Bridging Education and Practice: Lessons Learned from 10 Years of Using Reflective Practices in a Software Engineering Studio

Stan Kurkovsky
Central Connecticut State University
New Britain, CT, USA
kurkovsky@ccsu.edu

Chad Williams
Central Connecticut State University
New Britain, CT, USA
cwilliams@ccsu.edu

Abstract

Effective reflection is essential to professional software engineering practice, yet remains difficult to cultivate in educational contexts. Students often perceive reflective activities as secondary to technical work, and, without scaffolding, they rarely move beyond surface-level observations, missing opportunities to build professional skills. This paper reports on ten years of embedding structured reflection into a two-course capstone sequence in an ABET-accredited undergraduate Computer Science program. Our experience shows that reflection is most powerful when embedded within authentic, externally sponsored projects supported by multiple forms of formative feedback. The integration helps students bridge classroom learning with professional practice.

Analysis of student reflections collected across multiple project stages shows a clear trajectory: initial views of reflection as stressful or externally imposed shifted toward recognition of these practices as authentic preparation for professional work. Findings yield actionable instructional strategies for scaffolding this transition and are synthesized into five cross-cutting design principles. This work offers guidance for integrating reflection into project-based software engineering education to strengthen both technical and professional competencies.

CCS Concepts

• **Social and professional topics → Software engineering education**.

Keywords

Reflective practices, software projects, software studio

ACM Reference Format:
Stan Kurkovsky and Chad Williams. 2026. Bridging Education and Practice: Lessons Learned from 10 Years of Using Reflective Practices in a Software Engineering Studio. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-SEET '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3786580.3786963

1 Introduction

Reflection is essential to professional software engineering (SE) practice, fostering teamwork, communication, and decision-making skills. Yet, novice developers often view it as extraneous to technical work, requiring structured support to move beyond superficial observations [3, 15, 31].

While research demonstrates the benefits of integrating reflection into SE education, few studies examine its sustained role within a capstone experience involving external partners, iterative development, and structured feedback. Understanding student engagement with reflective practices is crucial for bridging the gap between academic preparation and professional expectations.

This paper reports on ten years of embedding reflective practices into a required two-course capstone sequence in an ABET-accredited BS Computer Science program at a regional public university in the USA. Student teams work on externally sponsored projects from industry, government, and community organizations. Reflection is deliberately embedded throughout the project lifecycle, including team agreements, scrums and retrospectives, and lessons-learned activities, while client check-ins and peer presentations provide formative feedback that fuels these reflections.

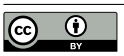
Our analysis of student reflections collected at multiple semester points, reveals how perceptions evolve from viewing reflective activities as externally imposed requirements to recognizing them as authentic professional practices. Findings show reflection supports better project outcomes and fosters transferable skills such as communication, accountability, and stakeholder engagement.

The paper's contributions are threefold: first, we detail the systematic embedding of reflective practices within a two-course capstone sequence using the *Scaffolded Projects for Social Good* (SPSG) framework [20, 36]; second, we report on an empirical study of student reflections revealing how perceptions of reflective practices changed across the project; and third, we synthesize these insights into actionable instructional strategies and cross-cutting design principles that other educators can adapt to their own contexts.

The remainder of the paper is structured as follows: Section 2 describes the course context and the SPSG framework. Section 3 outlines the design principles that guided the embedding of reflection. Section 4 explains the methodology for collecting and analyzing student reflections. Sections 5 through 8 present findings organized around specific reflective activities: peer presentations, team scrums and retrospectives, client check-ins, and peer learning from retrospectives. Section 9 synthesizes these findings into cross-cutting instructional principles. Finally, we discuss implications for teaching software engineering and directions for future work.

2 The Context

Over the past decade, we've observed students' engagement with reflection evolving across scrums, retrospectives, and peer presentations, revealing patterns that shaped our design principles. The study is embedded within a two-course capstone sequence (CS 410



ICSE-SEET '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2423-7/26/04
https://doi.org/10.1145/3786580.3786963

Software Engineering and CS 498 Senior Project) required for all students in the ABET-accredited BS Computer Science program at a large regional university in the USA.

This paper outlines a framework for embedding structured reflection into software engineering education, improving student learning outcomes and preparing them for professional practice. Projects follow the studio-based Scaffolded Projects for Social Good (SPSG) framework [20, 36]. This model emphasizes authentic, externally sourced multi-semester projects, iterative development organized around Scrum, and deliberate scaffolding for reflection at each stage, facilitated by sustained interaction among all participants (Table 1). This studio environment facilitates mentorship from industry professionals, recent graduates, and student peers, and ensures learning and skills development for all students in an iterative and collaborative process. This approach supports continuous feedback and improvement, and allows students to play different roles within the team to help them better understand various aspects of software engineering [19, 21].

Each semester, 10-15 teams of four students work on externally-sponsored projects, dedicating approximately ten hours per week outside of regular class meetings. Projects are drawn from diverse partners, including major insurance companies, aerospace and robotics, startups, municipal agencies, and community non-profits. The capstone sequence is designed to simulate professional engineering contexts, providing structured opportunities for reflection and feedback. While CS 410 emphasizes technical development supported by lectures and four sprints, CS 498 extends to six sprints and incorporates discussions and debates on professional, ethical, and societal issues.

Reflection is embedded throughout the project lifecycle and deliberately coupled with feedback from peers, instructors, and clients. This interplay makes reflection meaningful and transferable to professional practice. Figure 1 provides a simplified timeline showing how these key events interleave across the semester, highlighting how authentic feedback opportunities fuel subsequent reflection. Key elements include:

Whole-class weekly scrums. During development weeks, each team (typically 5-7 teams per section) delivers an 8-10 minute update using a visible project board. The purpose is to make work-in-progress transparent and to practice professional communication. The instructor and peers pose clarifying and probing questions, with suggestions encouraged but not graded, thereby keeping the emphasis on formative feedback.

Whole-class sprint retrospectives. At the conclusion of each sprint (four to six per term), teams participate in a structured retrospective lasting approximately 12-15 minutes. Using a fixed prompt—*what worked, what did not, and what we will change next*—teams reflect on their processes and outcomes. They also report quantitative metrics, including story points planned versus completed and the specific contributions made by each student.

While scrums, retrospectives, and written reflections serve as primary reflection points, the following are best understood as formative feedback opportunities providing raw material for student reflection:

Elaboration-phase critiques. During early phases, teams present key planning artifacts for instructor critique. These include the team agreement, proposal reflection, requirements specification (use cases and user stories), and product backlog. The product backlog is also presented in a Development Kick-off session to make project scope and priorities visible across the class. At this stage, peer feedback is minimal but encouraged.

Partner sprint reviews with live demos. Every two weeks, teams conduct virtual sprint reviews with their project partners that focus on demonstrating functionality completed during the sprint. Teams record meeting minutes, and partners confirm decisions through written follow-up, increasingly supported by recorded video and documentation.

Required weekly partner check-ins. Beyond sprint reviews, teams meet weekly with their project partner for 15-20 minute virtual check-in. Agendas are circulated in advance, and outcomes are documented as backlog updates and included in sprint reports. These meetings provide continuity, reduce ambiguity, and sustain accountability.

Final demo. At the end of the semester, each team delivers a public demonstration of the completed system, typically in person. The demo is followed by a brief "lessons learned" discussion. Some local partners attend these sessions and, in some cases, host the final demonstration on site, thereby reinforcing the authenticity of the project experience.

User & deployment/transition documentation. To ensure operational readiness, each team produces written documentation covering system deployment, user instructions, and transition planning. These artifacts make explicit the assumptions underlying the system and support hand-off to the client organization.

3 Rationale and Background Work

This section outlines several principles grounded in research literature that guided the design of reflective practices in the course. These principles translate the course structure (face-to-face class meetings, regular video-conference partner interactions, and 4-6 two-week sprints) into mechanisms that make feedback timely, visible, and actionable. Collectively, they demonstrate how reflection is positioned as a formative and recurring activity. Central to our design is integrating reflection with authentic audiences and feedback mechanisms, including peers, the instructor, and external partners, and how scaffolds such as prompts, rubrics, and process data support meaningful engagement, ensuring reflection is never abstract but always tied to visible work and professional expectations. These principles also explain the choice to emphasize low-stakes grading that rewards candor and improvement, to triangulate feedback across multiple perspectives, and to focus analysis on structured classroom interactions rather than informal team communication.

Formative by design. Reflection is embedded within routine project work so that assessment continuously informs both learning and teaching, rather than operating solely as an end-point judgment. In problem-based SE settings, such alignment enables timely redirection of effort and supports the development of decision-making, teamwork, and problem-solving competencies. Frequent, feedback-oriented checkpoints are therefore designed to close performance gaps as they emerge [6, 22, 24].

Public accountability leads to better articulation and quality. Whole-class, in-person scrums and retrospectives require teams to make

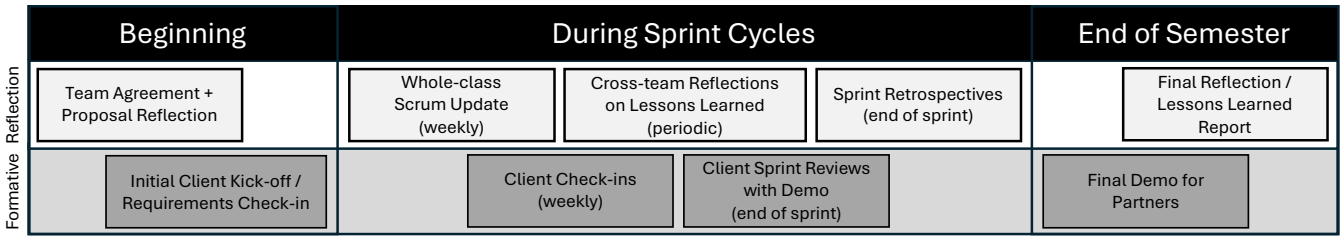


Figure 1: Interplay of Formative Feedback and Reflection

Actors	Cadence / Timing	Modality	Purpose / Outputs	Primary Artifacts
T ↔ PP	Weekly check-ins	VC	Status, clarify requirements, surface risks, confirm next steps (feeds team reflection)	Meeting notes; backlog updates; action items
T ↔ PP	Sprint reviews with live demos	VC	Increment demo; acceptance/rejection; reprioritization (students later reflect on adjustments)	Review decisions; acceptance notes
I ↔ T	Elaboration	F2F	Requirements quality; early critique	Team agreement; proposal reflection; requirements; product backlog
I ↔ T	Development	F2F	Reflect-in-action; blockers; coordination	Scrum notes; updates to sprint report
I ↔ T	End of each sprint	F2F	Reflect-on-action; decide process/plan changes	Written sprint report (lessons; next plan)
I ↔ T	Transition	F2F	Validation & handoff/continuation	Final demo; user guide; deployment/transition docs
T ↔ Class	Weekly scrums & sprint retros	F2F	Peer critique; cross-team learning	Class Q&A captured in team/instructor notes

Table 1: Interaction matrix: actors, modality, cadence, and artifacts showing feedback opportunities supporting reflection. Legend: T = team(s); PP = project partner (client); I = instructor; F2F = face-to-face, in class; VC = live video conference.

work visible, justify design choices, and respond to informed critique. This structure mirrors studio practice, where public presentations surface diverse viewpoints and cultivate professional communication as an integral part of reflective learning [3, 30, 31].

Authentic partner interactions build professional judgment. Regular partner check-ins and sprint reviews expose students to evolving constraints and stakeholder priorities, conditions under which professional judgment develops. Authentic audience/contact are a defining feature of studio-style courses; in engineering education, partner encounters are explicitly leveraged for formative feedback on professional behaviors [12, 15, 34].

Written sprint reports externalize learning, not just output. Concise, templated reports (e.g., goals vs. delivered, evidence/tests, one process insight, one change to try, backlog and risk snapshots) transform tacit team reasoning into durable traces that support self-, peer-, and instructor feedback. This approach is consistent with PBL/SE practices (journals, self/peer assessments) and with learning analytics findings on the value of maintaining activity traces for improvement [1, 22, 32].

Retrospective prompts matter because reflection is hard. Because unprompted reflection is uncommon, explicit metacognitive scaffolds are required. Focused prompts ("what worked/what didn't/what we'll change," coupled with process metrics) operationalize self-regulation (planning, monitoring, and control) and help students link actions to outcomes. Evidence from SE education associates such supports with improved task performance [18, 23, 27, 28].

Whole-class Q&A builds a community of practice. Running scrums and retrospectives in front of all teams fosters collective sensemaking: students appropriate others' strategies, refine questions, and normalize critical yet constructive discourse. Studio scholarship highlights "presentation sessions" as public examinations of work from multiple viewpoints; large-group presentations and retrospectives institutionalize this community effect [3, 30, 38].

Iteration with demos focuses reflection on evidence. End-of-sprint demonstrations anchor discussion in observable artifacts (working software/tests), sharpening feedback and limiting abstraction. Across CS/SE formative-assessment studies, students value immediate, granular feedback that supports the next step, which is supported by a two-week sprint cadence [4, 29, 34].

Partner + instructor + peer + class = richer feedback. Combining partner reviews (authenticity), instructor critique (standards), and whole-class Q&A (diverse heuristics) provides complementary perspectives that no single rater can offer. With the appropriate structured scaffolding, the quality of peer feedback can approach the quality and value of instructor findings [7, 22, 24].

Low-stakes grading emphasizes reflection quality over perfection. Grading scrums/retrospectives for participation and reflective quality encourages candor about risk, blockers, and process change, which enables the most impactful formative feedback. Higher-education research favors ongoing, actionable guidance; PBL literature stresses co-planning assessment with teaching so feedback can redirect effort during the project [2, 5, 32].

Formal and informal peer review helps sharpen analysis. Critiquing others' artifacts (and defending one's own) develops evaluation, defect-detection, and explanation skills that transfer to one's own work. SE education studies report analytical gains and instructor-comparable issue detection when peer review is well structured and embedded across phases [11, 13, 26, 35, 39].

Not all "talk" is reflection (and that's okay). Unstructured coordination channels (e.g., chat/email) often contain limited reflective content. Focusing analysis on prompt-driven, in-class rituals (scrums, retrospectives, reports, reviews) directs attention to richer evidence of reflection and keeps evaluation aligned with the course's reflective design. Reflection quality improves when goals and criteria are explicit and activities are aligned to them [16, 18, 24].

Industry-relevant skills and the "gap." Embedding reflection with real stakeholder engagement is crucial for bridging enduring gaps between curricula and practice, especially in stakeholder communication, decision-making under uncertainty, and teamwork. Recent mappings and perspectives document this gap's persistence and endorse the need for realistic, trend-aligned projects involving external stakeholders as remedies [8, 25, 33].

4 Methodology

Data Collection. The study draws on written reflections collected through open-ended survey questions embedded at multiple points in a semester-long software engineering project course. Questions were posed at three stages: pre-project (e.g., Q1a–Q3a), mid-project (e.g., Q4a–Q7a), and post-project (e.g., Q45b and others). These questions invited students to comment on the value, challenges, and outcomes of reflective practices such as weekly scrums, sprint retrospectives, and peer presentations. Across all questions, 30-60 responses were typically collected, with counts varying by question due to enrollment and survey completion rates.

Data Processing. Responses were exported from the survey system into one spreadsheet; each column represented a question, each row a student's response. Data were cleaned by removing blank entries, normalizing text (e.g., converting to lowercase, removing punctuation), and consolidating repeated responses. For analysis, both original student wording (to preserve nuance) and preprocessed version (to support keyword clustering) were maintained.

Thematic Coding. An inductive thematic analysis approach was used to code responses. Each question's dataset was examined for recurring ideas, which were then grouped into categories (e.g., accountability, feedback, stress, team alignment). Coding was iterative: initial clusters were refined through repeated reading and comparison across questions. Frequencies of responses per category were tabulated to provide a sense of the most prevalent themes, while the primary goal was to capture the range of perspectives rather than to produce statistical generalizations.

Comparative Analysis. Comparisons were conducted both within a project phase (e.g., advantages vs. disadvantages of scrums at mid-project) and across project phases (e.g., pre-project expectations vs. mid-project perceptions vs. post-project reflections). This longitudinal comparison allowed us to trace how student sentiment evolved over time. Initially, comments framing scrums as pressure often shifted in later reflections toward recognition of their value for coordination and professional preparation. Such comparisons highlighted movement from external compliance to internalized ownership of reflective practices.

Deriving Actionable Insights. Analysis concluded by translating thematic findings into actionable instructional practices. Coded themes were interpreted through the lens of reflective practice theory (e.g., Schön's distinction between reflection-on-action and reflection-in-action) and aligned with prior research in software engineering education on scaffolding, feedback, and authenticity. Each actionable insight was grounded in three layers: (1) student reflections (with representative quotes and counts), (2) thematic analysis of patterns across project stages, and (3) connections to existing literature. This triangulation ensured that resulting practices were both empirically grounded and pedagogically transferable.

5 Learning from the Reflection of Other Teams

Students were asked to answer the following questions at the beginning and at the end of the course:

- Q1. How much have you learned from classmates that are not part of your group in past classes? Please give an example.
- Q2. Describe the effect of seeing other teams' or classmates' presentations have on your own (or your team's) work. What if their work is better than yours? Or worse than yours?
- Q3. How does it help you improve your own (or your team's) work when you see the work of other students or teams?

A comparison of student responses to these questions at the beginning (Series A, 72 responses) and the end of the course (Series B, 48 responses) reveals a clear progression in how students engaged with peer learning opportunities. Early in the course, students largely viewed peer work as a benchmark against which to measure their own. Comments frequently emphasized whether another team's work appeared "better" or "worse," with one student noting, "I think when I see other groups perform more coherently." These observations were often surface-level, focusing on presentation quality or the presence of features, rather than on transferable practices. Several students reported minimal benefit, stating simply, "I have not learned from classmates very often."

By the end of the project, the tone of responses had shifted toward a more reflective and applied stance. Students described adopting specific practices from their peers, as in the comment, "From another group got the idea how to get going running the ROS." Others reflected on how peer presentations informed their own teamwork strategies: "Our attempts to increase asynchronous work were based on the teams who said they didn't do enough synchronous work – they assigned tasks and then never reconvened. We still meet all the time, but do a lot more independent work on our particular areas." This marks a significant change from individual benchmarking to collective learning and adaptation.

Learning outcome specificity also increased. While early responses often referred vaguely to being "inspired" by peers, later ones pointed to transferable practices such as using design patterns: "It gives us interesting concepts of how which design patterns can be applied to which scenario." Similarly, students moved from rejecting peer presentations as too general (e.g. "I don't seem to get much out of the classroom except for projects. Just broad concepts, no concrete knowledge") to critiquing knowledge they most valued: "I was more interested in learning about the interactions with the

client or the planning for future sprints." This indicates growing ability to discern and prioritize aspects of professional practice.

Finally, students began to see peer learning not only as a source of technical ideas but also as a perspective on shared challenges. As one student summarized, "It just help put in perspective what we could have done or see how similar our struggles are regardless if one is undergrad or grad student." This evolution suggests that students were developing empathy and professional awareness alongside technical and process-oriented insights.

5.1 Actionable Insights

Scaffold, compare, and adopt peer practices. At the start of the project, require instructor-facilitated reflections after peer presentations that capture both what to adopt and what to avoid. About 45 Series A responses showed that students defaulted to vague judgments (e.g. "I think when I see other groups perform more coherently") or claimed they learned little, such as, "I have not learned from classmates very often." By Series B, nearly 20 responses described concrete adoption of practices: "From another group got the idea how to get going running the ROS." To maximize the effect, instructors should frame peer observations as structured comparisons of trade-offs, then close the loop by requiring each team to record one specific practice they will adopt in the next sprint. The effect is that peer observation becomes a cycle: notice → analyze → implement, rather than remaining a spectator exercise.

Why it matters: Without guidance, students rarely move past simple benchmarking. Structuring peer observation as a cycle of reflection and adoption mirrors agile's inspect-and-adapt rhythm and Schön's movement from reflection-on-action to reflection-in-action [3, 27, 37]. This helps students internalize reflection as a habit that produces concrete improvements.

Frame peer observation as team checkpoints. In early sprints, require teams to document at least one planned process change (e.g., coordination, meeting structure, backlog refinement) after peer presentations, then revisit those changes at subsequent retrospectives. About 30 Series A responses focused on personal reassurance, such as, "When I see other teams' work I am relieved that others might be as overwhelmed or confused as I am." By Series B, over 15 responses described team-level changes, e.g., "Our attempts to increase asynchronous work were based on the teams who said they didn't do enough synchronous work – they assigned tasks and then never reconvened." Framing peer reviews as checkpoints for progress and challenges creates alignment across teams [9, 15]. The effect is that students stop thinking only as individuals and start engaging in collective improvement, both within and across teams.

Why it matters: Agile emphasizes transparency and adaptation at the team level. Structuring presentations as alignment checkpoints reinforces these values, ensuring that peer learning translates into better collaboration and workflow. This approach reduces isolation and builds habits of accountability across multiple teams.

Require evidence-based takeaways from presentations. Provide templates for reflections that prompt students to identify one element of another team's work, explain its effectiveness, and describe adaptation possibilities. About 25 Series A responses stayed impressionistic (e.g. "If I see something cool, or creative, then it is inspiring to me") or focused on flashy features like multiplayer add-ons. By Series B, ≈ 18 responses pointed to concrete improvements such as noticing more credible client demos or clearer explanations. Students learn to ground reflections in observable evidence rather than vague impressions, strengthening their ability to critically evaluate practices.

Why it matters: Evidence-based reflection helps students distinguish between polished artifacts and genuinely effective practices. This aligns with professional practice, where decisions are evaluated on evidence, outcomes, and trade-offs [28, 37]. Embedding this discipline prepares students for critical analysis in both technical and process domains.

Teach students to identify transferable lessons. Explicitly model in class how to distinguish between project-specific details and practices that can generalize. About 15 Series B responses expressed frustration that peer work was "too specific" to be useful, e.g., "Most of the time, the retrospectives and demos were very project specific... I was more interested in learning about the client." By highlighting transferable lessons, such as, "This team's demo technique could apply to all projects, regardless of domain," instructors help students practice abstraction. The effect is that students learn to extract enduring principles rather than confining lessons to one context.

Why it matters: Transferability is a cornerstone of reflective practice. Without support, students often reflect only on immediate technical details and miss broader applications [3, 27]. By modeling this process, instructors help students cultivate metacognitive skills necessary for lifelong learning.

Integrate professional skills into peer critique. Require students to comment on how teams handle professional aspects such as client communication, sprint planning, or backlog management. About 20 Series A responses dismissed presentations as unhelpful, e.g. "I don't seem to get much out of the classroom except for projects. Just broad concepts, no concrete knowledge." By Series B, more than a dozen responses showed interest in professional practice, e.g., "I was more interested in learning about the interactions with the client or the planning for future sprints." Adding professional skills to critique criteria ensures that students recognize the key components of software engineering and how they fit into the overall process, ultimately preparing them for workplace readiness. This broadened perspective allows students to begin to see software engineering not just as coding, but as a interplay of communication, coordination, client engagement, and opportunities for their own continual process improvement.

Why it matters: Professional success in software engineering depends on both technical and human factors. Embedding professional skills into peer critique helps students reflect on the full scope of software practice, reinforcing evidence from the literature that reflective practice improves teamwork and project management competence [8, 14].

Provide reflective prompts that evolve over time. Sequence prompts to match students' developmental stage. Early on, ask descriptive questions, e.g. "What did you notice?" or "What surprised you?" Later, raise the bar: "What principle from this presentation applies to your project?" or "How will you adapt this idea in your next sprint?" About 40 Series A responses stayed descriptive (e.g. "If I

see something cool, or creative, then it is inspiring to me") while ≈ 15 Series B responses demonstrated deeper reasoning, pointing to why peers' coordination or demo choices were effective. Evolving prompts gradually build students' reflective capacity, helping them transition from noticing to analyzing to generalizing.

Why it matters: Novices benefit from scaffolds that grow in complexity as reflective skills mature. This aligns with Schön's framework of reflection-on-action evolving into reflection-in-action [31, 37]. Structured progression ensures students move beyond surface-level reflection to develop generalizable insights.

Normalize struggle as part of learning. Encourage teams to present both successes and obstacles, and facilitate discussions that highlight lessons learned from difficulties. About 25 Series A responses minimized learning from peers, such as, "I have not learned from classmates very often." By Series B, at least 10 responses explicitly valued seeing common struggles, e.g., "It just help put in perspective what we could have done or see how similar our struggles are regardless if one is undergrad or grad student." Normalizing challenges helps students see setbacks as data for improvement rather than as failures. The effect is reduced anxiety, stronger empathy, and the development of resilience, which represent key soft skills in professional engineering.

Why it matters: Reflective practice thrives in environments where students feel safe to discuss struggles openly. Literature emphasizes that making difficulties explicit enhances learning and builds professional awareness [9, 15]. By normalizing struggle, instructors cultivate a classroom culture that mirrors professional practice, where continuous improvement depends on confronting problems candidly.

6 Learning from Teams' Own Reflections

Right after the first sprint, students were asked these questions (38 responses received):

- Q4a. What are the advantages of having weekly check-ins in class such as weekly scrums or sprint retrospectives?
- Q5a. What are the disadvantages of having weekly check-ins in class?

Then, at the end of the project, students were asked (48 responses):

- Q45b. How did your team benefit from your own regular scrums and sprint retrospectives?

Midproject, students generally valued weekly scrums and retrospectives for providing accountability, structure, and timely feedback from instructors and peers. Several emphasized that these sessions prevented procrastination and maintained steady progress: "It makes sure everyone is making some progress." Others noted transparency benefits, describing how meetings kept everyone aligned on goals and schedules. However, drawbacks were also reported. The most common concern was pressure to demonstrate visible progress each week, which some found stressful: "It can be nerve-wracking when you don't have much new to show." A smaller number felt the time spent on check-ins reduced opportunities for direct project work.

By the end of the project, however, reflections became more concrete and personalized. Students highlighted how their own scrums and retrospectives kept teams aligned with deliverables and supported coordination: "They kept us on track and made sure nothing slipped through the cracks." Others pointed to the motivational value of these rituals: "It was good motivation to keep working consistently." Many also recognized the professional relevance of these practices, noting gains in teamwork and communication skills such as presentation discipline and clearer collaboration routines. Although a minority of responses described ineffective dynamics—"Our scrums didn't help because our team wasn't working well together"—the prevailing sentiment was that scrums and retrospectives had become useful tools for both project success and professional preparation.

These reflections illustrate a clear progression from compliance to ownership. What began as externally mandated structures were ultimately reframed by students as authentic practices that supported both their immediate project work and their preparation for professional environments.

6.1 Actionable Insights

Scaffold reflective practices early, then transition to autonomy. Require structured, instructor-led scrums and retrospectives initially to establish accountability and shared habits. About 12 mid-project responses described these practices as mechanisms to "force progress" and ensure everyone was making contributions. By project end, 7 responses emphasized how their own scrums "kept us on track" with deliverables. This shift illustrates movement from compliance with externally imposed routines to ownership of practices as useful tools for teamwork and delivery. This creates greater consistency in progress, stronger team coordination, and adoption of reflective habits that persist beyond the course.

Why it matters: Reflective practice is not instinctive for novices. Schön describes the progression from reflection-on-action (external, after-the-fact) to reflection-in-action (internal, real-time) as critical for professional growth [15, 31]. Early scaffolding provides the structure students need, while later autonomy fosters independence and a sense of professional identity.

Address stress and normalize early discomfort. Mid-project reflections revealed concerns that scrums were stressful or a poor use of time. Sixteen responses cited time pressure and ten mentioned lost work time, with students describing check-ins as "time consuming" and "technically low yield." By the end of the course, these complaints faded, replaced by descriptions of concrete benefits such as improved coordination and motivation. This suggests that initial resistance can be reframed as temporary discomfort on the path to professional readiness. The effect is reduced student frustration, greater resilience in managing deadlines, and recognition of accountability as part of professional practice.

Why it matters: Prior et al. note that reflection is "hard to do and equally hard to teach," often requiring explicit support [27]. Normalizing early stress helps students reinterpret the discomfort of weekly rituals not as wasted effort but as rehearsal for industry norms where regular reporting and accountability are unavoidable.

Frame scrums and retrospectives as authentic professional practices. Students were more engaged once they recognized scrums as preparation for professional work. While only 4 mid-project responses

tied check-ins to professional practice, end-of-project reflections increasingly described them as essential for aligning with deliverables and developing skills such as communication and presentation. One student noted that "learning presentation skills was very valuable," while another said retrospectives provided "very good motivation." The effect is stronger buy-in, improved skill transfer, and the development of professional identity.

Why it matters: Research shows that realism and industry alignment are crucial for bridging the education–practice gap [8, 25]. Hazzan and Dubinsky also argue that reflective modes of thinking deepen students' understanding of teamwork and communication in authentic settings [14, 15]. Framing scrums explicitly as industry practice helps students see beyond classroom requirements to their relevance in the workplace.

Enhance reflection through facilitation and feedback. Students consistently valued receiving and acting on feedback. Eleven mid-project responses highlighted feedback from instructors or peers as check-ins' most useful aspect. Eventually, students described scrums as spaces where feedback improved team performance and sustained motivation. However, three responses reported ineffective scrums due to poor collaboration, describing experiences as "terrible" or unproductive. Providing feedback-rich environments alongside light coaching for struggling teams ensures inclusivity and depth of reflection. The effect is deeper learning from peers, improved team processes, and reduced risk of disengagement.

Why it matters: Bull and Whittle argue that reflection thrives in feedback-rich studio cultures built on "constant questioning" [1, 3]. At the same time, Hundhausen et al. found that most students engage in reflection only when explicitly prompted [16]. Embedding feedback loops and offering light coaching sustains reflection and helps students develop habits of critical dialogue essential to professional practice.

7 The Role of the Project Client/Partner

Students were asked these questions after completing the first sprint (38 responses were received):

- Q6a. What are the advantages of having weekly/regular check-ins with the project client?
- Q7a. What are the disadvantages of having weekly/regular check-ins with the project client? What are the advantages of having weekly check-ins in class such as weekly scrums or sprint retrospectives?

Then, at the end of the project, students were asked (48 responses):

- Q67b. Did regular check-ins (in class or with the client) help you improve your work? If so, how?

At the midpoint of the project, students described weekly client check-ins as a valuable mechanism for clarifying requirements, reducing misunderstandings, and keeping the project aligned with client expectations. One student noted, "It gave us a chance to ask questions and make sure we were building the right thing." Several also emphasized the motivational role of accountability: "Knowing we had to show something every week kept us on track."

However, drawbacks were reported. The most frequent concern was additional workload and stress from preparing for meetings, which some felt reduced development time. As one student explained, "It can be stressful if you don't have enough new progress to share." Others pointed to logistical challenges such as scheduling or clients being unprepared, and a few noted that meetings sometimes felt repetitive when little had changed.

By the end of the project, however, perspectives shifted. Students overwhelmingly acknowledged that regular check-ins had strengthened both their projects and their skills. Formative feedback from clients was repeatedly translated into revisions and refinements, and subsequent reflection helped students connect these lessons to professional practice making these meetings central to iterative improvement. One student wrote, "Client feedback helped us fix problems quickly and improve our design." Beyond technical outcomes, many emphasized professional growth in communication, presentation, and stakeholder management: "The weekly meetings gave us practice explaining our work to non-technical people."

What began as a practice that sometimes felt like an obligation or distraction was ultimately reframed as authentic preparation for industry. The features that caused stress in the middle of the semester, including external accountability, the need to prepare and present, and the requirement to engage regularly with a client, were later recognized as essential elements of professional practice.

7.1 Actionable Insights

Frame check-ins as formative feedback that drives reflection, not evaluation. Regular client meetings worked best when students saw them as opportunities to clarify requirements, surface misunderstandings, and adjust direction, rather than as high-stakes assessments. About 15 mid-project responses highlighted the value of clarification (e.g. "Get questions answered and get clarification on project details") while by the end $\approx$ 20 responses pointed to how client input directly shaped revisions and improved outcomes, e.g. "Client check-ins helped a lot [...] improved our ideas and implementation." Instructors can reinforce this by explicitly framing check-ins as feedback sessions that generate material for reflection, where missteps are treated as learning opportunities rather than failures. Why it matters: Formative, dialogic feedback fosters reflection-in-action, where students adapt their work in real time. This approach is consistent with reflective practice theory [30, 31] and research on formative assessment in computing education [10, 17].

Balance frequency with flexibility. While weekly client meetings provided structure and accountability, $\approx$ 20 students reported stress when little progress had been made. "It can be stressful if you know the check-in is coming up and you haven't gotten enough done," one student explained. Yet eventually, $\approx$ 5 responses reframed the cadence positively, noting that "Check-ins provided a means of getting back on task and eliminating distractions." Instructors can maximize value by scaffolding: maintain weekly touchpoints early to build habits, then allow flexibility (e.g., shorter updates or alternating formats) as teams demonstrate maturity. This keeps focus on productive improvement rather than meeting preparation fatigue. Why it matters: Balancing cadence ensures check-ins sustain forward momentum without becoming burdensome. Research on scaffolding reflection highlights structured accountability that gradually transitions into self-directed practice [3, 27].

Prepare clients to give effective feedback. The quality of client feedback shaped how much check-ins improved student work. About 8 responses described frustrations with clients being unavailable, inconsistent, or unclear (e.g. "We had to reiterate information we already provided") while others highlighted the opposite, noting that "Clients' feedback gave us better ideas on how to implement our project." Instructors can enhance value by providing clients with a short rubric or guiding questions, ensuring they are ready to give focused, actionable input. This preparation increases the likelihood that meetings produce concrete improvements in design and implementation, creating high-quality material for reflection, helping students link external perspectives to their own decisions. Why it matters: Effective client engagement mirrors industry practice and narrows the gap between education and the workplace. Research underscores the importance of authentic stakeholder involvement in project-based learning [8, 25].

Make professional skill development explicit. By semester's end, ≈ 7 responses emphasized how check-ins improved communication, presentation, and stakeholder management, which are among the skills that directly supported better project outcomes. One student noted that "Weekly meetings gave us guidance on how to handle ourselves during the development process," while another reflected, "It was very important to keep the client involved." Early in the course, some students dismissed meetings as "repetitive," but later they recognized that being required to explain progress clearly sharpened both their thinking and their deliverables. Instructors should underscore that client communication is not only about updating requirements, but also about practicing transferable skills critical to employability and then reflect on how to improve their approach. Why it matters: Reflection-in-practice includes the social dimensions of professional work. Studio-based SE education shows that communication and client-facing skills are integral to producing quality software [1, 15, 16].

Reframe redundancy as reflection. Only ≈ 2 responses labeled weekly meetings as redundant when little progress had occurred, e.g. "It can feel redundant if there are no new questions and things are proceeding as expected." Yet end-of-project reflections showed that even "repetitive" conversations often uncovered subtle issues or reinforced alignment: ≈ 20 responses highlighted iterative improvement. One student explained, "Client check-ins helped a lot, class check-ins were also very helpful because sometimes finding time outside class was difficult." Instructors can help students reframe redundancy as purposeful: articulating work-in-progress, revisiting assumptions, and refining their approach. Why it matters: Iteration is central to reflective practice. Schön's ladder of reflection emphasizes that revisiting and re-articulating progress builds deeper understanding and better outcomes [3, 14, 30].

Harness external accountability to drive steady progress. Across responses, ≈ 7 students mid-project and ≈ 5 at the end emphasized that the mere presence of client check-ins motivated them to maintain progress. Early on, this was described as external pressure (e.g."This is the most important and ensures we are creating the correct code that they want") but by the end it was reframed as constructive accountability (e.g. "Check-ins provided a means of getting back on task and eliminating distractions"). Instructors can intentionally frame client updates as authentic accountability structures, shifting the emphasis from grades to professional responsibility. Why it matters: External accountability scaffolds reflective habits by requiring visible progress, which students eventually internalize as self-regulation. Research on reflective studios shows that external critique and accountability help transform external demands into internalized professional standards [1, 9, 28].

8 Reflecting on Transferable Practices

After students had an opportunity hear other teams reflect on their work, we asked them (and received 38 responses):

- Q8a. Are there any specific practices you or your team have adopted after seeing other teams' presentations? If so, what are they and why did you adopt them?

At the end of the semester, we followed up with this question (48 responses):

- Q8b. How did you benefit from hearing other team's regular scrums and sprint retrospectives?

At the midpoint of the project (Q8a), responses suggested a tentative and surface-level engagement with peer learning. A considerable number reported "not really" or expressed uncertainty, indicating that they had not yet found clear value in comparing their work to others. Among those who did adopt new practices, most changes were highly visible and easy to replicate: adding live demos, streamlining slides, adjusting communication routines, or experimenting with simple workflow adjustments such as role specialization or pair programming. As one student explained, "Having a demo seems to be a really good thing to have for a presentation so people can see the work we've done." Overall, the sentiment was cautious: students noticed differences among teams but did not yet interpret those differences as opportunities for broader learning.

By the end of the project (Q8b), reflections had shifted markedly. Students emphasized process-oriented insights rather than surface-level practices, often noting the value of hearing about peers' challenges. One student wrote, "Hearing the planning and progress of other teams during the sprint retrospective was very insightful." Others described how seeing alternative strategies sparked new ideas: "It gave me some interesting ideas of how we can improve our project." A smaller group mentioned benchmarking benefits, such as using peers' updates to calibrate progress without slipping into competition: "We benefited to see where other teams are and how we can speed up or slow down our work." Unlike at mid-project, very few dismissed these activities as irrelevant; instead, students consistently recognized the value of reflective dialogue itself.

Together, these responses illustrate progression from surface-level imitation to authentic engagement with reflection. Initially, peer presentations were treated as spectator events or opportunities to copy visible practices. Over time, however, students appreciated scrums and retrospectives as spaces for shared learning, accountability, and mutual support. The shift from cautious adoption to genuine reflection underscores that meaningful peer learning requires time, scaffolding, and explicit instructor support.

8.1 Actionable Insights

Start with artifacts, move toward processes. Encourage students to begin by adopting visible practices from peers and then guide them toward reflecting on underlying processes and strategies. At mid-project, ≈ 10 of 38 responses described adopting visible practices such as adding demos, improving slide clarity, or adjusting communication routines. One team noted, "Having a demo seems to be a really good thing to have for a presentation so people can see the work we've done." Another wrote, "Just ensuring we communicate more with each other and with the clients to ensure everyone is on the same page." By the end, 11 of 48 responses shifted to valuing process learning: understanding peers' planning, strategy, and lessons learned from setbacks. As one student put it, "Hearing the planning and progress of other teams during the sprint retrospective was very insightful." Another said, "It gave me some interesting ideas of how we can improve our project." Why it matters: This progression shows how students move from copying surface-level tactics to internalizing deeper reflective practices. Scaffolding reflection first on artifacts, then steering toward reasoning and process, echoes Schön's distinction between reflection-on-action and reflection-in-action [3, 14, 31].

Make struggles a shared learning resource. Ask teams to discuss not only their successes but also their failures and difficulties, and highlight these as opportunities for peer learning. By the end of the project, at least 11 responses emphasized learning from others' difficulties, e.g. how teams handled obstacles or regrouped after mistakes. Students highlighted the value of candor: "Understand the team dynamics and strategy for problem solving." Another concluded, "What works and what doesn't." In contrast, mid-project reflections rarely mentioned struggles; most focused only on successes to imitate. Why it matters: Encouraging candid discussion of struggles normalizes setbacks as part of professional practice. Research on reflective studios shows that authentic reflection often emerges when learners analyze missteps, not just successes [9]. Instructors can strengthen retrospectives by explicitly asking teams to share "what didn't work" and framing those as learning assets.

Frame peer comparisons as calibration, not competition. Guide students to use peer progress as a way to calibrate their own pace and strategy rather than as a competitive measure. At the end, 3 responses described using peers' progress to benchmark their own, e.g. "We benefited to see where other teams are and how we can speed up or slow down our work," and "Compare to other team we were way ahead in completing the requirements." Mid-project comments, by contrast, were vague ("clarity with their information") and showed less purposeful calibration. Why it matters: Healthy benchmarking motivates without demoralizing. Framing peer learning as calibration supports accountability and mutual progress, consistent with formative assessment principles [17]. Instructors can reinforce this by reminding students that comparisons are diagnostic tools for adjustment, not rankings of success.

Use reflection prompts to deepen insights. Provide structured prompts that push students to move beyond vague responses and identify specific lessons from peers. Across both questions, nearly half of all responses (22 of 38 in Q8a, 25 of 48 in Q8b) were vague: "Not really," "Not sure," or simply "some." These responses suggest that students often struggled to articulate concrete lessons without explicit guidance. When prompted, however, they produced more specific insights, such as "After the first sprint and seeing what other teams did, we kept team members in specific roles." Why it matters: Students need structured guidance to move beyond vague acknowledgment. Formative assessment research shows that explicit prompts and criteria enhance the quality of reflection and self-regulation [10, 35]. Instructors can scaffold reflection by requiring teams to record one practice they would adopt and one pitfall they would avoid after each peer session.

Highlight reasoning, not just results. Require teams to explain rationale behind choices so peers can learn from decision-making rather than only outcomes. Eventually, 8 responses credited retrospectives with sparking "new ideas" or "creativity." One student reflected, "I have new ideas on how to implement patterns into my own project. It sparked my creativity and passion to develop." Another said, "It was interesting seeing the ways other groups were doing things." Mid-project, only a handful tied takeaways to underlying reasoning, such as, "We kept team members in specific roles… they could assist when another user story relating to that topic comes up." Why it matters: Students learn more hearing why peers made decisions, not just what they produced. Studio-based research shows reflective learning deepens when teams externalize rationale [15, 28]. Instructors can require teams to explain trade-offs and alternatives in retrospectives, transforming peer updates into reasoning lessons.

Reinforce the cumulative value of reflection. Explain to students that the benefits of scrums and retrospectives compound over time, even if early sessions feel repetitive or unclear. At mid-project, 6 responses explicitly dismissed the activity: "Not really," "We have not adopted any specific practices from other groups." By the end, only 1 of 48 responses said it wasn't helpful. Instead, students increasingly recognized benefits: "It gave me some interesting ideas of how we can improve our project," and "Hearing the planning and progress… was very insightful." Why it matters: Repetition and integration into the course rhythm help students see reflection as worthwhile. Literature on metacognitive scaffolding confirms that reflection rarely arises spontaneously and must be cultivated with consistent prompts [16, 37]. Instructors should emphasize that scrums and retrospectives are professional habits, not course rituals, and that their value compounds across a semester.

9 Cross-Cutting Instructional Principles

The four sets of insights described above identified actionable practices specific to their context. Taken together, they reveal broader design principles for embedding reflective practice into project-based software engineering courses. Below, we synthesize these insights into five cross-cutting principles, each grounded in our findings and supported by the literature.

Scaffold early, then transition to autonomy. Initially, students relied on structured prompts, instructor facilitation, and explicit reporting to sustain reflection. In peer reviews, prompts made students focus beyond superficial polish (Q1-3); in scrums, instructor facilitation reduced stress (Q45); and in client meetings, weekly check-ins created accountability (Q67). However, with experience, they reframed these structures as owned tools, e.g. using retrospectives to adapt

processes and client meetings to build stakeholder trust. Scaffolding ensures students learn to reflect before expecting them to do so independently. As scaffolds fade, students gain ownership of reflection as a professional habit, aligning with Schön's movement from "reflection-on-action" to "reflection-in-action" [30, 31]. Prior et al. emphasize that without scaffolding, reflection rarely occurs spontaneously [27], while Tariq shows that metacognitive scaffolding improves students' ability to make better design decisions [37].

Make reflection feedback-rich and grounded in authentic audiences. As shown in Figure 1, the semester-long rhythm alternates between feedback opportunities (client and peer input) and reflective practices (scrums, retrospectives, reports). This interplay was central to students recognizing reflection as a professional habit rather than a classroom exercise. Each takeaway set highlighted the value of feedback: peers offering inspiration and calibration (Q1-3, Q8), instructors modeling professional standards (Q45), and clients providing formative feedback tied to real project needs (Q67). When reflection is dialogic and tied to authentic audiences, students shift from private rumination to actionable improvement. Peer and client feedback situates reflection in real-world accountability, enhancing communication and teamwork skills that industry consistently finds lacking in graduates [25]. Studio literature stresses the role of "constant questioning" in cultivating reflective cultures [3], and formative peer review research shows how iterative feedback builds judgment and collaborative learning [35].

Frame reflection as professional preparation, not classroom compliance. Early responses described reflective activities as stressful requirements; by semester's end, students recognized scrums as coordination tools (Q45), client meetings as stakeholder management practice (Q67), and peer presentations as opportunities for professional calibration (Q1-3, Q8). Reframing these activities as rehearsals for professional practice increases student buy-in and motivation. Students come to see reflective practice as an essential competency for industry, not an academic hoop to jump through. This echoes Hazzan's argument that reflective practice is central to developing professional identity [14, 15], and aligns with studies on realism in projects, which show that authentic contexts close the industry–education gap [8].

Shift focus from artifacts to processes and transferable skills. Students began by comparing surface features, such as slide polish in presentations, visible task completion in scrums, or progress updates to clients. By the end, they were naming deeper lessons: why a design trade-off was made (Q1-3), how team routines reduced stress (Q45), how to negotiate priorities with clients (Q67), and what struggles were common across teams (Q8). By emphasizing processes and reasoning, reflection builds judgment that transfers across contexts. Students learn not only what was built, but how and why, which showcase the skills that endure beyond a single course project. Razavian et al. demonstrate that reflective questioning strengthens design reasoning [28], while Hazzan stresses that reflection on both cognitive and social processes enhances professional performance [14, 15].

Normalize struggle and iteration as learning opportunities. In every context, students identified value in seeing peers' and their own struggles: learning from common pitfalls in peer reviews (Q1-3, Q8), reducing anxiety when scrums revealed shared challenges (Q45), and treating repeated client check-ins as purposeful iteration rather than redundancy (Q67). The repeated cycles of critique and reflection evident in scrums, retrospectives, and client meetings were crucial for students to move beyond superficial observations to deeper understanding of design trade-offs and process improvement. Normalizing difficulty helps students build resilience and empathy. Struggles become data for improvement rather than signs of failure. Hundhausen et al. found that student reflection rarely arises spontaneously and often centers on challenges [16]. Dors et al. show that reflective practice in studios fosters idea generation precisely through cycles of critique and iteration [9]. By making setbacks explicit, instructors cultivate the professional mindset of continuous improvement.

10 Conclusion

This study reported on a decade of embedding structured reflection into a software engineering capstone course sequence. Across scrums, retrospectives, and written reports informed by formative feedback from peer presentations, client check-ins, students' views of reflection shifted from compliance with externally imposed rituals to ownership of authentic professional practices. Their reflections revealed recurring patterns: initial stress gave way to recognition of accountability as preparation for industrial practice, surface-level comparisons evolved into process-focused learning, and client meetings reframed from burdens to opportunities for professional growth. Our central lesson is that *reflection must be designed with authentic, feedback-rich projects*; together they transform reflection from an academic exercise into a professional habit.

From these experiences, we derived a set of actionable instructional practices and synthesized them into five cross-cutting design principles. Together, they show how deliberate scaffolding, authentic audiences, and structured prompts can transform reflection from an afterthought into a professional habit that strengthens both technical and human-centered competencies. For educators, the implication is clear: *reflection should not be treated as an optional add-on, but as a formative core of project-based courses.*

Looking forward, these results suggest two avenues for further work. First, more systematic study is needed to understand how different scaffolds (rubrics, prompts, and feedback mechanisms) affect reflection quality across diverse student populations and institutions. Second, reflection research in software engineering education would benefit from connecting more directly to learning analytics, enabling longitudinal insights into how reflective practices shape professional identity and career readiness.

By embedding reflection into daily project work, we can help students develop not only stronger projects but also the reflective capacities that define resilient, adaptive software professionals.

Acknowledgments

This work was supported in part by NSF award 2315322.

References

[1] Aline Andrade, Alessandro Maciel Schmidt, Tania Mara Dors, Regina Albuquerque, Fabio Binder, Dilmeire Vosgerau, Andreia Malucelli, and Sheila Reinehr. 2023. Education, Innovation and Software Production: the contributions of the

Reflective Practice in a Software Studio. *Journal of Software Engineering Research and Development* 11, 1 (2023), 7–1.

[2] Paul Black and Dylan Wiliam. 1998. Assessment and classroom learning. *Assessment in Education: principles, policy & practice* 5, 1 (1998), 7–74.

[3] Christopher N Bull and Jon Whittle. 2014. Supporting reflective practice in software engineering education through a studio-based approach. *IEEE software* 31, 4 (2014), 44–50.

[4] David Carless. 2007. Learning-oriented assessment: conceptual bases and practical implications. *Innovations in education and teaching international* 44, 1 (2007), 57–66.

[5] David Carless. 2015. *Excellence in university assessment: Learning from award-winning practice*. Routledge.

[6] David Carless and David Boud. 2018. The development of student feedback literacy: enabling uptake of feedback. *Assessment & Evaluation in Higher Education* 43, 8 (2018), 1315–1325.

[7] Kwangsu Cho and Christian D Schunn. 2007. Scaffolded writing and rewriting in the discipline: A web-based reciprocal peer review system. *Computers & Education* 48, 3 (2007), 409–426.

[8] Orges Cico, Letizia Jaccheri, Anh Nguyen-Duc, and He Zhang. 2021. Exploring the intersection between software industry and Software Engineering education-A systematic mapping of Software Engineering Trends. *Journal of Systems and Software* 172 (2021), 110736.

[9] Tania Mara Dors, Frederick MC Van Amstel, Fabio Binder, Sheila Reinehr, and Andreia Malucelli. 2020. Reflective Practice in Software Development studios: findings from an ethnographic study. In *2020 IEEE 32nd conference on software engineering education and training (CSEE&T)*. IEEE, 1–10.

[10] John K Estell and Susannah Howe. 2017. Development and use of a client interaction rubric for formative assessment. (2017).

[11] Vahid Garousi. 2009. Applying peer reviews in software engineering education: An experiment and lessons learned. *IEEE Transactions on Education* 53, 2 (2009), 182–193.

[12] Aida Guerra, Ronald Ulseth, and Anette Kolmos. 2017. *PBL in engineering education: international perspectives on curriculum change*. Springer.

[13] John Hamer, Helen Purchase, Andrew Luxton-Reilly, and Paul Denny. 2015. A comparison of peer and tutor feedback. *Assessment & Evaluation in Higher Education* 40, 1 (2015), 151–164. arXiv:https://doi.org/10.1080/02602938.2014.893418 doi:10.1080/02602938.2014.893418

[14] Orit Hazzan and Yael Dubinsky. 2009. Reflection in software engineering education. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*. 691–692.

[15] O. Hazzan and J.E. Tomayko. 2004. Reflection processes in the teaching and learning of human aspects of software engineering. In *17th Conference on Software Engineering Education and Training, 2004. Proceedings.* 32–38. doi:10.1109/CSEE.2004.1276507

[16] Christopher Hundhausen, Phill Conrad, Olusola Adesope, Ahsun Tariq, Samir Sbai, and Andrew Lu. 2023. Investigating reflection in undergraduate software development teams: An analysis of online chat transcripts. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 743–749.

[17] Alastair Irons and Sam Elkington. 2021. *Enhancing learning through formative assessment and feedback*. Routledge.

[18] David A Kolb. 2014. *Experiential learning: Experience as the source of learning and development*. FT press.

[19] Stan Kurkovsky. 2023. Student Reflections on Service-Learning in Software Engineering and Their Experiences with Non-technical Clients. In *Proceedings of the ACM Conference on Global Computing Education Vol 1* (Hyderabad, India) *(CompEd 2023)*. Association for Computing Machinery, New York, NY, USA, 112–118. doi:10.1145/3576882.3617929

[20] Stan Kurkovsky, Michael Goldweber, Chad A. Williams, and Nathan Sommer. 2025. Scaffolded Projects for the Social Good: Broadening Participation in Service Learning for Computing Curricula. In *Proceedings of the ACM Global on Computing Education Conference 2025 Vol 1* (Gaborone, Botswana) *(CompEd 2025)*. Association for Computing Machinery, New York, NY, USA, 294–300. doi:10.1145/3736181.3747128

[21] Stan Kurkovsky, Chad A. Williams, Mikey Goldweber, and Nathan Sommer. 2024. External Projects and Partners: Addressing Challenges and Minimizing Risks from the Outset. In *Proc. 2024 Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024), July 8–10, 2024, Milan, Italy* (Milan, Italy) *(ITiCSE '24)*. ACM, New York, NY, USA, 7 pages. doi:10.1145/3649217.3653593

[22] George G Mitchell and James Declan Delaney. 2004. An assessment strategy to determine learning outcomes in a software engineering problem-based learning course. *International Journal of Engineering Education* 20, 3 (2004), 494–502.

[23] Jennifer A Moon. 2013. *Reflection in learning and professional development: Theory and practice*. Routledge.

[24] David J Nicol and Debra Macfarlane-Dick. 2006. Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education* 31, 2 (2006), 199–218.

[25] Damla Oguz and Kaya Oguz. 2019. Perspectives on the gap between the software industry and the software engineering education. *Ieee Access* 7 (2019), 117527–117543.

[26] Marian Petre, Kate Sanders, Robert McCartney, Marzieh Ahmadzadeh, Cornelia Connolly, Sally Hamouda, Brian Harrington, Jérémie Lumbroso, Joseph Maguire, Lauri Malmi, Monica M. McGill, and Jan Vahrenhold. 2020. Mapping the Landscape of Peer Review in Computing Education Research. In *Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education* (Trondheim, Norway) *(ITiCSE-WGR '20)*. Association for Computing Machinery, New York, NY, USA, 173–209. doi:10.1145/3437800.3439207

[27] Julia Prior, Samuel Ferguson, and John Leaney. 2016. Reflection is hard: teaching and learning reflective practice in a software studio. In *Proceedings of the Australasian Computer Science Week Multiconference*. 1–8.

[28] Maryam Razavian, Antony Tang, Rafael Capilla, and Patricia Lago. 2016. In two minds: how reflections influence software design thinking. *Journal of Software: Evolution and Process* 28, 6 (2016), 394–426.

[29] D Royce Sadler. 1989. Formative assessment and the design of instructional systems. *Instructional science* 18, 2 (1989), 119–144.

[30] Donald A Schön. 1987. Educating the reflective practitioner. (1987).

[31] Donald A Schön. 1992. *The reflective practitioner: How professionals think in action*. Routledge.

[32] Valerie J Shute. 2008. Focus on formative feedback. *Review of educational research* 78, 1 (2008), 153–189.

[33] Rahul Simha and Helen Wright. 2025. *The CRA Practitioner-to-Professor (P2P) Survey: Towards Periodic Feedback from Industry to Academic Computing Programs*. Technical Report. Computing Research Association. https://cra.org/industry/wp-content/uploads/sites/9/2025/06/FINAL-P2P-Report-June-2025.pdf Final report, June 2025.

[34] Robbie Simpson and Tim Storer. 2017. Experimenting with realism in software engineering team projects: An experience report. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE, 87–96.

[35] Harald Søndergaard and Raoul A Mulder. 2012. Collaborative learning through formative peer review: Pedagogy, programs and potential. *Computer Science Education* 22, 4 (2012), 343–367.

[36] SPSG Hub. 2025. *Scaffolded Projects for the Social Good*. https://spsg-hub.github.io/

[37] Ahsun Tariq. 2023. Metacognitive Scaffolding to Leverage Decision Making in Software Engineering Education. In *Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2*. 127–129.

[38] Keith Topping. 1998. Peer assessment between students in colleges and universities. *Review of Educational Research* 68, 3 (1998), 249–276.

[39] Jakko Van der Pol, BAM Van den Berg, Wilfried F Admiraal, and P Robert-Jan Simons. 2008. The nature, reception, and use of online peer feedback in higher education. *Computers & Education* 51, 4 (2008), 1804–1817.