Journal of Education and Praxis Research. Volume 11, Issue 1, (2025) 11-19
http://www.cayley-nielson.com/jepr

The Role of Digital Education in the High-Quality Development of Undergraduate Teaching in Software Engineering

Desheng Li
School of Engineering, Guangzhou College of Technology and Business, Foshan City, Guangdong Province, 528138, China.

E-mail: lidesheng@gzgs.edu.cn

Abstract

As the global wave of digitalization sweeps across all industries, higher education, particularly engineering education, which is closely linked to information technology, faces unprecedented opportunities and challenges. As a strategic fulcrum for national development in the new era, "digitalization of education" is no longer an option but a must-answer for promoting educational modernization and achieving high-quality development. From the perspective of an undergraduate software engineering student, this article systematically explores how digital education can comprehensively and profoundly empower high-quality undergraduate education, drawing on the disciplinary characteristics and learning practices of this major. First, the article explains the inherent necessity and contemporary urgency of digital education for software engineering, a "natively digital" major. Second, it deeply analyzes the specific paths and potential effectiveness of digital application from multiple perspectives, including curriculum reconstruction, teaching resource development, teaching model innovation, practical ability cultivation, and evaluation system reform. Finally, it offers an outlook on the challenges and prospects for the future development of digital education. This article argues that the core of digital education is not a simple accumulation of technologies, but a systematic transformation of educational thought that is student-centered, competency-oriented, and data-driven. By building an open, collaborative, intelligent, and personalized new teaching ecosystem, we can effectively address the pain points in traditional engineering education, cultivate a new generation of outstanding software engineering talents with true innovative spirit, practical ability, and global competitiveness, and contribute solid strength to the grand goal of building a strong country in education and science and technology.

Keywords: High-Quality Development; Software Engineering; Undergraduate Teaching; Talent Training

1. Introduction

We are in a new era defined by data, algorithms, and computing power. From "Internet Plus" to "artificial intelligence" and now the global digital economy, digital transformation has become a core engine driving social progress and reshaping the industrial landscape. Within this grand narrative, higher education, as the cradle of future builders and innovators, must resonate with the pulse of the times in its reform and development. The report of the 20th National Congress of the Communist Party of China

explicitly called for "promoting the digitalization of education and building a learning society and a learning nation with lifelong learning for all," elevating the digitalization of education to the level of a national strategy. This is not only a profound insight into the trends of technological change, but also a far-reaching plan for achieving educational equity, improving educational quality, and serving national development.

Software engineering, as a discipline that studies and applies systematic, standardized, and measurable methods for developing and maintaining software, is itself a builder and shaper of the digital world. Our daily learning and practice, from writing the first line of "Hello, World!" code to designing complex distributed systems, are immersed in a digital environment. Code is our language, algorithms are our logic, and the development toolchain is our "digital workshop." Therefore, for software engineering majors, exploring "digital education" isn't just a technological embellishment to spice up existing teaching models. Rather, it's an inherent requirement and inevitable choice to return to the discipline's roots, adapt to industry trends, and achieve high-quality talent development.

However, in our own undergraduate teaching experience, we still perceive the tension between traditional education models and the talent demands of the digital age. On the one hand, we have unprecedented access to high-quality digital learning resources—a vast number of online courses (MOOCs)[1], open source communities like GitHub, advanced integrated development environments (IDEs), and cloud service platforms. On the other hand, some courses still rely on a linear process of "teacher lectures - student attendance - homework - final exam." Course content updates are slow to keep pace with the rapidly evolving technology stack, teaching cases are disconnected from real-world industry projects, and assessment methods focus on memorization of key knowledge points rather than comprehensive ability to solve complex engineering problems. This gap between "teaching" and "learning," and between "learning" and "application," is a crucial challenge that needs to be addressed in the pursuit of high-quality undergraduate education. Therefore, this "Educational Thought and High-Quality Development Discussion" provides us with a valuable opportunity to systematically reflect and examine the educational ecosystem in which we live. This article, drawing on the unique characteristics of the software engineering major, explores how digital education can become a key variable in resolving the aforementioned challenges and enabling high-quality development of undergraduate education. We will focus not only on the "artifacts" of technology but also on the profound changes in educational thought underlying it, exploring how to build a more open, collaborative, intelligent, and humane learning community, thereby truly cultivating exceptional software engineers who can adapt to and lead the future digital world.

2. The Logical Consistency of Software Engineering Majors Embracing Digital Education

Before we delve into how to apply digital education, we must first clearly define its connotation and deeply understand its special necessity for software engineering majors.

2.1. An Ecosystem beyond the Online Classroom

When it comes to digital education, many people's first thought may be the widespread adoption of live or recorded online classes during the pandemic. While this is undoubtedly an important component of digital education, it is far from the entire story. A truly digital education ecosystem is a data-driven, intelligent educational environment that encompasses the entire process of teaching, learning, management, evaluation, and research. It should encompass at least the following four core aspects:

(1) Resource Digitization

Transforming traditional teaching content, such as textbooks, handouts, and exercises, into digital resources that are accessible, searchable, interactive, and media-rich online. This includes not only course resources developed by schools and teachers themselves, but also global Open Educational Resources (OER)[2], such as MOOCs, academic databases, open source software projects, and technical documentation repositories, forming a massive, dynamic, and open knowledge graph.

(2) Process Digitization

Utilizing digital platforms and tools to record, manage, and support teaching interactions, learning behaviors, and project collaboration. For example, using a Learning Management System (LMS)[3] for course management and discussion, online programming platforms (such as Online Judge)[4] for real-time practice and feedback, and code hosting and collaboration platforms (such as GitHub/GitLab) for team project development.

(3) Assessment Digitization

Leveraging technology to achieve diversified, process-based, and intelligent assessment of learning outcomes and capabilities. This includes automated code review, duplication detection, and quality analysis, tracking and analyzing learning behavior data, building e-portfolios, and peer review-based collaborative assessment.

(4) Intelligent Environment

Building on the above foundation, we leverage big data and artificial intelligence technologies to provide personalized support and services for teachers and students. For example, we recommend personalized learning paths and resources for students, provide intelligent Q&A and tutoring, and offer teachers student analysis and teaching improvement suggestions, thereby achieving personalized and targeted teaching.

2.2. The "Native" Digital Education Attributes of the Software Engineering Major

If digital education empowers all majors, then for software engineering, it represents a more profound "return" and "fit." The inherent attributes of this major determine its high degree of homology with the concepts and practices of digital education.

(1) Rapid Iteration of Knowledge Systems

The software industry's technological advancements are measured in months or even weeks. New programming languages, frameworks, and architectural models emerge constantly. Traditional paper textbooks have a lengthy development cycle, from writing and reviewing to publishing, and their content is often partially outdated by the time they are released. Digital educational resources, such as official documentation, technical blogs, open source communities, and online courses, are inherently agile and can reflect cutting-edge technology in near real time. Accustoming students to learning from these dynamic digital sources from day one is key to fostering lifelong learning.

(2) Tool Dependence in the Practice Process

Software development itself is an engineering activity that relies heavily on a digital toolchain. From requirements analysis and design modeling (UML), coding (IDE), version control (Git), testing (automated testing frameworks), build and deployment (CI/CD), to project management (Jira/Trello)[5], the entire lifecycle is integrated with digital tools. Deeply integrating these mainstream digital tools into the teaching process goes beyond simply "teaching students how to use the tools"; it allows students to

immerse themselves in and master the concepts, methods, and processes of software engineering through the process of "putting the tools to work."

(3) Collaborative and Distributed Work Model

Modern large-scale software systems are rarely completed independently by individuals; rather, they are the product of global, distributed teamwork. The operating model of open source communities exemplifies this characteristic. Collaborative programming tools (such as VS Code Live Share)[6], code review mechanisms (Pull Requests), and asynchronous communication platforms (Slack/Discord) in digital educational environments can perfectly simulate and train students to adapt to this remote, distributed collaboration model, fostering the communication, collaboration, and community engagement skills essential for modern software engineers.

(4) Digitalization and Measurability of Outputs

Software engineering outputs—code, documentation, design drawings, test reports, and more—are inherently digital. This enables accurate, objective, and automated evaluation. Every code submission, every test pass, and every document update can be recorded, analyzed, and evaluated. This enables us to construct an evaluation system that is more comprehensive, objective, and process-oriented than traditional exams, accurately portraying students' capabilities across multiple dimensions, including knowledge acquisition, programming skills, and engineering literacy.

In summary, there is a natural and profound logical connection between software engineering and digital education. Promoting digital education within this major is not a cross-disciplinary integration, but rather a "journey home." It aims to break down the barriers between the classroom and the industry, theory and practice, and learning and creativity, bringing the educational process closer to the real industrial environment and laying a solid foundation for high-quality talent development.

3. Multi-Dimensional Exploration of Digital Education Empowering High-Quality Development of Software Engineering Teaching

Based on the understanding of the connotation and necessity of digital education, we can explore the practical path of digital education to empower the high-quality development of software engineering undergraduate teaching from multiple dimensions such as curriculum, resources, models, practices, and evaluation.

3.1. Curriculum Restructuring—From "Static Knowledge Map" to "Dynamic Competency Lego"

The traditional curriculum is like a fixed knowledge map, where students follow a pre-set path step by step. However, in the rapidly changing software industry, we need a "competency Lego" system that allows students to freely combine and rapidly iterate based on their interests and industry needs.

(1) Separation of Core and Frontier Knowledge, Dynamic Updates

With the help of digital platforms, course content can be broken down into two parts. One is the relatively stable "core knowledge modules," such as data structures, algorithms, operating systems, and computer networks. This part can utilize high-quality classic recorded courses or self-developed resources to ensure a solid foundation. The other part is the "frontier technology micro-modules," such as "Large-Scale Language Model Application Development," "Cloud Native and Containerization Technology," and "Responsive Front-End Framework Practice." These micro-modules can be jointly developed by on-campus faculty, corporate experts, and even outstanding alumni, and presented through

short videos, online workshops, and project-driven initiatives. Content can even be dynamically updated, replaced, and supplemented each semester.

(2) Personalized Learning Path Navigation

Schools can build a "software engineering competency map" to categorize the core competencies and technology stacks required for different roles within the industry (such as front-end development, back-end development, data scientists, and DevOps engineers). Based on their career plans, students can select different "micro-modules" to create their own personalized course packages, guided by their mentors and AI learning assistants. The LMS system records students' learning trajectories and intelligently recommends subsequent courses or supplementary resources based on their performance, enabling personalized training programs.

3.2. Teaching Resource Development—From "Isolated Lecture Notes" to a "Global Knowledge Center"

In traditional teaching, each teacher's PowerPoint presentations and lecture notes are often isolated "information islands." Digital education, on the other hand, strives to build an open, aggregated, and shared "Global Knowledge Center."

(1) Sharing and Contributing Within-School Resources

Establish a school- or department-level teaching resource repository to encourage teachers to upload their own high-quality courseware, code examples, exercise banks, and lab guides. These resources should be annotated with unified metadata to facilitate retrieval and reuse by other teachers and students. Through institutional incentives, promote collaboration among teaching teams across courses and grades to jointly develop high-quality digital resources.

(2) Effective Integration of Global Resources

Classrooms should not be confined to the campus. The teacher's role should shift from being "the sole transmitter of knowledge" to being "a guide for learning and an integrator of resources." For example, when teaching "Machine Learning," students could be assigned to watch Professor Andrew Ng's classic Coursera video course as a prep course. Classroom time should focus on answering questions about core concepts, practicing code, and discussing projects. When teaching "Open Source Software Development," the contribution guide, code repository, and community discussions of a well-known open source project (such as TensorFlow [7] or VS Code) can be directly used as core learning materials.

(3) Ready-to-use virtual lab environment

Many software engineering courses require complex development environment configuration, which often becomes a stumbling block for beginners. By leveraging virtualization and containerization technologies (such as Docker and Kubernetes [8]), schools can pre-build standardized cloud-based development environments for each course and experiment. Students can launch these environments with a single click in their browser, eliminating the need for tedious local installation and configuration, allowing them to focus on developing core knowledge and skills. This "cloud lab" not only lowers the learning barrier but also ensures a consistent teaching environment.

3.3. Innovation in Teaching Models: A Paradigm Shift from "Teacher-Centered" to "Student-Centered"

Technology is a catalyst; its true value lies in triggering a profound transformation in teaching models, achieving a shift from "teaching-centered" to "learning-centered."

(1) Deeply Integrated Blended Learning

This isn't simply the sum of online and offline learning, but rather a complementary approach. Online platforms are used for standardized knowledge transfer (such as watching videos and reading documents), while valuable offline classroom time is utilized for higher-level cognitive activities such as Project-Based Learning (PBL)[9], group discussions, code reviews, and teacher-student debates. For example, in a "Database Systems" course, students learn SQL syntax and paradigm theory online, while offline teams work together to design and implement a database for a small application system. These solutions are then presented, defended, and peer reviewed in class.

(2) Data-Driven Precision Teaching

Teachers can access detailed learning data in real time through LMSs and online programming platforms: student completion rates, pauses, and rewinds for each knowledge point video; accuracy rates and common incorrect answers for each question in online quizzes; code submission frequency, compilation error types, and test case passing times in programming assignments. By analyzing this data, teachers can pinpoint difficulties and student knowledge gaps, enabling targeted instruction and coaching in subsequent lessons or providing one-on-one intervention for students experiencing learning difficulties.

(3) AI-Powered Intelligent Learning Partners

Artificial intelligence, particularly large-scale language models, is becoming a powerful aid to learning. We can develop or introduce AI-embedded programming learning environments [10]. When students encounter compilation errors or logic problems, AI can act like a patient teaching assistant, providing multi-level prompts rather than direct answers. It can also assess the coding style and complexity of students' code and offer optimization suggestions. It can even act as an "interviewer," engaging in technical conversations with students to assess their deep understanding of concepts. This provides students with 24/7 instant and personalized tutoring, greatly improving their learning efficiency and independent learning ability.

3.4. Cultivating Practical Skills—Seamlessly Transforming "In-Class Experiments" into the "Real World"

Software engineering is an art of practice, and high-quality talent development must be centered on intensive, highly realistic practical training. Digital education provides an unprecedentedly broad platform for this.

(1) A "Digital Campus" [11] Centered on a Code Hosting Platform

Fully migrate teaching activities within a college or program to code hosting platforms such as GitHub and GitLab. Each course is a code repository, and each assignment is a commit or pull request. Students' entire academic career is a process of continuously accumulating code, participating in projects, and contributing to the community on the platform. This "code resume" is not only a complete record of their learning process but also the most intuitive and credible evidence of their technical proficiency, far more convincing than a transcript.

(2) Project-Driven Development

Starting in their freshman year, small projects based on real-world scenarios should be introduced. As students progress through the grade levels, the scale, complexity, teamwork requirements, and engineering level of these projects should gradually increase. For example, the freshman course "Programming Fundamentals" might require students to develop a small console game independently or in pairs; the sophomore course "Web Development" requires teams to develop a simple information

publishing website; and the junior course "Software Engineering" requires teams of approximately 10 to follow an agile development process and utilize a full engineering toolchain to develop a software product with real user value. These projects should be managed on a unified digital platform, and all processes (requirements, design, coding, testing, and deployment) should leave a digital footprint.

(3) Embrace Open Source

Encourage or even make participation in open source projects a requirement or bonus in some advanced courses. Instructors can guide students starting with fixing a simple documentation error (typo), gradually progressing to fixing a bug, and even contributing a new feature. By participating in open source, students not only gain access to the industry's most cutting-edge technologies and the highest-level code, but also learn standardized development processes, a rigorous engineering culture, and effective communication skills through interaction and collaboration with top engineers worldwide. This is a valuable experience that cannot be replaced by any classroom instruction.

3.5. Reforming the Evaluation System—Reshaping the Concept from "Summative Assessment" [12] to "Value-Added Assessment" [13]

Evaluation is a guiding principle; what is evaluated and how it is evaluated directly determine what and how students learn. Digital education supports us in building a more scientific, comprehensive, and humane evaluation system.

(1) Process-based Assessment, Supplementary

Students' final grades should be based on a comprehensive set of process-based indicators, with the weight of final exams significantly reduced or even eliminated. These process-based indicators may include: completion of programming assignments and code quality (assessed through automated testing and static code analysis tools); contributions to team projects (measured through Git commit history, task completion, peer reviews, and other metrics); active and in-depth participation in classroom and online community discussions; writing and sharing technical blogs; and contributions to open source communities.

(2) Competency Radar Charts Replace Percentile Scores

The final output of evaluation should not be a cold, hard number, but a multi-dimensional "competency radar chart." This chart clearly demonstrates students' proficiency in multiple sub-dimensions, including "algorithm design," "system architecture," "software testing," "teamwork," "technical communication," and "document writing." This not only allows students to clearly identify their strengths and weaknesses, but also provides valuable insights for targeted guidance from teachers and targeted recruitment by companies.

(3) Growth and Value-Added Assessment

The digital system records a student's complete learning trajectory from enrollment to graduation. Through longitudinal data analysis, we can assess a student's "value-added" across various competency dimensions—that is, how much they have improved over their four years of university compared to their initial foundation. This evaluation approach, which focuses on "growth" rather than "ranking," can better stimulate students' intrinsic motivation to learn and is more aligned with the essence of education.

4. Challenges and Prospects: Prudence and Expectations for Future Intelligent Education

While drawing up a grand blueprint for digital education, we must also be aware that this profound transformation is not an easy journey and still faces numerous challenges.

(1) The Difficulty of Deeply Integrating Technology and Education

Technology is a means, not an end. Avoiding "digitalization for its own sake," preventing teaching from being hijacked by flashy technology, and ensuring that technology truly serves educational goals, requires teachers to possess higher levels of digital literacy and instructional design skills. This requires schools to strengthen teacher training and support.

(2) The Digital Divide and Educational Equity

Different families' economic circumstances and information literacy levels can lead to new forms of inequality. Schools need to ensure that all students have equal access to digital devices and online resources and provide relevant information literacy training, ensuring that technology serves as a tool for promoting equity rather than widening disparities.

(3) Cultivating Students' Autonomous Learning Ability

Digital education offers a vast array of resources and a great deal of freedom, but it also places higher demands on students' independent learning, time management, and information sifting abilities. How to grant students freedom while providing necessary guidance and support to prevent them from getting lost in the ocean of information is a crucial issue.

(4) Risk of Lack of Humanistic Care and Emotional Connection between Teachers and Students

Over-reliance on online interactions may weaken face-to-face emotional exchange and intellectual exchange between teachers and students, and between classmates. High-quality education is not only about imparting knowledge, but also about guiding values and shaping character. Therefore, in the process of digitalization, it is necessary to consciously design and preserve high-quality offline interactions to ensure that education retains its proper warmth.

Looking ahead, with the further development of technologies such as Artificial Intelligence (AI), Virtual Reality (VR/AR) [14], and big data, digital education will evolve into a higher-level form—intelligent education. Future learning scenarios may include: students donning VR glasses and immersing themselves in the three-dimensional architecture of a large-scale software system; AI learning partners analyzing students' brainwaves and eye movement data in real time, gaining insights into their cognitive load and attention span, and dynamically adjusting teaching strategies; learning will no longer be confined to the campus but will be integrated into a lifelong learning network seamlessly connected to society and industry.

5. Conclusion

"Focusing on digital education and empowering high-quality development of undergraduate education" is a systematic project and a profound revolution in educational thought. As software engineering students, we are not only beneficiaries of this transformation, but we should also be active participants and promoters. We hope that this discussion will build consensus and propel schools and colleges to take more solid steps on the path of digital education.

This isn't just about introducing more online platforms or deploying more smart devices; it's about redefining "learning" itself. It requires us to view learning as a creative process of active construction, continuous exploration, and extensive collaboration. It requires our teachers to transform from "sages on the podium" to "guides at our side," and our students to transform from "containers of knowledge" to "active explorers."

The road may be long, but if we keep walking, we will reach our destination; the task may be difficult, but if we persist, we will succeed. We firmly believe that as long as we insist on focusing on student

development, aiming at cultivating innovative talents, and guided by the transformation of educational thinking, and prudently and firmly advancing the digital education strategy, we will be able to inject new vitality into the undergraduate teaching of software engineering and even the entire engineering discipline, achieve high-quality development in the true sense, cultivate outstanding engineers worthy of the times, and contribute the wisdom and strength of our generation to my country's scientific and technological self-reliance and the construction of a digital power.

References

[1] Baturay MH. An overview of the world of MOOCs. Procedia-Social and Behavioral Sciences. 2015 Feb 12;174:427-33.

[2] Butcher N. Basic guide to open educational resources (OER).

[3] Bradley VM. Learning Management System (LMS) use with online instruction. International Journal of Technology in Education. 2021;4(1):68-92.

[4] Kurnia A, Lim A, Cheang B. Online judge. Computers & Education. 2001 May 1;36(4):299-315.

[5] Cheverda SS. Analysis of project management software. Financial Strategies of Innovative Economic Development. Збірник наукових праць входить до міжнародної індексації. 2022:48.

[6] Tan X, Lv X, Jiang J, Zhang L. Understanding real-time collaborative programming: a study of visual studio live share. ACM Transactions on Software Engineering and Methodology. 2024 Apr 20;33(4):1-28.

[7] Pang B, Nijkamp E, Wu YN. Deep learning with tensorflow: A review. Journal of Educational and Behavioral Statistics. 2020 Apr;45(2):227-48.

[8] Bernstein D. Containers and cloud: From lxc to docker to kubernetes. IEEE cloud computing. 2014 Sep 30;1(3):81-4.

[9] Kokotsaki D, Menzies V, Wiggins A. Project-based learning: A review of the literature. Improving schools. 2016 Nov;19(3):267-77.

[10] Tawfik E, Ali S, Yahia Y. Towards Evaluating The AI-Embedded Educational Model for Architecture Students and Its Impact on Their Design Outcomes. Delta University Scientific Journal. 2024 Nov 23;7(3):154-73.

[11] Galustyan OV. Digital Campus as Electronic Image of the University. Rupkatha Journal on Interdisciplinary Studies in Humanities. 2015 Sep 1;7(3):263-70.

[12] Knight PT. Summative assessment in higher education: practices in disarray. Studies in higher Education. 2002 Aug 1;27(3):275-86.

[13] Amrein-Beardsley A. Methodological concerns about the education value-added assessment system. Educational researcher. 2008 Mar;37(2):65-75.

[14] Li X, Yi W, Chi HL, Wang X, Chan AP. A critical review of virtual and augmented reality (VR/AR) applications in construction safety. Automation in construction. 2018 Feb 1;86:150-62.