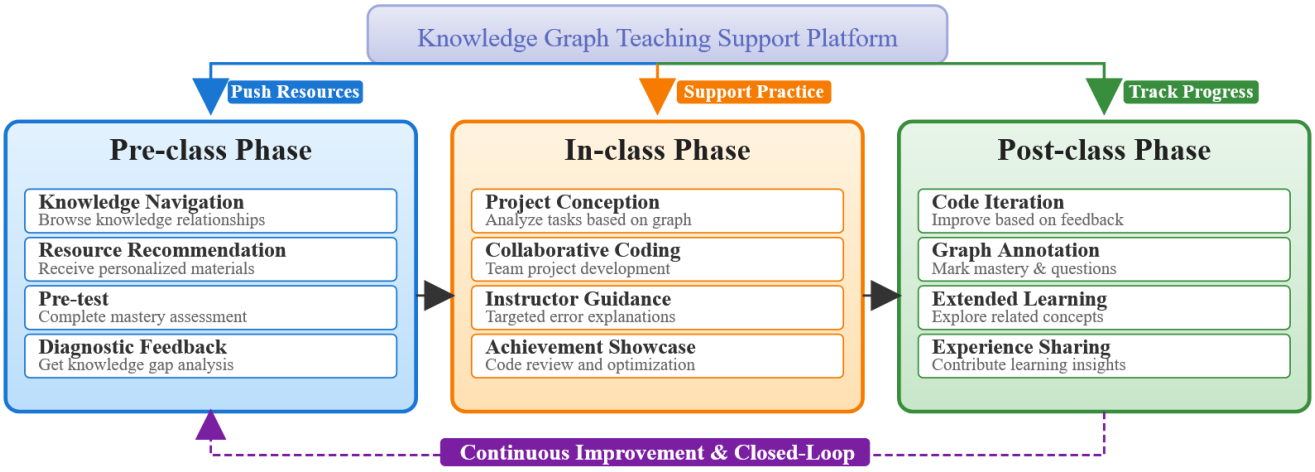
Module 6: Operate. The final module aligns with the Operate (O) stage, cultivating project delivery capabilities. It covers: (1) integrating third-party UI libraries (e.g., Element UI); (2) connecting to backend RESTful APIs using Axios with error handling; (3) applying performance optimizations (e.g., lazy loading); (4) building and deploying the project with Vite/Webpack; and (5) writing project documentation. This module embodies DevOps practices, and the project scenario layer of the knowledge graph provides authentic context for these operational tasks.

4.Teaching Reform Implementation Process

4.1 Teaching Implementation Process

The implementation follows an integrated "pre-class – in-class – post-class" design, supported throughout by the knowledge graph and the CDIO framework, thereby forming a closed-loop instructional process (as illustrated in Figure 3).

Figure 3: Teaching Implementation Process Flowchart



The primary objective of the pre-class stage is to equip students with the foundational knowledge required for the experimental tasks. The specific procedures include:

(1) Knowledge Graph Navigation: Students access the Vue.js knowledge graph via an online learning platform (e.g., Learning Pass) to browse knowledge points relevant to the experiment and their interrelationships. Starting from core concepts, students can explore associated knowledge domains.

(2) Personalized Resource Recommendation: Based on students' historical learning data—such as knowledge mastery, learning styles, and error records—the system recommends appropriate learning resources.

(3) Pre-knowledge Testing: Students complete micro-tests (5–10 multiple-choice questions) drawn from the resource-layer question bank. The system automatically evaluates knowledge mastery, employing adaptive testing technology to dynamically adjust question difficulty according to student responses.

During the in-class stage, CDIO engineering practice serves as the main thread, with instructors guiding students to apply acquired knowledge in authentic projects. This phase comprises:

(1) Conceive Stage: Instructors assign module tasks, requiring students to analyze the relationship between the tasks and prior knowledge using the knowledge graph. By querying the graph to identify key concepts, students engage in solution conception, thereby cultivating systematic thinking.

(2) Design & Implement Stage: Students undertake programming practice in small groups (4–6 members).

(3) Operate Stage: The final 20 minutes of each session are dedicated to outcome presentation and review. Each group demonstrates the iterative results of the module, while other groups conduct code reviews, highlighting strengths and suggesting improvements.

The post-class stage focuses on knowledge consolidation and expansion, closing the loop of continuous learning. Activities include:

(1) Code Iteration: Students revise their code based on feedback from in-class reviews and submit updates to Git repositories.

(2) Knowledge Graph Annotation: Students mark mastered knowledge points (green) and points requiring further clarification (red) on the knowledge graph. The system then generates personalized knowledge maps based on these annotations, visually representing mastery levels and helping students identify knowledge gaps.

(3) Extended Learning: Students with additional capacity may pursue in-depth study through extended resources linked to the knowledge graph, such as analyzing open-source projects, reading technical blogs, examining source code, and undertaking innovative practice.

4.2 Construction of Diversified Evaluation System

A diversified evaluation system aligned with CDIO capability objectives and the knowledge-graph-enhanced learning process was established (Table 2). This system emphasizes process-oriented, value-added, and comprehensive assessment, shifting away from a traditional score-centric approach.

Table 2 Diversified Evaluation System Based on CDIO Philosophy

Evaluation Dimension	Evaluation Content	Evaluation Method	Proportion
Process Evaluation (50%)	Pre-class knowledge tests, in-class participation, code submission quality/frequency, group collaboration, problem-solving trajectory	Platform auto-recording (30%), teacher observation (15%), peer evaluation (5%)	50%
Summative Evaluation (40%)	Final project completeness, code standardization, system functionality/performance, project defense and report	Teacher evaluation (30%), cross-review (10%)	40%
Value-Added Evaluation (10%)	Improvement in knowledge graph mastery; completion of extended tasks (e.g., open-source contributions, technical blogs)	Knowledge graph system analysis (8%), teacher assessment (2%)	10%

Process Evaluation focuses on students' learning progress and trajectory rather than solely on final outcomes. It encompasses pre-class tests, in-class engagement, code submission quality and frequency, group collaboration, and problem-solving processes. Data are automatically captured via the learning platform, ensuring objectivity and traceability.

Summative Evaluation assesses students' comprehensive abilities at the course conclusion, including final project delivery, code standardization, system functionality and performance, and the quality of project defense and reporting. A multi-round cross-review mechanism involving different instructors and groups helps mitigate subjective bias.

Value-Added Evaluation emphasizes student growth and extended learning, encouraging efforts beyond basic course requirements. This includes measuring progress in knowledge graph mastery and recognizing accomplishments such as contributing to open-source projects on GitHub, publishing Vue.js-related technical articles, participating in community discussions, and mastering advanced Vue.js features not covered in the core syllabus. The underlying philosophy is to promote continuous learning and self-improvement rather than mere compliance with course minima.

5.Analysis and Discussion of Teaching Effects

5.1 Experimental Design and Data Collection

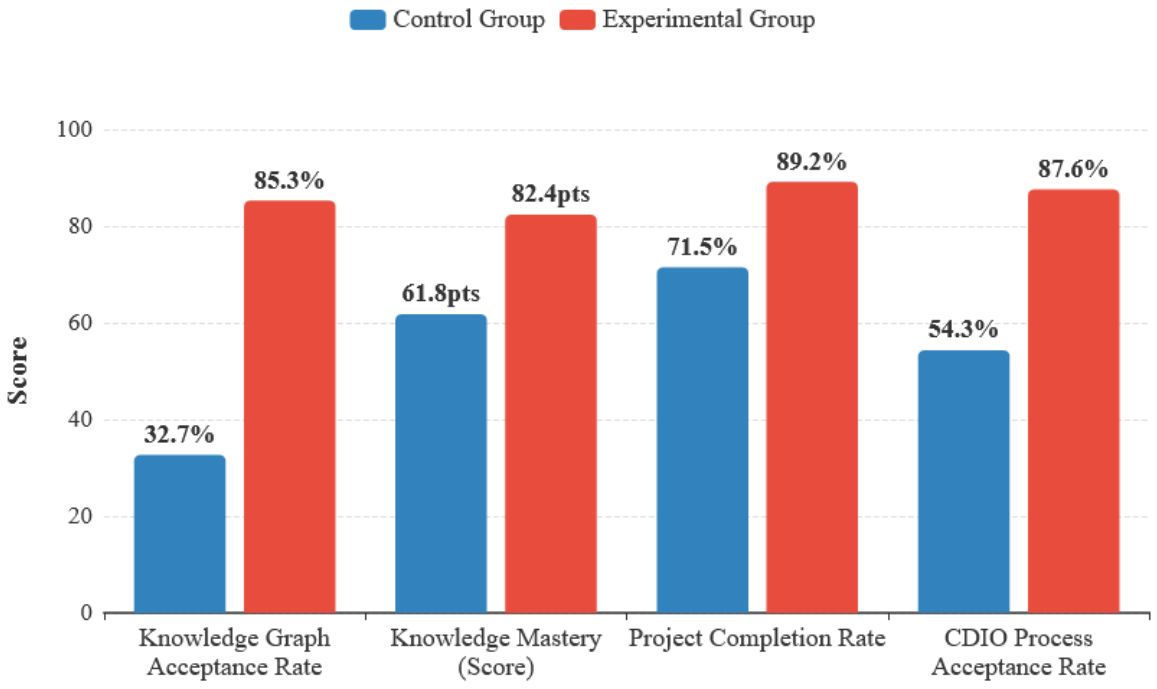
A comparative teaching experiment was conducted during the 2024–2025 academic year at the School of Computer and Software Engineering, University of Science and Technology Liaoning. The experimental group ($n \approx 500$ students across 14 classes; comprising 6 undergraduate classes with approximately 200 students and 8 top-up program classes with approximately 300 students) was instructed using the CDIO-knowledge graph integrated teaching model. The control group

(n ≈ 500 students across 14 classes) was selected from the same major during the preceding 2023–2024 academic year and received instruction via traditional experimental teaching methods. To ensure internal validity, key teaching conditions—including textbooks, total class hours, and assessment criteria—were held consistent between the two groups. Data collected for analysis included academic performance metrics (experiment scores, final examination scores, and project scores) and post-course questionnaire responses.

5.2 Teaching Effect Analysis

Questionnaire results indicate a pronounced positive perception of knowledge-graph-assisted learning among the experimental group. Specifically, 85.3% of these students acknowledged that the knowledge graph aided in understanding interconnections between knowledge points and in constructing a systematic cognitive framework—a proportion significantly higher than the 32.7% reported in the control group. Academic performance data further substantiate the model's effectiveness: the experimental group achieved an average score of 82.4 points, markedly surpassing the control group's average of 61.8 points. In terms of engineering practice capability, the experimental group demonstrated exceptional outcomes, with average project completeness reaching 89.2%, far exceeding the 71.5% observed in the control group. Notably, 87.6% of students in the experimental group reported that project-based practice substantially deepened their theoretical understanding, compared to 54.3% in the control group. These findings collectively underscore the synergistic benefits of integrating theoretical knowledge with practical application, contributing significantly to the cultivation of comprehensive student competencies.

Figure 4: Comparison of Learning Outcomes between the Experimental and Control Groups



6.Conclusion

This study addresses the prevalent issues of knowledge fragmentation and inadequate development of engineering practice skills in Vue.js experimental instruction by proposing an integrated instructional model based on CDIO and knowledge graphs. The construction of a dynamic Vue.js knowledge graph provides students with a systematic and visually structured navigation tool for knowledge acquisition. Concurrently, the design of a six-module, iterative experimental project—deeply embedded with the CDIO philosophy—enables students to undergo comprehensive training across the complete Conceive–Design–Implement–Operate cycle within an authentic engineering context. Teaching practices and effect evaluation demonstrate that this integrated model yields significant positive outcomes. It aligns effectively with the talent development objectives of emerging engineering disciplines and the standards of engineering education accreditation. Future work will focus on validating and refining this model across a broader range of institutions and academic majors.

Funding

Experimental Teaching Reform Project of University of Science and Technology Liaoning - Research on Experimental Teaching Reform of Vue.js Course Based on 'CDIO + Knowledge Graph' (Project No.: PX-2125437).

Conflict of Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Reference

[1] Li, N., & Zhang, B. (2021). The Research on Single Page Application Front-end development Based on Vue. Journal of Physics: Conference Series, 1883(1), 012030. https://doi.org/10.1088/1742-6596/1883/1/012030

[2] Jianfeng, B., Hu, L., Li, Y., Tian, Z., Xie, L., Wang, L., Zhou, M., Guan, J., & Xie, H. (2013). The Progress of CDIO Engineering Education Reform in Several China Universities: A Review. Procedia - Social and Behavioral Sciences, 93, 381–385. https://doi.org/10.1016/j.sbspro.2013.09.207

[3] Qu, K., Li, K. C., Wong, B. T. M., Wu, M. M. F., & Liu, M. (2024). A Survey of Knowledge Graph Approaches and Applications in Education. Electronics, 13(13), 2537. https://doi.org/10.3390/electronics13132537

[4] Yuan, X., Wan, J., An, D., Lu, J., & Yuan, P. (2024). Multi-method integrated experimental teaching reform of a programming course based on the OBE-CDIO model under the background of engineering education. Scientific Reports, 14(1), 16623. https://doi.org/10.1038/s41598-024-67667-6

[5] Wang, S., Liu, J., Dong, Y., & Wei, Q. (2021). C programming language teaching based on cdio. Proceedings of the International CDIO Conference.

[6] Pavlich-Mariscal, J., & Curiel, M. (2019). CDIO curriculum design for computing: A graph-based approach. Proceedings of the 15th International CDIO Conference.

[7] Albreiki, B., Habuza, T., Palakkal, N., & Zaki, N. (2024). Clustering-based knowledge graphs and entity-relation representation improves the detection of at risk students. Education and Information Technologies, 29(6), 6791–6820. https://doi.org/10.1007/s10639-023-11938-8

[8] Troussas, C., Krouska, A., Tselenti, P., Kardaras, D., & Barbounaki, S. (2023). Enhancing Personalized Educational Content Recommendation through Cosine Similarity-Based Knowledge Graphs and Contextual Signals. Information, 14, 505. https://doi.org/10.3390/info14090505

[9] Xiao, X., Fang, Z., Zou, S., Zhang, C., & Chen, X. (2024). Effects of an intelligent cues recognition-based multilevel knowledge graphs generation method on students in online learning environments. Interactive Learning Environments, 32(9), 5801–5821. https://doi.org/10.1080/10494820.2023.2236668

[10] Zablith, F. (2022). Constructing social media links to formal learning: A knowledge Graph Approach. Educational Technology Research and Development, 70(2), 559–584. https://doi.org/10.1007/s11423-022-10091-2