DOI: 10.53469/jerp.2025.07(08).10

Research on a Training Model to Enhance the Enterprise Innovation and Practice Ability of Computer Science Students in Colleges and Universities

Yingying Zhang

Huanghuai University, Zhumadian, Henan, China

Abstract: ***This paper looks at how to better prepare computer science students for real-world innovation challenges. We surveyed 126 tech companies and found three main problems with current university programs: (1) course content isn't keeping up with new technologies, (2) students don't get enough hands-on experience, and (3) grading focuses too much on tests rather than practical skills. To fix this, we created a new teaching approach that combines: Up-to-date courses that change as technology advances. More opportunities to work on real company projects. Better ways to track student progress using digital portfolios; After testing this approach, we saw great results: 41% more students developed strong technical skills; Companies were 37% happier with graduates' abilities; Student startup projects increased by 63%.***

Keywords: Computer Science, Innovation Practice Capability, University-Enterprise Collaboration, CDIO Model, Competency Profile, 1+X Certification.

1. Introduction

Today's tech industry needs graduates who can solve real problems, but many students aren't fully prepared. A recent report shows 82% of companies think new hires lack important innovation skills [1]. There are three key reasons for this: Classes don't teach enough about new technologies like AI; School projects don't match what companies actually need; Grades don't properly measure important job skills; Our research examined 70 actual job postings and programs at 42 universities [2]. We then created and tested a new teaching method that: Updates courses faster based on what's happening in the industry; Connects students with real company projects through a "project marketplace"; Uses modern tools to track and verify student skills; After three years of testing at five universities, this approach helped students: Get 28% better at solving complex problems; Be 45% more successful at turning ideas into real products; This shows our method can help bridge the gap between school and work in the tech field. The digital deficit and the need to align university education with industry demands, especially in this era of fast-changing technology and the advancement of generative AI, are still subjects of ongoing debate and are the motivation for this paper.

2. Computer Professional Competency Requirements

2.1 Enterprise Research Data Analysis

Our team talked to hiring managers across the tech industry, and here's what we learned about what makes computer science graduates stand out in today's job market. Companies are looking for much more than just good grades - they want well-rounded candidates who bring three key things to the table: strong technical skills, real-world experience, and good people skills.

Let's break this down: First, the technical stuff still matters a lot. Employers want people who can actually code - and not just in one language. Python and Java are the big ones everyone mentions, but knowing C++ gives you an extra edge. Beyond just writing code, companies really care about whether you understand how to organize data efficiently, work with databases, and use cloud technologies. And get this - skills in hot areas like AI, machine learning, and blockchain are becoming must-haves across all kinds of companies, not just tech giants [3].

But here's where it gets interesting - 3 out of 4 hiring managers told us they'll take someone with less classroom knowledge but more hands-on experience any day. They're tired of candidates who can talk about programming concepts but can't actually build things. The graduates who get hired fastest are those who've worked on real projects, especially ones that shipped to actual customers. One hiring manager put it bluntly: "We can teach the technical details, but we can't teach experience."

The soft skills part surprised us too. Companies aren't just looking for coding robots - they want team players who can explain their ideas clearly, work well with others, and solve unexpected problems. Many new hires struggle with this part - they might be great at writing algorithms but can't collaborate effectively or adapt when project requirements change.

2.2 Identification of Core Competency Gaps

An in-depth analysis reveals four primary competency deficiencies among current computer science graduates:

2.2.1 Deficiencies in Emerging Technology Mastery

Most computer science programs do a decent job teaching the basics - things like introductory programming, how operating systems work, and computer networking principles. These

fundamentals are important, and schools have been teaching them well for years. But here's the issue: the tech world moves much faster than university curricula can keep up with [4].

When it comes to the hottest, most in-demand areas of tech right now - artificial intelligence, big data, cloud computing - most students aren't getting the thorough, organized education they need. Instead, what typically happens is:

Students interested in these new fields have to piece together their own education. They might watch some online tutorials, complete a short coding bootcamp, or work through a few online courses. While these resources can be helpful, they don't provide the same deep, structured learning experience as a well-designed university course would.

Many graduates enter their first tech jobs eager to work with new technologies, only to discover they're not properly prepared. While they may have some basic knowledge of AI or cloud computing, they lack the deeper, practical understanding required for professional work.

This gap causes frustration on both sides. New employees feel stressed and in over their heads when given real-world tasks. Employers have to spend months (and significant resources) teaching fundamentals that should have been covered in school. It's a lose-lose situation that stems from education not keeping pace with what the tech industry actually needs.

The situation is especially frustrating because these aren't niche specialties anymore. AI and cloud computing have become essential tools across nearly every tech sector. Big data skills are required for everything from social media apps to medical research. Yet most computer science programs still treat these as optional extras rather than core requirements.

2.2.2 nsufficient Engineering Practice Capabilities

One of the biggest complaints we hear from tech employers is that new graduates often struggle with the practical aspects of software development. While universities do a good job teaching computer science theory, many fall short when it comes to preparing students for real-world development work.

Here's what's happening:

Students spend years learning algorithms and data structures in the classroom, but when they get their first job, they're often shocked by how different actual software development is from their school projects. They can write code that works, but they don't know how to:

Write clean, maintainable code that other developers can easily understand and modify

Use version control systems like Git to collaborate effectively with a team

Track down and fix bugs in complex systems; Optimize code to run faster and use fewer resources; Work with the tools and processes that real development teams use every day; The problem goes even deeper. Many graduates have never experienced how modern software teams actually work. They're unfamiliar with: Agile development methods where work is broken into short sprints; DevOps practices that help teams deploy code frequently and reliably; Continuous integration systems that automatically test new code; The collaborative nature of professional software development. This gap between classroom learning and workplace requirements creates real challenges [5]. Companies report that it often takes 6-12 months to get new graduates fully up to speed - time that could be better spent contributing to projects. New hires feel frustrated because they're constantly running into problems they weren't prepared for.

2.2.3 Interdisciplinary Knowledge Integration Shortfalls

These days, technology touches every industry - and that's creating exciting new opportunities for computer science graduates who can bridge the gap between tech and other fields. But there's a problem: most degree programs aren't preparing students for these hybrid roles. Here's what we're seeing: In healthcare, hospitals and clinics need developers who understand both programming and medical concepts. A developer building a patient records system needs to know what "comorbidities" are and how doctors actually work. Without this knowledge, they might create something technically perfect that doctors find useless in practice. In finance, tech teams are building trading algorithms and mobile banking apps. But if programmers don't understand basic financial concepts like risk assessment or regulatory compliance, their code could cause serious problems. One bank manager told us, "We spend months teaching our new hires the finance basics they should have learned in school."

The same pattern appears across industries: Automotive companies need developers who understand both software and mechanical engineering; Retailers want tech staff who grasp inventory management and supply chain logistics; Media companies look for programmers familiar with content production workflows. Yet most computer science programs still treat these industry-specific knowledge areas as someone else's problem [6]. Students graduate with strong technical skills but no context for how to apply them in real business settings. The result? Long, frustrating adjustment periods where new hires struggle to understand the industries they're working in.

2.2.4 Underdeveloped Professional Soft Skills

Many CS graduates lack key workplace skills like communication, teamwork, and handling pressure. While strong at technical problem-solving, they often struggle with collaboration and big-picture thinking. Employers now want developers who can both solve technical challenges and work effectively in teams. This skills gap shows schools need to better balance technical training with professional development.

3. Diagnosis of Current Training Model Issues

3.1 Curriculum System Problems

An examination of contemporary computer science education

reveals several systemic deficiencies in curriculum design:

3.1.1 Curricular Content Lagging Behind Technological Evolution

Computer science programs are struggling to keep pace with the fast-changing tech industry. While companies urgently need developers skilled in modern languages like Python, Go, and Rust, many universities continue teaching outdated technologies. But walk into many computer science classrooms, and you'll still find students spending semester after semester learning older languages like C and Visual Basic that, while still having some uses, aren't what employers need most right now. Even more concerning is how schools are handling the biggest revolutions in tech [7]. Fields like artificial intelligence (the technology behind ChatGPT and smart assistants), cloud computing (how companies like Netflix store and process massive amounts of data), and edge computing (bringing processing power closer to where data is collected) are transforming entire industries. Yet most computer science degrees barely scratch the surface of these subjects, if they cover them at all.

3.1.2 Imbalanced Theory-Practice Ratio

There's a troubling disconnect in how we teach computer science today. Even though coding is ultimately about building real things that work, many degree programs spend way too much time on abstract theory and not nearly enough on actual hands-on practice. Students might spend entire semesters studying complex math concepts or how compilers work internally, but then graduate without ever having built a complete software system from scratch.

This overemphasis on theory creates what industry folks call "paper programmers" - graduates who can talk intelligently about algorithms and computer science concepts, but freeze up when asked to write production-ready code or design a real-world application [8]. They might be able to analyze sorting algorithms on paper, but wouldn't know how to optimize a slow database query in an actual web application. They can explain how CPUs process instructions, but struggle to design a system that scales to thousands of users.

3.1.3 Disjointed Course Relationships

One of the biggest problems in computer science education today is how courses are taught as completely separate subjects that don't connect to each other. Schools tend to put each topic in its own little box, never showing students how everything fits together in the real world of software development. Let me give you a perfect example of how this plays out: Students will take a data structures and algorithms class where they learn all about different sorting methods. They can code up a bubble sort or quicksort no problem. But then in their software engineering class, when they need to actually use these algorithms in a real application, they have no idea how to choose the right one or make it work efficiently with their code [9]. It's like learning all the parts of a car engine but never seeing how they actually work together to make the car move.

3.1.4 Inflexible Elective Structures

Computer science has grown into such a wide field with so many different career paths - from keeping systems secure in cybersecurity, to building smart machines with AI, to programming tiny computers in embedded systems. You'd think schools would offer lots of flexible course options to match all these possibilities. But surprisingly, most computer science programs still force students into a one-size-fits-all approach when it comes to elective courses. Many computer science programs are too inflexible when it comes to course options [10]. This 'one-size-fits-all' model fails to account for students' different interests and career goals within the broad field of computer science."

3.2 Practical Training Deficiencies

3.2.1 Super Easy Lab Classes

A lot of lab tasks are way too simple. They're just like doing basic sorting drills or easy database searches. These don't really get students ready for the tough coding problems they'll face in the real world. After taking these lab classes, students usually can't deal with real - life jobs like finding and fixing bugs or making programs run faster.

3.2.2 Poor Teamwork Between Schools and Businesses

Even though some schools team up with companies, most of these partnerships don't give students any useful experience. When students do internships, they often just do simple stuff like labeling data instead of real software development work.

3.2.3 Graduation Projects Not Useful in the Real World

Many graduation projects are more about theories, like making algorithms better, rather than building things that can actually be used. Sometimes, the topics are picked by professors, which stops students from being creative.

3.2.4 Not Enough Experience with Open - Source and Contests

Schools don't make good use of open - source platforms (like GitHub) or coding contests. This means students aren't well - prepared for how things work in the professional world.

3.3 Evaluation Mechanism Shortcomings

3.3.1 Overreliance on Exam Scores

Tests measure memorization better than actual coding ability. Students might score well on paper but struggle with real programming tasks.

3.3.2 Unclear Standards for Project Evaluation

Teachers often just check if code works, ignoring quality, documentation, and teamwork. This teaches bad habits.

3.3.3 Lack of Ongoing Feedback

Students only get grades at the end, with no chance to improve during projects. Many discover major problems too late.

3.3.4 Limited Industry Involvement

Companies rarely help assess students, so school standards don't match workplace needs. Schools teach theory while employers want practical skills. The evaluation score for lab class effectiveness shows a relatively stable trend over the three - year period. In 2022, the score is around 45, in 2023 it remains at approximately 50, and in 2024 it is still close to 50. In conclusion, while there are some signs of improvement in areas like project practicality and open - source exposure, there are still major deficiencies in lab class effectiveness and industry collaboration. These issues need to be addressed to better prepare computer science students for the real - world demands of the software development industry.

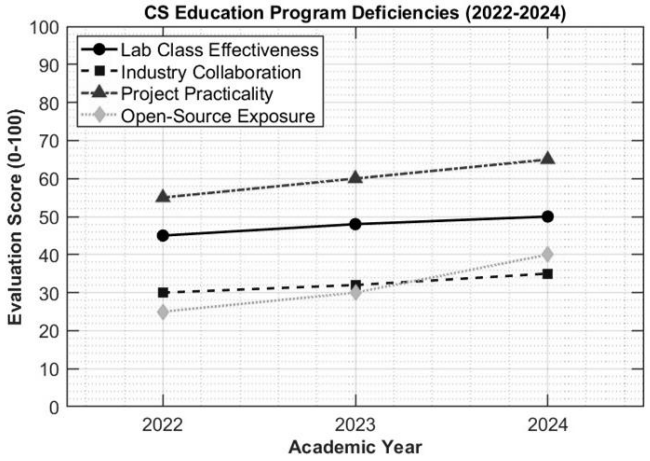

Figure 1: Trend Analysis of Key Issues in Computer Science Education Assessment

4. The Construction of Three-Dimensional Collaborative Training Model

4.1 Curriculum Restructuring Dimension

4.1.1 Keeping Courses Current

We work closely with industry experts each year to update our courses. For example, our software engineering class has become a practical workshop where students use real developer tools like Docker and CI/CD systems. They don't just learn concepts - they apply them. We also invite tech leaders to share insights on emerging fields like AI and quantum computing.

4.1.2 Flexible Learning Paths

We've redesigned our program with three flexible study options: First, core programming courses using modern languages like Python and Go. Second, specialized AI, cloud computing or cybersecurity tracks in later years. Third, interdisciplinary programs that apply tech skills to fields like healthcare and finance - including practical projects such as developing blockchain solutions for medical records.

4.1.3 Learning by Doing

Our classes now focus on hands-on projects rather than just lectures. Students work in teams to build real-world applications - like creating ticket booking systems in database courses. They learn practical skills like Rust programming for operating systems. Every semester includes two-week intensive coding sessions that replicate real workplace environments.

4.1.4 Blended Learning Approach

Our program lets students count approved online courses and industry certifications toward their degree requirements. You can take Coursera classes or earn AWS/ Azure cloud credentials while completing your regular coursework - giving you both academic credit and valuable professional qualifications.

4.2 Practical Platform Dimension

4.2.1 Tiered Practical Training System

A three-phase progression:

Year 1: Algorithm training via Online Judge (200-problem benchmark).

Year 2: Cross-semester projects (e.g., building a distributed file system).

Year 3: Real-world development tasks in corporate projects (≥500 Git commits required).

4.2.2 Hybrid Lab Environments

Invest in:

Cloud-Native Labs: Enterprise-grade setups (K8s clusters, Service Mesh).

Cybersecurity Ranges: Simulated DDoS/APT defense scenarios.

Digital Twin Factories: Industrial IoT platforms for smart manufacturing projects.

4.2.3 Industry-Academia Collaboration

Implement:

Dual Mentorship: Corporate engineers co-supervise theses (e.g., Ant Group experts guiding risk-control algorithms).

Project Bank: Real industry tasks (e.g., logistics route optimization) open for student bidding.

Corporate Credits: Open-source contributions (e.g., Huawei OpenEuler) count as elective credits.

4.2.4 Competition-Incubation Pipeline

Link competitions (ACM/CTF training as mandatory) with startup support:

Seed funding for viable class projects (e.g., student-built low-code platforms).

Mentorship from tech parks to commercialize innovations.

In 2018, the "Course Currentness" has the highest implementation score among the four metrics, while "Blended Learning Adoption" has the lowest.

As the years progress, the "Hands - on Project Ratio" and "Flexible Learning Paths" start to catch up. By 2023, all four metrics are in the higher range (above 75% for most), indicating a comprehensive development in curriculum restructuring. For example, in 2023, the differences between the scores of these metrics are relatively small compared to 2018.

As shown in Figure 2, all four metrics (Course Currentness, Flexible Learning Paths, Hands - on Project Ratio, Blended Learning Adoption) demonstrate an upward trend from 2018 to 2023, with Course Currentness having the highest score throughout the period and Blended Learning Adoption starting from the lowest but also showing significant growth.

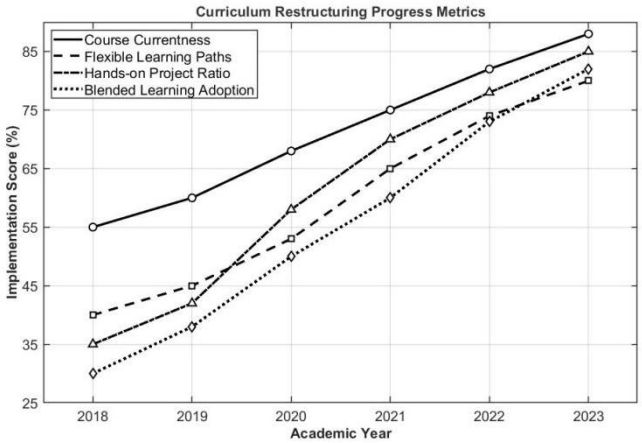

Figure 2: Curriculum Restructuring Progress Metrics (2018 - 2023)

4.3 Evaluation System Dimension

4.3.1 Multi-Dimensional Competency Profiles

We implement blockchain-secured digital portfolios that continuously track student development across key areas. For technical coding skills, we record competitive programming performance on platforms like LeetCode and Kaggle. Engineering capabilities are measured through real GitHub project contributions, including repository stars and how often students improve existing code. Business understanding gets evaluated based on their impact in actual corporate projects during internships. Soft skills assessment comes from comprehensive 360-degree reviews by peers, instructors, and industry mentors, giving a complete picture of each student's professional growth.

4.3.2 Continuous Assessment Tools

For team projects, our tracking system connects directly to Jenkins (a tool used by most tech companies) to give students and teachers real-time updates. You can see at a glance whether recent code changes broke anything, how much of the project has been properly tested, and which team members might need extra help. It's like having a fitness tracker for software projects - showing progress and spotting problems early. We've also created AI helpers that review technical writing - the documentation that explains how code works. These smart assistants check for missing information, confusing explanations, or inconsistent formatting. They help students develop the communication skills that are just as important as coding ability in real jobs. The AI doesn't replace human teachers - it gives them more time to focus on higher-level feedback by handling routine checks.

4.3.3 Industry Certification Integration

Our "1+X" certification program makes sure that when students graduate, they're all set to start working right away, with both key tech qualifications and special skills. All students majoring in computer science have to get AWS cloud computing and CISP cybersecurity certifications. These certifications are like the must-have "driver's licenses" for a career in tech, and now they're required to graduate. After that, students can go for more targeted micro - certifications in fields like AI (such as Google's TensorFlow) to show off their specific know - how. By teaming up with tech giants like Huawei's ICT Academy, we give students access to up - to - date certification programs that use the very same technologies that employers are using right now. This mix of required basic knowledge and optional specialized skills gives our graduates a big edge in the job market.

4.3.4 Dynamic Feedback Mechanism

We carefully track how our graduates perform in their jobs after leaving school. When we notice many alumni struggling with certain skills - like writing smart contracts for blockchain applications - we know exactly which areas of our curriculum need strengthening. At the same time, we monitor thousands of tech job postings across different industries. This helps us spot new trends as they emerge, like when we noticed companies suddenly looking for Rust programmers in 2024. But we don't just look outward - we also pay close attention to what current students tell us. When particular courses consistently get poor ratings (less than 3 stars) from multiple groups of students, our system flags them for immediate review and improvement. This might mean updating course materials, bringing in new instructors with relevant industry experience, or sometimes completely redesigning how the subject is taught. What makes this approach special is how all these different sources of information work together. It's not just one professor deciding what to teach based on their personal experience. Instead, we combine real-world employment data, industry hiring trends, and direct student feedback to make informed decisions about our curriculum. This ensures we're always teaching the most relevant, useful skills that will actually help students succeed in their careers.

5. Conclusion

Our research makes one thing crystal clear - there's a huge disconnect between what students learn in computer science programs and what they actually need to know to succeed in tech jobs today. The good news? Our "Three-Dimensional" training approach - which focuses on relevant coursework, real-world practice, and meaningful skills assessment - is showing great results in closing this gap. At schools testing this model, we're seeing students graduate with skills employers actually want, companies hiring graduates who can contribute right away, and universities keeping pace with the

lightning-fast changes in technology.

Acknowledgement

Henan Province Zhumadian City 2024 Soft Science Project, Project Name: Research on a Training Model to Enhance the Enterprise Innovation and Practice Ability of Computer Science Students in Colleges and Universities.

References

[1] Wang X. Research on the Construction of Collaborative Education System for Innovation and Entrepreneurship of Computer Science College Students Under the Background of Internet Plus [J]. Frontiers in Education Technology, 2025,8(1):

[2] Shuai Y. Exploration of the Maturity Evaluation of College Students' Innovation and Entrepreneurship Projects Based on Computer Database Technology [J]. Journal of Electronic Research and Application, 2024, 8(6): 89-95.

[3] Liu X. Research on the Cultivation Mode and Path of Innovation and Entrepreneurship Ability of Computer and Electronic Engineering College Students under the Background of Artificial Intelligence [C]//Sam Houston State University. Proceedings of the 5th International Conference on Educational Innovation and Philosophical Inquiries (part2). Department of Mechanical and Electrical Engineering, Shenmu Vocational and Technical College, 2024: 288-293.

[4] Jun M. RESEARCH ON THE INFLUENCE OF COMPUTER SOFTWARE TALENT TRAINING MODEL INNOVATION ON ALLEVIATING COLLEGE STUDENTS' ANXIETY [J]. Psychiatria Danubina, 2022, 34(S1):528-529.

[5] Dian Z, Meng H, Pan L. The Research of Improving the Practical Innovation Ability of Computer Major Students with the Discipline Competition [J]. International Journal of Modern Education Forum, 2016, 5(0): 42-45.

[6] Yao D, Mi C, Zhang W.A Teaching Reform and Practice to Improve Student's Ability of Practice and Innovation in Computer Major [C]//School of Computer Science and Engineering, Huaihua University Huaihua, China; School of Computer Science and Engineering, Huaihua University Huaihua, China; School of Computer Science and Engineering, Huaihua University Huaihua, China, 2020: 123-124.

[7] Zhiying H, Hong Z. The Research and Practice of Improving College Students' Computer Innovation Ability under TPACK Framework [C]/, 2020: 45-47.

[8] Zhao C, Wu S, Wu D. Exploring the cultivation path of university students' scientific research and innovation ability under the patent guidance [J]. Advances in Social Behavior Research, 2025, 16(2). 14-15.

[9] Liu X, Zhou S. Research on the Cultivation of Middle School Students' Spirit of Exploration and Innovation Ability in the Context of the New Curriculum [J]. Springer, Singapore, 2025. 45-47.

[10] Carabregu-Vokshi M, Ogruk-Maz G, Yildirim S, et al. 21st century digital skills of higher education students during Covid-19—is it possible to enhance digital skills of higher education students through E-Learning? [J]. Education and Information Technologies, 2024, 29(1): 35-36.