ISSN 2522-6398 Vol. 7, Issue 12: 35-42, DOI: 10.25236/FER.2024.071205

Research on Strategies and Practices for Reforming Computer Programming Language Teaching in Undergraduate Education Based on Project-Driven Approach

Hairong Zhu[1], Xiaoqiang Luo[1,*], Zhuo Feng[1], Yilan Zeng[2]

[1]Guangxi Normal University of Science and Technology, Laibin, Guangxi, 546199, China
[2]The Second Junior High School of Nanmu Town, Guigang City, Guangxi, 537226, China
[*]Corresponding author

Abstract: *This paper explores the strategies and effectiveness of promoting programming language teaching reform in undergraduate computer science education using a project-driven approach. Through a comparative analysis of project-driven and traditional teaching methods, the advantages of project-driven teaching in stimulating student interest, enhancing practical skills, and promoting the development of comprehensive competencies are demonstrated. Using specific case studies, the paper illustrates how project-driven teaching can be used to restructure programming language courses, broaden student perspectives, improve practical application skills, enhance teamwork, and foster innovative thinking. Finally, the paper proposes a complete teaching reform framework from the perspectives of teaching philosophy, course design, teaching methods, and evaluation systems, with the aim of providing valuable insights for the reform of programming language teaching in undergraduate computer education.*

Keywords: *Programming language teaching; comprehensive competencies; project-driven learning; practical teaching methods; teamwork skills*

1. Introduction

1.1 Research Background and Significance

The rapid development of computer science and technology presents new challenges for undergraduate computer education, particularly in programming language instruction. Traditional programming language teaching models often lack support for practical applications and project-based learning, leading to low student engagement and insufficient hands-on and practical skills[1]. Therefore, reforming programming language teaching models has become an urgent task.

The project-driven teaching model emphasizes practice, interdisciplinary integration, and student-led learning. Through team-based projects, students develop problem-solving, innovation, and collaboration skills. Introducing project-driven teaching into undergraduate programming language education is expected to boost student motivation and practical abilities, nurture comprehensive competencies, and foster innovation[2]. This reform holds significant implications for improving teaching quality, preparing students for employment, and aligning computer science education with the evolving needs of the IT industry.

The main significance of the current programming language teaching reform in computer science includes: enhancing teaching quality, sparking student interest, and cultivating practical application skills; aligning with societal demands to support student career development[3]; and promoting the integration of higher education with the IT industry. Therefore, research on project-driven strategies and practices for programming language teaching reform is theoretically and practically significant for advancing computer science education, enhancing the quality of higher education, and cultivating well-rounded, innovative, and application-oriented talents[4].

1.2 Current State of Research and Development Trends

Research on programming language teaching reform in undergraduate computer science education, both domestically and internationally, presents a diverse and upward trend.

Internationally, the project-driven teaching model has been widely applied in computer science education with notable success[5]. For example, Harvard University's CS50 course (Introduction to Computer Science) employs a project-driven approach, using practical projects and hands-on exercises to teach programming languages[6], attracting large student participation and achieving excellent outcomes. Additionally, European institutions, such as the University of Cambridge in the UK and the Technical University of Munich in Germany, are also actively exploring project-driven teaching methods in undergraduate computer science education[7].

In China, with the continuous development of computer science education, the project-driven teaching model has gained attention from scholars and educators[3]. Some universities and educational institutions have begun incorporating project-driven teaching into programming language courses, achieving positive results[8]. For example, leading universities such as Tsinghua University and Peking University have undertaken teaching reforms in their computer science departments, adopting project-driven models and seeing some success[9].

Both domestic and international research on programming language teaching reform is progressing in a diverse and positive direction, with future trends likely to deepen reforms[10], improve teaching quality, and cultivate more high-quality computer science professionals.

2. Application of Project-Driven Teaching in Programming Language Instruction

The project-driven teaching model is primarily based on constructivist learning theory and project-based learning theory. Constructivism holds that learning is an active process of constructing knowledge, where students engage in real-world projects and solve practical problems to build their knowledge framework. Project-based learning theory emphasizes that learning through real projects is more effective than traditional classroom learning because projects provide concrete contexts and challenges that motivate students, foster deeper understanding, and develop practical application skills.

In programming language instruction, project-driven teaching is a highly effective approach. Students design, develop, and implement real projects to deepen their understanding of programming languages while cultivating problem-solving and teamwork skills. For example, students can learn programming languages by developing web applications, game programs, or solving real-life problems, which is both engaging and a good exercise for programming skills. Project-driven teaching also promotes innovative thinking and hands-on skills, laying a solid foundation for students' future career development.

2.1 Course Design for Project-Driven Programming Language Teaching

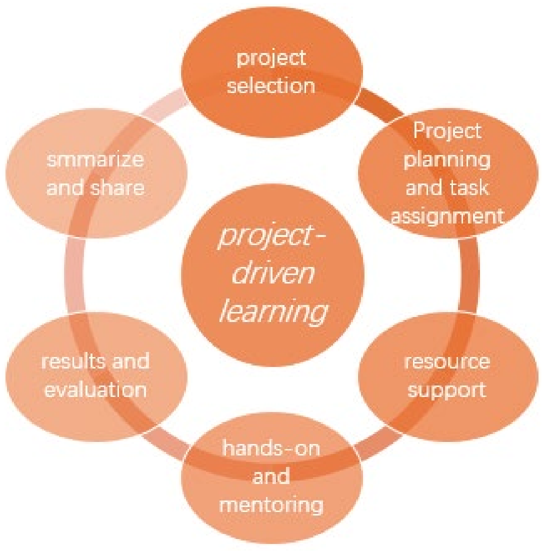


Figure 1: Project-Driven Teaching Framework

The course design in a project-driven teaching model provides a rich teaching experience and deep learning outcomes. Below is a process for designing a project-driven programming language course. As shown in Figure 1, students' abilities are developed in six iterative steps, each reinforcing the previous ones.

2.1.1 Topic Selection and Project Choice:

- First, tutors choose a suitable programming language based on students' learning levels and interests, such as Python, Java, or C programming under Linux.

- Then, tutors determine one or more appropriate project topics, which could involve simple game development, website design, data analysis, or machine learning. Project selection should meet students' actual needs and future career goals.

2.1.2 Project Planning and Task Assignment:

- Students develop a detailed project plan and schedule for the selected project, including goals, milestones, and deadlines.

- Tutors Assign tasks based on project complexity and student abilities, allowing students to take on various roles such as project manager, developer, or tester, promoting teamwork and communication skills.

2.1.3 Teaching Methods and Resource Support:

- Tutors use a variety of teaching methods, such as lectures, case studies, hands-on exercises, and group discussions, tailored to students' needs and learning characteristics to boost engagement.

- Tutors provide ample resources, including video tutorials, online learning materials, and coding tools, to support student autonomy and practice.

2.1.4 Practical Exercises and Guidance:

- Tutors encourage students to engage in practical exercises, writing code to implement project functionalities and features, which deepens their understanding of programming languages.

- Tutors offer ample guidance and support to help students overcome challenges and enhance their confidence and learning outcomes.

2.1.5 Project Presentation and Evaluation:

- After project completion, tutors organize presentations where students demonstrate their project results and share their learning experiences.

- Tutors use diverse evaluation methods, including project reports, code reviews, and peer assessments, to comprehensively assess students' outcomes, learning progress, and teamwork skills.

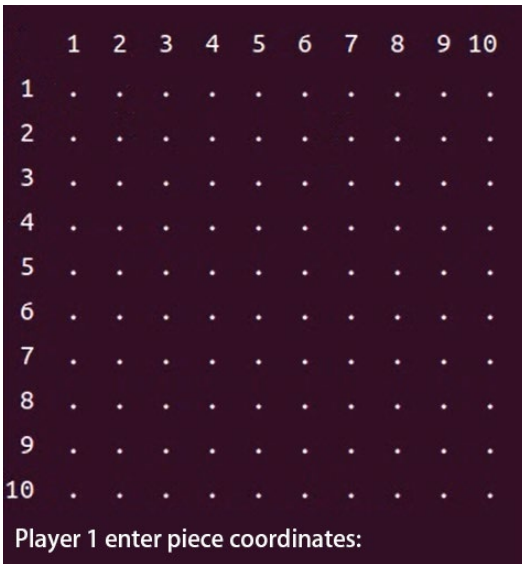
2.1.6 Reflection and Experience Sharing:

- Tutors guide students to reflect on their project experiences, identifying strengths and areas for improvement, gaining insights for future projects.

- Tutors encourage students to share their experiences and insights with peers, promoting mutual learning and progress.

Through this course design, project-driven programming language courses can effectively engage students, cultivate problem-solving abilities, and foster teamwork. This enables students to gain a deeper understanding of programming language applications and hands-on experience, laying a solid foundation for their future careers.

2.2 Case Studies and Evaluation of Teaching Outcomes



```
    1  2  3  4  5  6  7  8  9  10
1   .  .  .  .  .  .  .  .  .  .
2   .  .  .  .  .  .  .  .  .  .
3   .  .  .  .  .  .  .  .  .  .
4   .  .  .  .  .  .  .  .  .  .
5   .  .  .  .  .  .  .  .  .  .
6   .  .  .  .  .  .  .  .  .  .
7   .  .  .  .  .  .  .  .  .  .
8   .  .  .  .  .  .  .  .  .  .
9   .  .  .  .  .  .  .  .  .  .
10  .  .  .  .  .  .  .  .  .  .
Player 1 enter piece coordinates:
```

Figure 2: Gomoku Game Interface in Linux Terminal

2.2.1 Case Study: Gomoku Game Implementation (C Language)

Project Description: Students are tasked with using C language to implement a simple Gomoku game in the Linux terminal under Ubuntu. The game should display the board in the terminal, allow two players to take turns placing pieces, determine the winner, and display the result after the game ends. The GUI is shown in Figure 2.

Implementation Steps:

a) Project Planning and Design: Students analyze game rules and requirements, design the board display and game logic, decide on data structures (e.g., a 2D array to represent the board), and design the logic for user input and game rule enforcement.

b) Board Display: Students use terminal output functions to display the board, using ASCII art to show the board with markers for empty spaces and placed pieces.

c) Game Logic: Students write logic functions to handle piece placement, victory conditions, and restricted moves, ensuring proper game rule enforcement.

d) User Input: Students write input functions to accept player input for move coordinates, check for valid input, and prevent illegal moves such as out-of-bound or duplicate placements.

e) Game Loop: Students implement the game’s main loop, allowing players to take turns until the game concludes.

f) Game End: Students implement a function to determine the winner and display the result after the game concludes.

2.2.2 Teaching Outcome Evaluation:

a) Code Completeness: Assess whether students fully implemented the Gomoku game according to the rules, including board display, game logic, and user input.

b) Code Quality: Evaluate the clarity of the code structure, variable naming, and adequacy of comments.

c) Technical Proficiency: Assess students' mastery of C programming and Linux terminal operations, including syntax, file handling, and terminal commands.

d) Game Functionality: Evaluate whether students successfully implemented the game's core functions, such as board display, move validation, and victory detection.

e) User Experience: Test the game’s usability and interactivity, assessing its ease of use, smooth operation, and interface design.

f) Self-Learning and Improvement: Encourage students to reflect on the project experience, evaluate their learning progress, and consider areas for improvement, such as a deeper understanding of C programming and Linux proficiency.

Through the above teaching effect evaluation, students 'learning situation and results in the Gobang game project can be comprehensively evaluated, and further learning and improvement direction can be provided for them.

3. Project-Driven Programming Language Teaching Reform Framework

The project-driven programming language teaching reform framework includes the following key elements: first, determine the project goals and scope, clearly define the learning objectives and project requirements; next, design project tasks and practical activities, integrating real-world cases and application scenarios to stimulate student interest; then, form student teams to cultivate teamwork and communication skills; after that, guide students through project implementation, providing technical support and mentorship; and finally, evaluate project outcomes, give feedback on student performance, and promote self-improvement. This framework emphasizes practice, collaboration, and feedback, helping to improve students' programming skills and overall competencies.

3.1 Teaching Philosophy and Goals

In project-driven programming language teaching reform, the teaching philosophy and goals are of critical importance. This teaching method advocates a student-centered approach, emphasizing their autonomy and practical skills. Project practice is considered an effective way for students to understand and master core concepts and skills in programming languages, making project-driven teaching a highly recommended learning approach.

This teaching method aims to cultivate students' diverse skills and comprehensive abilities. It not only focuses on the syntax and logic of programming languages but also emphasizes the development of students' innovative thinking, problem-solving abilities, and teamwork skills. By participating in real projects, students face practical challenges and develop their problem-solving skills and thinking patterns, thus becoming competitive professionals in computer science.

This teaching method is closely aligned with industry demands and societal development. It aims to train individuals with practical application skills and an innovative mindset to adapt to the rapidly changing IT industry and emerging technologies. Through project practice, students continuously explore, learn, and grow, contributing to societal development and becoming outstanding professionals capable of addressing various challenges.

This teaching method also aims to promote lifelong learning and personal development for students. Through the project-driven teaching model, students not only master current programming skills but also cultivate their learning ability and awareness of self-improvement, thereby achieving career development and value realization.

The project-driven programming language teaching approach emphasizes a student-centered focus, fostering diverse skills and comprehensive abilities, closely integrating with industry demands and societal development, and promoting lifelong learning and self-development for students. This approach provides effective support and guidance for students' growth and development and is worth further research and practice.

3.2 Course Design and Implementation

Course design and implementation are critical components of project-driven programming language teaching. In this section, we explore the course design and implementation based on C programming under the Ubuntu Linux system.

3.2.1 Course Design

a) Teaching Objectives

- Master basic C language syntax and logic: Students should understand the basic syntax structures, data types, and operators in C and grasp basic programming logic.

- Practice project-driven learning: Through project practice, students apply learned knowledge to solve real-world problems, developing practical skills and innovative thinking.

- Deepen understanding of the Linux system: By programming under Ubuntu, students gain an in-depth understanding of the principles and characteristics of the Linux system, enhancing their understanding and application of operating systems.

b) Course Content

- Basic syntax and programming practice: Includes basic topics like data types, control flow statements, functions, and pointers, consolidated through small-scale programming exercises.

- File operations and system calls: Introduces file operation functions in the Linux system, such as ‘open’, ‘read’, and ‘write’, as well as the basic principles of system calls.

- Multithreaded programming: Covers basic concepts and functions related to multithreaded programming, with project practice exploring its role in real applications.

- Network programming: Introduces the basics of socket programming and network communication, with practical projects exploring the application of network programming in real-world scenarios.

c) Teaching Methods

- Project-driven approach: Projects are central to learning, guiding students to solve practical problems and stimulating their interest and hands-on abilities.

- Case studies: Case studies are used to help students deeply understand the learned knowledge and apply it to real-world problem-solving, fostering problem-solving skills.

- Practical operations: Focuses on students' practical skills, reinforcing and applying learned knowledge through programming exercises and project practice to develop their application capabilities.

3.2.2 Course Implementation

a) Teaching Resources

- Software environment: Provide an Ubuntu operating system environment and a C language compiler to offer students a conducive programming environment.

- Textbooks and materials: Choose suitable textbooks and resources, and combine them with online materials to provide students with abundant learning and reference materials.

- Laboratory facilities: Provide lab facilities to support students in conducting experiments and project practice, ensuring smooth implementation of the teaching process.

b) Teaching Organization

- Classroom teaching: Combine theoretical explanations with demonstrations to introduce students to basic concepts and programming techniques in C language.

- Experimental practice: Arrange experimental sessions where students engage in programming practice in the lab, reinforcing learned knowledge.

- Project guidance: Provide project mentorship and guidance, helping students complete project tasks and cultivating their practical skills and teamwork abilities.

c) Teaching Evaluation

- Exams and assessments: Evaluate students' understanding of theoretical knowledge through exams and assignments.

- Project assessment: Assess students based on project outcomes and practical skills, measuring their application abilities and innovative thinking.

- Feedback and improvement: Collect student feedback in a timely manner and make targeted improvements to teaching methods and content to enhance teaching effectiveness.

By following the above course design and implementation, students' programming abilities, practical skills, and innovative thinking can be effectively enhanced, laying a solid foundation for their future career development.

4. Conclusion and Future Prospects

This study explores a project-driven programming language teaching reform framework, emphasizing the enhancement of students' programming skills and overall competencies through practice, collaboration, and feedback. The proposed teaching philosophy and goals are student-centered, aiming to develop diverse skills and comprehensive abilities to enable students to meet the fast-changing demands of the IT industry. This teaching model not only focuses on fundamental knowledge of programming languages but also highlights students' innovation, problem-solving, and teamwork abilities. By engaging in real projects, students hone their thinking skills and improve their ability to tackle real-world challenges.

In terms of course design and implementation, using the C programming language under Ubuntu as an example, we defined teaching objectives, course content, and teaching methods. Through modules covering basic syntax, file operations, system calls, multithreading, and network programming, the course content comprehensively addresses core programming knowledge, while project practice enhances students' hands-on capabilities. This project-driven teaching method not only motivates students but also significantly improves their practical application skills.

Effective allocation of teaching resources and scientific arrangement of teaching organization are key to ensuring the successful implementation of the course. Providing a suitable software environment, rich learning materials, and well-equipped lab facilities gives students strong support for learning and practice. The combination of classroom teaching, experimental practice, and project guidance forms a systematic teaching process that helps students grasp knowledge and improve skills.

In terms of teaching evaluation, we suggest combining exams with project evaluations to comprehensively assess students' theoretical knowledge and practical skills. Additionally, timely collection of student feedback and targeted improvements to teaching methods will continuously optimize teaching effectiveness.

Looking ahead, the project-driven programming language teaching reform framework has broad application prospects. In an era of rapid IT development, educators need to continuously update teaching content and integrate emerging technologies and industry trends to keep courses practical and forward-looking. Future research can further explore how to incorporate more interdisciplinary elements into project-driven teaching to enhance students' comprehensive skills.

With the development of online education and blended learning models, the combination of project-driven teaching methods with digital tools can be explored to create new teaching formats and evaluation mechanisms that better meet the needs of different learners. Through continued research and practice, the project-driven programming language teaching reform framework is poised to contribute to the cultivation of high-quality computer science professionals who can adapt to future societal developments.

Acknowledgments

This work was supported by Project for Enhancing Young and Middle-aged Teacher's Research Basis Ability in Colleges of Guangxi (Gui Ke Education Research [2021] No.1) 2021KY0853, 2021KY0859; Guangxi Zhuang Autonomous Region-level College Student Innovation and Entrepreneurship Training Program (Gui Jiao Gaojiao [2024] No.36) X2024114, S202411546068; Guangxi Science and Technology Normal University Undergraduate Research Fund Project GXKS2022DXSA007, GXKS2022DXSB017, GXKS2021DXSB011, GXKS2023DXS015, GXKS2023DXS017, GXKS2024YB047.

References

[1] Chen, C., Chen, T., Huang, Y., & Wu, J. (2020). Effects of Project-Based Learning on Students' Programming Performance and Motivation in Computer Science Education. Journal of Educational Technology & Society, 23(3), 51-62.
[2] Su, S., Liu, M., & Han, X. (2020). A Review of Project-Based Learning in Computing Education: Recent Advances and Future Directions. IEEE Transactions on Education, 63(4), 350-362.
[3] König, J., Jütte, L., & Eichler, D. (2021). Project-Based Learning in Higher Education: A Systematic Literature Review. Educational Research Review, 33, 100396.
[4] Liao, Y., Shum, S. B., & Chu, S. K. (2021). Effects of Project-Based Learning in STEM Education: A Meta-Analysis. Educational Psychology Review, 33(2), 285-317.

[5] Hwang, G. J., Lai, C. L., & Wang, S. Y. (2020). Effects of the Flipped Classroom Model on Knowledge Engagement and Project Performance in Programming Education. Journal of Computer Assisted Learning, 36(6), 801-815.
[6] Du, Y., Lin, J., & Fang, H. (2020). Integrating Project-Based Learning and Online Learning in Computer Science Education: An Empirical Study. Computers & Education, 157, 103978.
[7] Suh, S., Lee, J., & Kim, J. (2020). Exploring the Impact of Project-Based Learning on Programming Achievement in Elementary School. Computers & Education, 154, 103944.
[8] Baepler, P., & Walker, J. D. (2021). The Value of Authenticity for Project-Based Learning in Higher Education: A Review. Educational Psychology Review, 33(3), 805-835.
[9] Yıldırım, B., & Çolak, İ. (2021). Investigating the Effects of Project-Based Learning on Academic Achievement and Self-Efficacy Beliefs in Programming Education. Journal of Educational Computing Research, 59(6), 1197-1218.
[10] Wang, Y., & Huang, Y. M. (2022). Enhancing Programming Learning through Project-Based Learning: A Meta-analysis of Empirical Studies. Computers & Education, 176, 104481.