Journal of International Education and Development
2025, VOL. 9, NO. 12, 77-80
DOI: 10.47297/wspiedWSP2516-250017.20250912

A Teaching Reform Practice for Enhancing Creative and Analytical Thinking in Software Engineering Courses

Chong Zeng, Jiali Chen

Longyan University, Longyan, Fujian, 364012, China

ABSTRACT
This study reports a teaching reform practice implemented in a Year 3 Software Engineering course, aiming to support students' creative and analytical thinking during software design tasks. In software engineering education, students often demonstrate strong coding abilities but experience difficulties in generating diverse design ideas and systematically refining them into feasible solutions. To address this challenge, a structured teaching model integrating divergent and convergent thinking was introduced into project-based learning activities. The proposed model consists of three stages: Imagining, Linking, and Inference. The Imagining stage encourages divergent thinking through open-ended design activities; the Linking stage supports guided reflection and learning motivation; and the Inference stage focuses on convergent thinking through feasibility analysis and design justification. The model was applied across several teaching weeks in a Year 3 Software Engineering course. Classroom observations and teaching reflections indicate that the approach helped students develop clearer thinking structures, improved their ability to justify design decisions, and enhanced engagement in design discussions. This study demonstrates the value of structured thinking training in software engineering education and provides a transferable teaching model for similar practice-oriented courses.

KEYWORDS
Software engineering education; Teaching reform; Creative thinking; Analytical thinking; Project-based learning

1 Introduction

Software Engineering education aims to cultivate not only technical proficiency but also the ability to design innovative, justified, and adaptable solutions for complex problems . While many undergraduate programs emphasize programming accuracy and process compliance, students often struggle with open-ended design tasks that require both creative exploration and systematic reasoning.

In Year 3 Software Engineering courses, students are typically expected to undertake more complex and less structured tasks, such as system-level design and project-based development. However, classroom experience indicates that many students struggle to generate diverse design ideas and tend to converge prematurely on familiar solutions. As a result, design decisions are sometimes made without sufficient exploration of alternatives or clear justification, limiting both creativity and the quality of final solutions.

This challenge highlights a common tension in software engineering education: how to balance creative exploration with analytical rigour. On the one hand, students need opportunities to explore multiple possibilities and think creatively; on the other hand, they must learn to evaluate options systematically under practical constraints. Without explicit instructional support, students may perceive creativity and analytical reasoning as conflicting rather than complementary processes.

To address this issue, this paper presents a teaching reform practice that introduces a structured teaching model to support students' transition from divergent thinking to convergent thinking during software design activities. The model integrates three stages, Imagining, Linking, and Inference, into regular teaching practice, guiding students from idea generation through reflection to reasoned design justification. By making the thinking process explicit and structured, the approach aims to enhance students' creative engagement while maintaining the analytical standards required in software engineering education.

2 Teaching Context and Objectives

2.1 Teaching Context

The teaching reform was implemented in a Year 3 undergraduate Software Engineering course that forms part of a computing degree programme. The course focuses on core software engineering concepts, including requirement analysis, system design, and solution evaluation, and is delivered through a combination of lectures, tutorials, and project-based learning activities. Students are typically required to work in small groups to propose, design, and justify software solutions for realistic problem scenarios.

* **Corresponding Author:** Jiali Chen, chenjiali@lyun.edu.cn

At this stage of study, students have already acquired foundational programming skills and basic knowledge of software development processes. However, many students have limited experience in handling ill-defined design problems that allow for multiple valid solutions. In previous teaching iterations, students often approached design tasks with a strong focus on implementation details while giving insufficient attention to idea exploration and design rationale.

The course therefore provides an appropriate context for introducing structured support for creative and analytical thinking. By embedding the teaching model within existing project-based activities, the reform was designed to enhance students' design thinking without significantly altering course content or assessment requirements.

2.2 Teaching Objectives

The primary objectives of the teaching reform are as follows:

To encourage divergent thinking during the early stages of software design, enabling students to explore a wider range of possible solutions.

To support students in transitioning from idea exploration to structured evaluation through guided reflection and discussion.

To enhance students' ability to justify design decisions based on feasibility, requirements, and practical constraints.

These objectives align with the learning outcomes of the Software Engineering course and address observed challenges in students' design thinking processes. Rather than replacing existing teaching methods, the proposed model complements traditional instruction by providing a clear structure for managing creativity and analysis within software design tasks .

3 Teaching Model Design

The proposed teaching model is designed to support students in transitioning from divergent thinking to convergent thinking during software design tasks. In traditional software engineering teaching, students are often required to produce a single "correct" solution within a limited timeframe, which may restrict creative exploration and result in premature decision-making. In contrast, this model introduces a structured three-stage process consisting of Imagining, Linking, and Inference to guide students through idea generation, reflection, and rational evaluation in a systematic manner.

The three stages correspond to different cognitive focuses within the design process. The Imagining stage emphasises divergent thinking and idea exploration; the Linking stage provides an intermediate step that supports reflection, motivation, and selection; and the Inference stage concentrates on convergent thinking through feasibility analysis and design justification. By explicitly separating these stages, the model helps students recognise that creativity and analytical reasoning are complementary rather than conflicting processes in software engineering design.

Stage 1: Imagining-Supporting Divergent Thinking

The Imagining stage aims to encourage divergent thinking in the early phase of software design. At this stage, students are invited to explore a wide range of possible solutions without immediate evaluation or judgement. Typical classroom activities include brainstorming sessions, open-ended design prompts, and group discussions focused on generating multiple ideas for addressing a given software problem.

During this stage, students are explicitly informed that there are no "wrong answers" and that the quantity and diversity of ideas are prioritised over correctness. This approach helps reduce students' anxiety about making mistakes and encourages them to move beyond familiar or conventional solutions. For Year 3 Software Engineering students, who often have strong technical habits, this stage provides an opportunity to rethink problems from multiple perspectives before committing to a specific design direction.

The teacher's role at this stage is primarily facilitative rather than evaluative. Instead of guiding students toward a predefined solution, the teacher encourages exploration by posing open questions, highlighting alternative viewpoints, and ensuring that all group members contribute to the discussion. All generated ideas are recorded for later use, forming the basis for subsequent reflection and selection.

Stage 2: Linking – Guided Reflection and Learning Motivation

The Linking stage serves as a transitional phase between divergent and convergent thinking. Its primary purpose is to support students in reflecting on the ideas generated during the Imagining stage and identifying those that are worth further development. Rather than relying on technical feasibility alone, this stage emphasises guided reflection, peer discussion, and learning motivation.

In classroom practice, students review their generated ideas through structured reflection questions, such as which ideas they find most interesting, which align best with project goals, and which they feel motivated to develop further. Peer discussions are encouraged to help students articulate their preferences and understand alternative viewpoints. Through this process, students begin to form a sense of ownership over their design choices, which supports sustained

engagement in later stages of the project.

This stage does not aim to eliminate creativity but to channel it in a purposeful direction. By linking personal interest with project relevance, students are better prepared to invest effort in refining their ideas. The teacher provides guidance by prompting reflection and helping students recognise the value of aligning motivation with learning objectives, rather than imposing direct selection criteria.

Stage 3: Inference – Convergent Thinking and Design Validation

The Inference stage focuses on convergent thinking and the rational evaluation of selected design ideas. At this stage, students are required to refine their chosen solutions through feasibility analysis, requirement alignment, and design justification. This process reflects core practices in software engineering, where decisions must be supported by logical reasoning and practical constraints.

Students evaluate their designs based on factors such as system requirements, technical feasibility, scalability, and potential trade-offs. Group discussions and presentations are used to encourage students to explain and justify their decisions, fostering analytical thinking and communication skills . Through this process, students learn to move from intuitive ideas to well-reasoned design solutions that can withstand critical questioning.

The teacher's role in this stage is more evaluative and supportive, providing feedback on students' reasoning processes and highlighting strengths and weaknesses in their justifications. By completing the three-stage process, students experience a structured transition from creative exploration to practical decision-making, reinforcing the idea that creativity and analytical rigour are both essential components of effective software engineering design.

4 Classroom Implementation and Observation

The proposed teaching model was implemented in a Year 3 Software Engineering course over several teaching weeks as part of a project-based learning component. The course required students to work in small groups on software design tasks, including problem analysis, solution proposal, and design justification. The three-stage teaching model, Imagining, Linking, and Inference, was integrated into regular teaching activities rather than introduced as a separate or additional module.

During the Imagining stage, students were asked to explore multiple possible solutions to a given software problem before making any design decisions. Brainstorming sessions and open-ended design prompts were incorporated into classroom discussions. Students were encouraged to generate as many ideas as possible and to record them collectively within their groups. Classroom observation indicated that students initially tended to propose familiar or technically straightforward solutions; however, with repeated practice, they gradually began to explore a wider range of alternatives and demonstrated greater willingness to suggest unconventional ideas.

The Linking stage was implemented through guided reflection and peer discussion. After idea generation, students reviewed their proposed solutions and discussed which ideas they felt motivated to develop further and which aligned most closely with project objectives. Reflection questions were provided by the teacher to support this process, such as asking students to explain why certain ideas attracted their interest or how different ideas might contribute to the overall system design. Informal observation suggested that this stage helped students clarify their design intentions and increased their engagement in group discussions.

In the Inference stage, students refined their selected ideas through feasibility analysis and design justification. They were required to explain their design choices in relation to system requirements, technical constraints, and potential trade-offs. Group presentations and design reviews were used to support this process. Compared with earlier teaching iterations, students appeared more confident in articulating their reasoning and were better able to explain why particular solutions were chosen over alternatives .

Overall, classroom observations and informal student feedback indicated that the structured three-stage process helped students develop clearer thinking paths during software design tasks. Students showed improved ability to move from idea exploration to reasoned decision-making, and classroom discussions became more focused and purposeful. While the implementation did not involve formal quantitative evaluation, the observed changes in student engagement and design reasoning suggest that the teaching model provided effective support for creative and analytical thinking in software engineering education.

5 Teaching Reflection and Discussion

Reflecting on the classroom implementation, several positive outcomes were observed. One notable benefit of the teaching model was its ability to make the thinking process explicit for students. By separating idea generation, reflection, and evaluation into distinct stages, students gained a clearer understanding of how creative exploration and analytical reasoning can coexist within software engineering design. This structure helped reduce students' tendency to rush toward a single solution and encouraged more thoughtful consideration of alternatives.

The model also contributed to improved student engagement. The Linking stage, in particular, supported students in

developing a sense of ownership over their design choices. By allowing students to reflect on their interests and motivations before committing to a solution, the teaching approach helped sustain participation throughout later design and validation activities. Students appeared more willing to contribute to discussions and to defend their ideas when asked to justify design decisions.

Despite these positive outcomes, several challenges were identified. Some students expressed discomfort with the open-ended nature of the Imagining stage, indicating a need for clearer prompts and time guidance in early design activities. Time management also emerged as a practical concern, as allocating sufficient time for each stage within regular teaching sessions required careful planning.

Another limitation of the current implementation is that observations were primarily qualitative and based on teaching reflection rather than formal assessment measures. While this approach is appropriate for a practice-oriented teaching reform study, future work could incorporate more systematic evaluation methods, such as reflective journals or structured peer feedback, to further examine learning outcomes .

In summary, the teaching reform demonstrated that a structured transition from divergent to convergent thinking can effectively support Year 3 Software Engineering students in software design tasks. The experience highlights the importance of balancing creativity with analytical rigour and suggests that structured thinking models can play a valuable role in enhancing software engineering education. Future refinements of the model will focus on improving scaffolding strategies and exploring its applicability to other practice-oriented computing courses.

6 Conclusion

This paper has presented a teaching reform practice implemented in a Year 3 Software Engineering course, focusing on supporting students' transition from divergent thinking to convergent thinking during software design activities. By introducing a structured three-stage teaching model, Imagining, Linking, and Inference, the approach provides explicit guidance for managing creative exploration and analytical reasoning within the software engineering design process.

Classroom implementation and teaching observation suggest that the model helped students develop clearer thinking structures, encouraged broader idea exploration, and improved their ability to justify design decisions under practical constraints. By separating idea generation, reflection, and evaluation into distinct stages, students were better able to understand how creativity and analytical rigour can complement each other rather than conflict.

The proposed teaching model is intentionally practice-oriented and adaptable. It does not require significant changes to course content or assessment design and can be integrated into existing project-based learning activities in software engineering courses. As such, the approach has the potential to be transferred to other practice-oriented computing courses where students face similar challenges in balancing creativity and structured reasoning.

Several limitations should be acknowledged. The current study is based primarily on qualitative classroom observation and teaching reflection rather than formal quantitative evaluation. In addition, some students required additional scaffolding to adapt to the open-ended nature of early-stage design activities. Future teaching practice may incorporate more structured reflective tools and explore complementary evaluation methods to further examine learning outcomes.

In conclusion, this teaching reform demonstrates that structured thinking support can play a valuable role in enhancing software engineering education. By guiding students through a systematic transition from idea exploration to reasoned decision-making, the proposed model contributes a practical and transferable approach to supporting creative and analytical thinking in undergraduate software engineering courses.

Funding

2024 Fujian Provincial Department of Education Undergraduate Teaching and Learning Research Project "A Study on the Development of Modern Industry Colleges under the New Engineering Education Framework" (NO: FBJY20240069)

References

[1] Jiang S, Pang J. Enhancing Design Thinking in Engineering Students with Project-Based Learning [J]. Computer Applications in Engineering Education, 2023, 31(4): 814–830.

[2] Caratozzolo P, Álvarez-Delgado J. Improving Creative Thinking in Engineering Students Through Art Appreciation [C]// Proceedings of the ASEE Annual Conference & Exposition. Washington, D.C.: ASEE, 2019.

[3] Song H. Promote Teaching Reform with Innovation Competition [C]// Proceedings of the 2017 International Conference on Social Science, Humanities, and Education (ICSSHE 2017). Paris: Atlantis Press, 2017: 415–418.

[4] Luo X, Wu L. Optimization of Course Content and Teaching Pathways for the JavaEE Framework Technology Course Oriented Toward Innovative Talent Cultivation [J]. Education Insights, 2025, 8(2): 55–62.

[5] Schroeter D, Forrester S. Diverging from the Dogma: A Call to Train Creative Thinkers in Science [J]. The Bulletin of the Ecological Society of America, 2018, 99(4): e01464.