https://doi.org/10.34074/rere.00504

Reflection

Beyond the Diagram: Structured Reflection and Peer Feedback in Novice Software Design Learning

Asanthika Imbulpitiya & Waruni Hewage

OTAGO POLYTECHNIC AUCKLAND INTERNATIONAL CAMPUS

ABSTRACT

Teaching abstract modelling specifically class and entity relationship diagrams remains a persistent challenge in undergraduate computing. Even in project-based courses focused on providing hands-on experience, beginners frequently replicate patterns, complicate models unnecessarily, or incorrectly apply relationships without explaining the reasoning behind their decisions. This opinion piece discusses our experiences in teaching database and software design in two first-year courses and describes a simple pedagogical intervention implemented between the requirements analysis and the individual diagramming stages. This approach consist of four main structured stages: Scenario Analysis, Peer Review, Reflection and Diagram Construction. Although this is not a controlled study, our reflections indicate that incorporating early peer review and reflective practices can help novices shift their emphasis from "what belongs where" to "why this abstraction exists," ultimately enhancing the quality and communicative effectiveness of their diagrams. We present practical suggestions for educators to integrate this approach into similar courses that require teaching design specifically for novice students.

Keywords: Computer Science Education, *Reflective Learning*, *Peer Feedback*, *Software Design*

INTRODUCTION

Software design is a fundamental element of the computing curriculum, yet effectively teaching it, specifically to novices remains a significant challenge (Batra, 2007; Katz & Shmallo, 2016). Design diagrams such as class or entity relationship diagrams require learners to combine their understanding of the domain, abstraction skills, and the conventions of notation. Although activity-based learning and real-world examples are popular in in teaching this domain, many novices still tend to approach design tasks with superficial reasoning, replicating familiar patterns without fully grasping the underlying principles guiding their decisions. This gap is important: When students view diagrams as mere decorative elements instead of interpretive models, it adversely affects the later stages of implementation, testing, and maintenance.

This paper discusses a recent intervention in the classroom that led to significant improvements in student engagement and comprehension of software designs. This intervention is rooted in reflective practice and collaborative learning concepts (Schön, 2017; Topping, 1998). While the change we did was small, it has a substantial impact on how students approach their learning. Instead of allowing students to move straight from requirements analysis to individual diagram creation, we incorporated a short, structured checkpoints where learners articulated their reasoning, examined different abstractions, and provided specific critiques prior to finalising their designs. The goal was not to increase the content but to alter the timing and manner in which understanding occurs.

The study is situated in two first-year studio courses offered at a Polytechnic in New Zealand, in which students developed both database and software designs to integrate into their project implementations (Hewage & Imbulpitiya,

2025). These courses emphasises a collaborative learning environment with hands-on experience, and despite this supportive nature, students find it difficult to work with these abstract models. This often leads to overcomplicating the models, misapplied relationships, or treating notation as a checklist rather than a language for articulating design decisions. Our intervention addresses this particular challenge in the learning process.

This article is a collective reflection on our experience teaching software design, highlighting the methods we employed throughout the teaching process. Presented as an opinion piece instead of a formal study, it extracts practical insights from our classroom experience. We aspire that the lessons shared here will assist educators and instructors in related fields to reassess elements of their instruction in teaching software design for novices.

The remainder of this article is structured as follows. The next section reviews existing work in this research domain. We then describe in detail the intervention applied in the teaching process. Finally, we present a discussion of our findings and conclude the paper.

LITERATURE REVIEW

Cognitive complexity in software design

Grasping the numerous interrelated abstract concepts in software design poses a well-recognised challenge for beginners (Amin et al., 2019; Katz & Shmallo, 2016). In a standard introductory software design course, students must quickly comprehend various concepts such as requirements analysis, conceptual modelling, design principles, and implementation factors. These concepts typically do not function independently; rather, they are best understood in relation to one another. For example, creating a design model requires not only familiarity with the relevant notations but also a clear understanding of the essential requirements, an anticipation of implementation challenges, and a consideration for maintainability. This interconnection results in a complicated learning environment where understanding individual topics in isolation does not suffice for proficient performance.

Software design is fundamentally an interactive learning area, where the concurrent handling of numerous interconnected components creates a significant intrinsic cognitive burden (Sweller, 1988, 2010). For beginners, this burden can impede the development of strong mental models that bring together procedural knowledge (how to implement techniques), declarative knowledge (what the concepts signify), and conditional knowledge (when and why to apply particular methods). Consequently, learners frequently find it challenging to "think like designers," lacking the conceptual synergy that experts utilise to assess trade-offs, foresee subsequent effects, and make well-informed design choices. Previous studies in software engineering education have indicated that these challenges are not confined to a particular design subfield—like databases or user interfaces—but are widespread throughout the overall design process due to its cognitive demands (Batra, 2007; Katz & Shmallo, 2015).

In light of these challenges, educators have investigated a range of pedagogical strategies designed to minimise unnecessary cognitive burden while encouraging deeper conceptual relationships. Approaches such as pedagogical models, work examples, scaffolding, and visual modeling tools have proven effective in managing complexity and making abstract ideas more understandable (Katz, 2018, 2020; Katz & Shmallo, 2016; Watson, 2006). Nevertheless, while these techniques facilitate knowledge acquisition, they often do not directly prompt learners to reflect on their thinking processes or acknowledge the evolution of their understanding over time. This is where reflective learning adds another layer: by encouraging students to express their reasoning, assess their design choices, and link new understandings to existing knowledge, reflection can aid in solidifying mental frameworks and enhance the application of learning in new situations. Therefore, in the realm of software design education, incorporating

reflective learning into teaching methods offers a means not only to alleviate cognitive overload but also to cultivate the metacognitive abilities critical for developing design proficiency.

Reflective learning as a pedagogical strategy

Reflective learning is a teaching approach where students actively consider, evaluate, and comprehend their own learning and experiences (Moon, 2013; Schön, 2017). Reflective learning asks students to stop and critically consider their assumptions and decision-making processes, rather than focus only on the acquisition and application of technical knowledge. This kind of reflection, which aims to bring hidden ideas to the surface, can take many different forms, including written journals, reflections following tasks, design justifications, or organised conversations. In software design, reflective techniques allow students to document their design choices and the rationale behind them, establishing the foundation for deeper understanding.

In terms of cognitive load, reflective learning can assist in managing the inherent complexity of software design by allowing learners to decompose complicated tasks into smaller, more manageable parts. By expressing their thought processes, students can articulate and structure their mental models, which helps to relieve working memory and aids in the development of schemas (Sweller & Ayres, 2011). Furthermore, reflection promotes progressive integration of concepts in which learners revisit previous ideas in the context of new understanding, thereby reinforcing the links between related topics. This repetitive process reflects the method of an expert designer who continually evaluates decisions against changing constraints, requirements, and design principles.

Peer feedback in learning

Peer feedback is a collaborative learning approach where students assess and offer constructive critiques on their peers' work, usually based on a defined criteria (Topping, 1998). In computer science education, peer feedback has been demonstrated to improve conceptual comprehension and participation (Turner & Perez-Quinones, 2009), strengthen learning in visualisation classes (Beasley et al., 2020) and enhance self-regulated assessment and motivation in software engineering (Groeneveld et al., 2020). A recent meta-analysis corroborates these beneficial effects across various computer-based learning environments (Li et al., 2024).

In design education, peer feedback fulfils multiple roles: it introduces students to various approaches, encourages critical analysis, and helps them internalise standards of quality. By reviewing a classmate's design, learners must use the same evaluation criteria they apply to their own work, which deepens their understanding of the design process. This evaluative exchange also fosters a sense of ownership over their learning, as students actively contribute to each other's growth rather than relying solely on instructors for feedback. More recently, researchers have utilised this specifically in database courses. A recent study conducted in an introductory database course investigates team-based peer reviews (Catania et al., 2022). In this study, participants apply a cooperative methodology where teams review one another's design artifacts as part of final design activities. This approach extends peer feedback into the design domain and promotes richer conceptual discussion.

Since both reflective learning and peer feedback have been identified in the literature as effective techniques for learning abstract and complex concepts such as designing, we attempted to combine both strategies in our teaching approach. The aim was to investigate how this integrated approach could support novice learners in software design by reducing cognitive load, enabling progressive conceptual integration, and fostering the metacognitive skills needed to develop design expertise.

THE INTERVENTION

Through our experience in delivering courses that incorporate conceptual design elements, it has become clear that many students find it challenging to grasp the abstract concepts needed to finalise their designs. This difficulty also applies to some of the students who are technically proficient. Recognising the complexities involved in the design process, we have experimented with various strategies to enhance students' comprehension of this process. As we explored different methods, we discovered that a significant factor in the students' struggles is their lack of reflective practices, which involve reviewing their actions, experiences, or learning to find meaning in them. Reflection encompasses more than mere recall; it involves analysing, questioning, and linking experiences to enrich understanding. Although we anticipate that the students will reflect on their learning, it has not occurred without explicit intervention. Consequently, we have introduced design activities that incorporate a mandatory reflective component, requiring students to complete this step to finish the activity. This revised approach introduced a four-step process, explicitly incorporating peer feedback and reflection before students committed to their designs.

The following is the process followed in the Entity Relationship Diagram (ERD) design activity according to the proposed revised approach.

Step 1: Scenario Analysis - Students analysed their selected project topics to identify entities, attributes, and relationships in groups. They extracted candidate elements and constructed a logical draft ERD based on their findings.

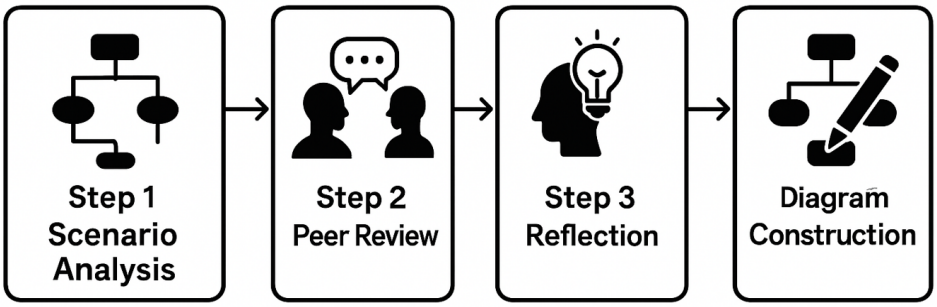
Step 2: Peer Review - The groups exchanged their draft ERD, visited the work of other groups, and provided structured feedback guided by specific review questions.

Step 3: Reflection – Students reviewed and reflected on feedback received from other groups. They critically evaluated the comments, discussed their assumptions, and revised their logical ERD designs accordingly.

Step 4: Diagram Construction - After reflection and critical evaluation, students finalised the logical ERD design, ensuring that the entities, attributes, and relationships were accurately modelled.

The Figure 1 represents the four main stages of the proposed approach.

Figure 1: A structured four-step learning process



This staged method intentionally allows students to incorporate reflection into their learning process. In Step 1, students start by examining their own beliefs about the issue at hand, which promotes self-awareness and enhances critical thinking skills. Step 2 brings in diverse viewpoints through peer review, motivating students to evaluate their reasoning against that of others and to acknowledge different strategies. In Step 3, guided reflection enables students to thoughtfully assess the feedback they have received, prioritise between constructive and less helpful feedback, and link theoretical concepts with practical application as they enhance their designs. Ultimately, Step 4 wraps up this reflective journey by converting insights into a more refined and precise logical ERD. Hence, the activity not only

navigates students through the technical components of database design, but also fosters the reflective practices essential for deeper understanding, flexibility, and ongoing growth. Please refer to Appendix 1 for the complete lesson plan of this activity.

Following the success of this intervention, we adopted a similar structured method for another course on software design, particularly focusing on the creation of class diagrams. Understanding that students faced similar challenges when moving from abstract problem descriptions to actual class structures, we modified the four-step process for this new setting.

In this activity, students initially participated in scenario analysis, where they investigated a specific problem domain to identify potential classes, their attributes, and possible methods. This phase encouraged them to extract and organise crucial information, forming a solid groundwork for informed design choices. During the peer review segment, groups shared their preliminary findings and offered structured feedback. This process allowed students to question themselves and re-evaluate their assumptions and design decisions to identify areas for improvement. In the reflection phase, students thoughtfully reviewed the feedback they received, engaged in discussions about improvements within their groups, and updated their findings accordingly. Lastly, in the diagram construction phase, they integrated these insights to create their initial class diagrams with enhanced clarity and confidence.

This approach allowed students to incorporate reflective practices into their learning process, progressing from technical skills to a more profound comprehension of software design choices. Similar to the ERD activity, students expressed increased confidence in expressing their design decisions, showed greater clarity, and demonstrated an improved ability to justify their modelling choices.

Moreover, the structured design provides the lecturer with natural opportunities to intervene and offer guidance. At each stage the lecturer can monitor group progress and step in when students appear stuck or uncertain. Such timely interventions not only help to resolve misunderstandings but also model effective problem-solving strategies, ensuring that students remain engaged and supported throughout the learning process. Although this sequencing delayed the construction phase slightly, the additional time devoted to cognitive rehearsal and social learning proved valuable. Students demonstrated clearer articulation of their design decisions and displayed greater confidence in explaining and justifying their diagrams.

DISCUSSION AND CONCLUSION

This intervention demonstrated how including scaffolded peer review and reflection into the design process significantly enhances students' learning outcomes beyond mere technical competencies. While the produced diagrams still required refinement to reach industry standards, the enhancement in clarity of the students' thought processes and the intention behind the design was striking. Most importantly, students started to demonstrate more advanced skills that go beyond merely creating accurate models demonstrating an understanding of the principles rather than fixating on the rules of drawing diagrams. Moreover, their design rationale was based on peer interactions. This suggested that the provision of collaborative feedback led to more richly justified design decisions. In addition, students were more confident than before in explaining their decisions and the associated constraints, indicating a shift from passive to actively taking part in the learning process.

Structured peer review proved to be especially effective. If peer interactions are well planned, guided, and adequately supported, it creates a low-pressure environment for students to explore ideas, recognise misunderstandings and embrace different viewpoints (Liu & Carless, 2006). This indicates that social aspect of learning which could be often overlooked in courses with technical emphasis, can act as an important link between theoretical understand-

ing and practical application. Specifically in tasks that require the understanding of abstract concepts like software design.

As educators, we were particularly impressed by the meta-cognitive advancement this approach facilitated. Students not only acquired knowledge on how to design but also started to recognise the significance of their design choices a vital distinction in cultivating reflective practitioners. This underscores the importance of deliberately integrating reflection as a key component of design education rather than expecting it to develop naturally.

The outcomes of this intervention align with earlier research that highlights the importance of reflection in enhancing deep learning and conceptual understanding(Moon, 2013). By incorporating structured opportunities for critique and guided reflection, the activities addressed common challenges faced by novice designers, specifically their tendency to focus on syntactic accuracy rather than conceptual understanding and their hesitation evaluate their own work (Batra, 2007; Katz & Shmallo, 2015). This observation reinforces the idea that participating in reflective practice not only results in producing improved artefacts but also fosters the cognitive skills necessary for sustained professional development.

The findings of this study reveal a number of practical approaches for educators seeking to design effective learning activities around abstract concepts:

- **Prioritise reflection before design:** Place reflection and evaluation ahead of design deliverables to establish a strong conceptual foundation.
- **Structured peer interaction:** Provide scaffolding to ensure feedback is constructive and promotes critical engagement.
- **Emphasise justification over output:** Motivate students to articulate the reasons for their design choices instead of merely showcasing technical skills.

As the field of computing education evolves rapidly, it is important to comprehend that reflective practices are fundamental in learning than supplementary. By incorporating peer review and guided reflection at the beginning of the design process, abstract and often intimidating tasks can be transformed into opportunities for genuine comprehension and professional growth.

REFERENCES

Amin, M., Romney, G. W., Dey, P., & Sinha, B. (2019). Teaching relational database normalization in an innovative way. *J. Comput. Sci. Coll.*, *35*(2), 48–56.

Batra, D. (2007). Cognitive complexity in data modeling: Causes and recommendations. *Requirements Engineering*, *12*(4), 231–244. https://doi.org/10.1007/s00766-006-0040-y

Beasley, Z., Friedman, A., Pieg, L., & Rosen, P. (2020). Leveraging peer feedback to improve visualization education. *2020 IEEE Pacific Visualization Symposium (PacificVis)*, 146–155. https://doi.org/10.1109/PacificVis48177.2020.1261

Catania, B., Guerrini, G., & Traversaro, D. (2022). Collaborative learning in an introductory database course: A study with think-pair-share and team peer review. *Proceedings of the 1st International Workshop on Data Systems Education*, 60–66. https://doi.org/10.1145/3531072.3535330

Groeneveld, W., Vennekens, J., & Aerts, K. (2020). Engaging Software Engineering Students in Grading: The effects of peer assessment on self-evaluation, motivation, and study time. https://doi.org/10.48550/ARXIV.2012.03521

Hewage, W., & Imbulpitiya. (2025). From classroom to career: Student perceptions of workplace simulation in it education. *Proceedings of the International Conference on Education*, *11*, 275–291. https://doi.org/https://doi.org/10.17501/24246700.2025.11120

Katz, A. (2018). Visualizing the syntax of gradual transitions in teaching database modeling.

Katz, A. (2020). Improved Teaching of Database Schema Modeling by Visualizing Changes in Levels of Abstraction. *Journal of Information Systems Education*, *31*(4), 294. Retrieved August 20, 2025, from https://jise.org/Volume31/n4/JISEv31n4p294.html

Katz, A., & Shmallo, R. (2015). Improving relational data modeling through learning from errors.

Katz, A., & Shmallo, R. (2016). Learning from errors as a pedagogic approach for reaching a higher conceptual level in database modeling. In J. Krogstie, H. Mouratidis, & J. Su (Eds.), *Advanced information systems engineering workshops* (pp. 93–102). Springer International Publishing.

Li, C., Yang, Z., & Yang, Y. (2024). The impact of peer feedback on student learning effectiveness: A meta-analysis based on 39 experimental or quasiexperimental studies. In J. Gan, Y. Pan, J. Zhou, D. Liu, X. Song, & Z. Lu (Eds.), *Computer science and educational informatization* (pp. 42–52). Springer Nature Singapore.

Liu, N.-F., & Carless, D. (2006). Peer feedback: The learning element of peer assessment. *Teaching in Higher Education*, *11*(3), 279–290. https://doi.org/10.1080/13562510600680582

Moon, J. A. (2013, April). *A Handbook of Reflective and Experiential Learning* (0th ed.). Routledge. https://doi.org/10.4324/9780203416150

Schön, D. A. (2017, March). *The Reflective Practitioner* (0th ed.). Routledge. https://doi.org/10.4324/9781315237473

Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*, *12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Sweller, J. (2010). Element interactivity and intrinsic, extraneous, and germane cognitive load. *Educational Psychology Review*, *22*(2), 123–138. https://doi.org/10.1007/s10648-010-9128-5

Sweller, J., & Ayres, S., Paul nd Kalyuga. (2011). *Cognitive Load Theory* (1st ed.). Springer. https://doi.org/10.1007/978-1-4419-8126-4

Topping, K. (1998). Peer Assessment Between Students in Colleges and Universities. *Review of Educational Research*, *68*(3), 249–276. https://doi.org/10.3102/00346543068003249

Turner, S., & Perez-Quinones, M. A. (2009). Exploring Peer Review in the Computer Science Classroom. https://doi.org/10.48550/ARXIV.0907.3456

Watson, R. (2006). The Essential Skills of Data Modeling. *Journal of Information Systems Education*, *17*(1), 39–42. https://aisel.aisnet.org/jise/vol17/iss1/7

Appendix - Lesson Plan - Week 6 (Session 2)

Course Name	Studio 1 (Level 5)	**Date**	05/06/2025
Lesson Topic	Entity Relationship diagram (ERD)	**Lesson length**	65 minutes

Relevant Course Learning Outcomes:

Redacted

Lesson Objectives – by the end of this session, the <u>learners</u> will be able to:

1. Collaboratively design a logical ERD.
2. Review a logical ERD created by someone else and provide constructive feedback.
3. Review and reflect on the feedback received on your ERD and construct the final diagram.

Task purpose *(this will clearly align with Lesson Objectives)*	**Task procedure – what the <u>teacher</u> is doing**	**Task procedure – what the <u>learners</u> are doing** *(include interaction patterns: T-S/ T-Ss/ Ss-Ss/ S-S/ S-T)* [1]	**Output**	**Time & Duration**
Refreshing the memory of the concepts that they have learnt so far. The task is to create the logical ERD for their group project. This is a group activity. Groups are pre-formed with 3-4 students.	Interact with students to refresh their understanding of previously learned concepts. Some generic questions related to ERDs will be asked to make sure students are confident on the task. Sample questions: *What is an ERD? what are the main components of ERD? How many types of ERDs are there? What are they?* Explaining the purpose of the task and how it's directly related to their final project.	***T-Ss-T***: Involve in the discussion and answer to the questions ***T-Ss***: Listening to the instructions provided by the lecturer to understand the task clearly and how It directly related to the project. ***S-T***: Asking questions to clarify any doubts or understand the task better (If any).	No Concrete output	10 min
Completing the **step 1** of the activity: **Scenario analysis**	Walking to every group and checking the progress. *(is everyone actively engaged in the discussion? has someone taken the responsibility to draw the ERD on the paper etc.)* Making sure and encouraging all the students to actively engage in the activity. *(Ask a few specific questions about the tasks or ask them to explain a relationship or reason behind making a certain choice etc.)*	***Ss-Ss***: Analyse the selected topic to identify entities, attributes and relationships. ***Ss-Ss***: Draw the draft **ERD** on the paper provided based on the discussions. Further discussions may need in deciding relationships. ***S-T/Ss-T***: Asking questions or asking for support when needed to complete the task successfully.	Logical ERD on paper	15 mins.

[1] T-S: Teacher to student, T-Ss: Teacher to students, Ss-Ss: Students to students, S-S: Student to Students, S-T: Student to Teacher

Task purpose *(this will clearly align with Lesson Objectives)*	**Task procedure – what the <u>teacher</u> is doing**	**Task procedure – what the <u>learners</u> are doing** *(include interaction patterns: T-S/ T-Ss/ Ss-Ss/ S-S/ S-T)* [1]	**Output**	**Time & Duration**
Completing the **step 2** of the activity: **Peer Review** Students are expected to visit other groups and provide feedback on their ERDs	Instructing students on how to move from table to table. Check the time and instruct them to move to other groups if they spend too much time on one group	***Ss-Ss***: Each group will go to all the other groups according to the instructions, check the ERDs (entities, relationships, attributes) and write down their feedback (use a different colour pen). *(Are all relevant entities included? Are the relationships accurately created? Are any important attributes missing?)*	Papers with the comments from other groups.	15 mins.
Completing the **step 3** of the activity: **Reflection** Everyone will discuss and reflect on the feedback provided stories from other groups	Instructing students to review feedback provided by other groups and reflect/critique on that. *(Is it relevant? Does it help improve your design? Should you adopt it or not? Justifications of decisions made)* Leading the discussion whenever needed. Check the time and instruct them to move to other groups if they spend too much time on one group	***Ss-Ss***: Discuss and reflect on the feedback provided by other groups. Explain the decisions to the class. Explain why they chose to adopt or reject each suggestion.	Decisions and justifications related to the peer feedback.	15 mins.
Completing the **step 4** of the activity: **Diagram Construction**	Instructing students to update their logical ERD based on the review and reflection decisions.	***Ss-Ss***: Update the **ERDs** based on the decisions made during the reflection.	Final logical **ERD**	10 mins.