Honest Learning in a Distracted World: Formative Assessment for Introductory Programming

Corinna List
Augsburg Technical University
of Applied Sciences
Augsburg, Germany
Email: corinna.list@tha.de

Michael Kipp
Augsburg Technical University
of Applied Sciences
Augsburg, Germany
Email: kipp@tha.de

***Abstract*—Introductory programming courses face growing challenges in an era where students can disengage from class without immediate academic penalty and GenAI tools enable a dangerous sense of false competence. This experience report describes the iterative design of a formative assessment ecosystem for a CS1 course serving a heterogeneous cohort of design-oriented and technically motivated students in an Interactive Media program. We analyze which formative activities provide honest signals of exam readiness – and which do not. Our findings reveal a striking contrast: At-Home Checkups and In-Lecture Peer Instruction showed no meaningful relationship with final exam outcomes, while active task completion during In-Lab Coding Challenges correlated moderately to strongly (Spearman's $\rho = -0.58$), and Mock-Exam Peer Assessment proved a strong early predictor ($\rho = 0.73$). The pattern suggests that activities requiring genuine cognitive effort under authentic, AI-free conditions signal competence in ways that passive or easily substituted activities cannot. In addition, a misconception-oriented coaching intervention for students preparing for an exam retake yielded significant gains in confidence and self-efficacy, with the overall retake failure rate dropping by 74% compared to the average of the preceding three years. These findings offer practical guidance for designing formative assessment ecosystems that generate honest learning signals in a distracted, AI-enabled world.**

I. Introduction

Recent shifts in the higher education landscape have made teaching introductory programming courses harder in new ways. Post-pandemic patterns have led to a significant and persistent decline in in-person attendance, as students have adapted to flexible and remote learning modalities that became normalized during the pandemic [1]. And the rapid adoption of GenAI tools has introduced a subtler challenge: students who rely on AI assistance for exercises may develop a false sense of competence [2][3], believing they understand material they have never independently worked through. Together, these forces make it increasingly difficult to distinguish genuine learning from surface engagement – a challenge that is particularly acute in heterogeneous cohorts where students already differ widely in motivation, background, and learning goals [4].

The course described in this paper is part of an *Interactive Media* program and attracts both technically oriented students and design-oriented students whose primary focus lies in creative and visual practice. This heterogeneity is a defining characteristic of the program, and supporting this broad spectrum of learners while ensuring that all students develop genuine programming competence has been its central didactic challenge. The course has been developed and refined over more than fifteen years of teaching, shaped by honest reflection on what works and what does not, and building on the principle of constructive alignment [5].

To address the differing motivations, skillsets, and backgrounds of the students, we iteratively developed a formative assessment ecosystem combining interactive lectures, continuous low-stakes assessment, peer-supported reflection, and targeted interventions for students at risk of failure. The ecosystem includes At-Home Checkups (chapter-based online quizzes), In-Lab Coding Challenges (short programming exercises during lab sessions), In-Lecture Peer Instruction, and Mock-Exam Peer Assessment. Misconception-oriented coaching sessions were additionally introduced to support students preparing for exam retakes. The design of individual activities was informed by the ICAP framework [6], which distinguishes cognitive engagement modes – from Passive and Active to Constructive and Interactive – with deeper modes predicted to produce greater learning gains.

Our analysis across multiple semesters reveals a striking and counterintuitive pattern. At-Home Checkups and In-Lecture Peer Instruction participation – both widely used and seemingly engaging – showed no meaningful relationship with final exam outcomes. In contrast, active task completion during In-Lab Coding Challenges correlated moderately to strongly with exam performance (Spearman's $\rho = -0.58$)[1], and Mock-Exam Peer Assessment proved a strong early predictor of exam readiness ($\rho = 0.73$), while mere attendance did not. Activities requiring genuine cognitive effort under authentic, AI-free conditions allow clearer evaluation of competence than low-effort or easily outsourced activities – a distinction with growing practical relevance.

A misconception-oriented coaching intervention for students preparing for an exam retake further reinforced this picture: eight of nine participants passed with significant gains in

[1]Note that due to the German grading system, where 1 represents the highest and 5 the lowest grade (with 1 corresponding to an A and 5 to an F in the U.S. grading system), negative ρ values indicate a positive relationship between the variable and exam performance

topic-specific confidence and self-efficacy, and the overall retake failure rate dropped by 74% compared to the three-year average.

The main contributions of this paper are: (1) an experience report on a formative assessment ecosystem for heterogeneous CS1 cohorts, refined over fifteen years of practice; (2) empirical evidence on which formative activities provide honest signals of exam readiness and why others do not; and (3) results from a misconception-oriented coaching intervention for at-risk students.

II. Related Work

This paper engages with three established research areas: teaching heterogeneous CS1 cohorts, formative assessment design and its relationship to exam performance, and misconception-oriented coaching for at-risk students.

Teaching Heterogeneous Student Cohorts in CS1. Introductory programming courses have long faced the challenge of serving students with widely varying backgrounds, motivations, and learning goals. Forte and Guzdial [4] identified distinct audience segments within CS1, with non-majors bringing fundamentally different motivations than CS-major peers, and argued that effective instructional design must account for these differences explicitly. Research using Processing as a context for generative art and creative coding has similarly aimed to make programming accessible and engaging for diverse student cohorts [7]. Studies comparing CS and non-CS students in joint courses find that non-CS students tend to adopt more surface-level learning goals, focusing on task completion rather than deeper conceptual engagement [8]. Our course faces an analogous challenge: serving design-oriented and technically motivated students within the same instructional framework, using Processing's visual output as a motivation entry point while maintaining conceptual depth.

Formative Assessment, Peer Instruction, and Peer Assessment. Biggs' principle of constructive alignment [5] provides a theoretical foundation for integrating assessment coherently: learning objectives, teaching activities, and assessment formats should be deliberately aligned so that students practice precisely the competencies they will be assessed on. The ICAP framework [6] offers a finer-grained lens: it categorizes cognitive engagement as *Passive*, *Active*, *Constructive*, or *Interactive*, with deeper modes predicted to produce greater learning gains. Both frameworks inform our formative assessment design and the interpretation of our findings.

Peer Instruction (PI), introduced by Mazur [9], has been extensively studied in CS1. Porter et al. [10] reported that PI reduced failure rates from 23% to 8% across four CS1 courses, and Zingaro [11] showed that combining peer discussion with instructor intervention produced larger gains particularly for weaker students. However, these studies primarily measured learning gains through pre/post concept inventories rather than final exam performance – a distinction central to our analysis. Regarding peer assessment, while prior work has shown benefits for code review and accountability [12], its use as a signal of exam readiness has received little empirical attention and is one focus of this paper.

Fig. 1. A custom online textbook (in German) is the main learning resource. Each chapter opens with learning objectives and prerequisites; the book provides detailed explanations, exercises, embedded videos, and summaries across all chapter topics.

Programming Misconceptions and Targeted Coaching. A substantial body of research documents the persistent misconceptions that novice programmers develop [13], [14], [15]: prior knowledge and flawed mental models emerge as the main sources of difficulty, and such errors often resist correction unless explicitly targeted [16]. Despite Bloom's landmark findings on the effectiveness of one-on-one tutoring [17], such interventions remain rare in introductory programming education, likely due to the instructor time they require and the difficulty of scaling to large cohorts. Our coaching program addresses this gap by targeting students at the exam retake phase – when regular course structures are no longer available and students are largely left to self-study – and grounding each session in a structured misconception repository. Rather than repeating course content generically, sessions are explicitly designed to identify and resolve individual conceptual errors. This paper reports empirical outcomes of this misconception-oriented approach in a CS1 context.

III. Context and Design Principles of the Course

A. An Interdisciplinary Cohort: Designers and Programmers

The course described in this paper is part of the undergraduate curriculum of the *Interactive Media* program at Augsburg Technial University of Applied Sciences and introduces fundamental programming concepts to cohorts of typically 65–80 first-semester students. Interactive Media attracts students with widely different orientations: some are drawn primarily to (visual) design and creative practice, while others come with a strong interest in technology and computation. This heterogeneity is a defining characteristic of the program, and poses the central didactic challenge of the course.

A pre-course survey conducted in winter semester 2025/26 (n = 41) confirmed this asymmetry empirically: 53.6% of students identified primarily or fully as designers, while only 14.6% identified primarily with a technical or computational orientation. The remaining 31.7% placed themselves equally

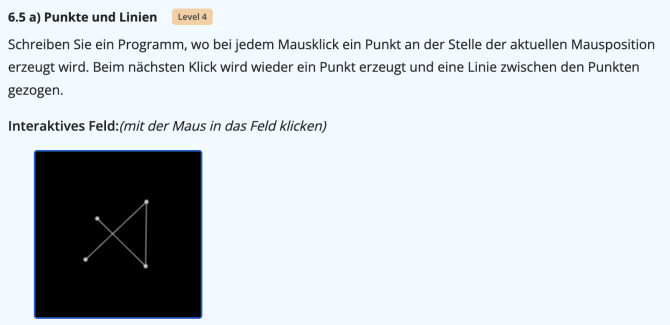


Fig. 2. A difficulty-rated exercise with a live preview widget – students observe the expected program behavior before writing their own solution.

between both perspectives. The course must therefore support students with little intrinsic interest in programming sufficiently to ensure successful completion, while simultaneously providing enough depth and challenge to engage those who are highly motivated to develop programming expertise.

The course and online textbook[2] (Fig. 1) have evolved iteratively over more than fifteen years of teaching by the same professor, shaped by student feedback and honest reflection on what works. Earlier iterations included a traditional Java course with slides, an objects-first approach using BlueJ and Greenfoot, and most recently an inverted classroom experiment – which was discontinued after attendance in voluntary in-class sessions dropped significantly. The current design reflects the lessons of these iterations.

B. Course Structure and Learning Environment

Each week follows a three-phase structure designed to create an effective feedback loop between instruction and practice. The first lecture, taught by the senior instructor, introduces new concepts through live coding demonstrations, writing and sketching on a digital board and active learning elements rather than traditional slide-based presentations. This is followed 1-2 days later by a lab session in small groups of around 20 students, supported by both instructors and tutors, where students apply what they have just learned through hands-on exercises. The second lecture, taught by the PhD teaching assistant who joined the course as a student tutor and has grown with it over several years, typically picks up on difficulties observed in the lab, addressing common errors and misconceptions in a timely manner. This weekly structure ensures that student difficulties surface quickly and are addressed before the next topic is introduced.

The course uses Processing[3], a Java-based programming language whose visual orientation enables immediate graphical feedback. Students can create compelling visual and interactive outputs from the very first sessions, providing an accessible entry point for design-oriented students while its Java foundation supports the introduction of rigorous programming concepts.

[2]https://www.michaelkipp.de/processing
[3]https://www.processing.org

A central resource supporting the course is a custom online textbook developed over many years (Fig. 1). Each chapter opens with explicit learning objectives and a list of prerequisite chapters, forming a navigable knowledge graph that supports self-paced learning. Each chapter includes embedded short videos (around 10 minutes each) explaining key concepts through worked coding examples, giving students an additional modality for understanding before attempting exercises. All exercises are labeled with a difficulty level (1–5), as in Fig. 2; some have collapsible hints, and many include live preview widgets where students can observe and try out the expected program behavior before writing their own solution.

The final exam takes place at a lab computer under supervised conditions and lasts 90 minutes. Students are expected to produce fully working programs rather than code fragments or pseudocode – a design choice that deliberately mirrors the kind of effortful, authentic practice emphasized throughout the course. Additional platforms include Moodle for asynchronous content and a campus app [18] for live quizzes and in-class interaction.

All course data analyzed in this paper was pseudonymized prior to analysis, and students provided informed consent for its use in research.

C. Didactic Design Principles

The course design follows constructive alignment [5]: learning objectives, teaching activities, and assessments are deliberately matched to the competencies students must demonstrate in the exam. At a finer grain, the ICAP framework [6] categorizes learning by cognitive engagement mode – *Passive* (receiving information), *Active* (manipulating material without transforming it), *Constructive* (generating novel output), and *Interactive* (two learners mutually building on each other's contributions) – with deeper modes predicted to produce greater learning gains.

This guided our formative assessment design: In-Lab Coding Challenges require students to produce and reason about code under time pressure (Constructive); the Mock-Exam Peer Assessment combines independent problem-solving with critical peer evaluation (Constructive/Interactive); low-stakes At-Home Checkups risk remaining at the Active or Passive level if completed without genuine effort. As discussed in Section VI, ICAP helps explain the differential relationship between formative activities and exam outcomes.

A further priority is making misconceptions visible and addressable throughout the course. Research documents the persistent conceptual errors novice programmers develop [13], [14], [15], which often resist correction unless explicitly targeted [16]. To this end, we maintain a structured misconception repository for Processing/Java [19] that informs exercises, Coding Challenge tasks, Peer Instruction questions, and the coaching intervention described in Section V.

IV. Formative Assessments: Design and Findings

The course integrates four complementary formative assessment activities that accompany students throughout the

semester, forming an ecosystem of low-stakes learning opportunities designed to support continuous feedback, encourage reflection, and make conceptual misunderstandings visible.

At-Home Checkups. Ten online chapter quizzes (one per chapter) were implemented in Moodle and made available for approximately two weeks each. They were deliberately designed as low-threshold activities to motivate continuous engagement with course material at home rather than to assess deep problem-solving skills. In ICAP terms, Checkups risk remaining at the *Active* or even *Passive* level when completed without genuine effort. To explore their relationship with exam performance, we analyzed data from two semesters ($n = 103$). No meaningful correlation between Checkup scores and final grades was observed (Spearman's $\rho = -0.26$, $p = .008$). Notably, 51.4% of students who failed the final exam had achieved above-average Checkup scores – consistent with their design intention: high scores reflect engagement, not necessarily exam readiness.

In-Lab Coding Challenges. During lab sessions, short exercises were conducted with some degree of time pressure. Each challenge consisted of three tasks to be completed in ten minutes: two solved on paper, requiring students to reason about code behavior (predicting output, sketching graphical results, or identifying errors), and one small programming task solved digitally. The time constraint was intentionally tight to remain challenging even for advanced students. Results and common difficulties were discussed at the opening of the following lecture. By requiring students to produce and reason about code under time pressure, Coding Challenges operate at the *Constructive* level of ICAP. Data from the first implementation (winter semester 2025/26, $n = 46$) revealed a moderate to strong correlation between the average number of tasks completed per attended session and the final exam grade (Spearman's $\rho = -0.58$, $p < .001$). Notably, attendance alone showed no relationship with exam outcomes ($\rho = -0.05$, $p = .724$), suggesting that active task engagement – not mere presence – is what matters.

In-Lecture Peer Instruction. During lectures, questions targeting typical misconceptions and common programming errors were posed. Students first voted anonymously via the campus app, discussed their reasoning with a neighbor, then voted again. The correct solution was subsequently discussed with the full class. While the peer discussion phase is designed to engage students at the *Interactive* level, participation metrics such as attendance or quiz count do not capture whether that interaction was genuine. Analysis of $n = 49$ students showed no significant relationship between PI participation and final grades (Spearman's $\rho = 0.16$, $p = .283$) or pass/fail status, a result that held when restricting the analysis to more regular attendees.

Mock-Exam Peer Assessment. Seven weeks before the final exam, students completed a set of exam-style tasks individually under exam-like conditions – without external resources or AI tools – then evaluated two anonymized peer submissions using a provided rubric and sample solution. The activity combines *Constructive* engagement (independent problem-solving under pressure) with *Interactive* engagement (critically evaluating a peer's reasoning), placing it at the deepest levels of ICAP. Peer Assessment was a deliberate choice over automated grading or grading by TA: it scales efficiently while preserving human interaction, and stepping into the assessor's role makes the exam's grading criteria transparent to students. The full process was implemented via Moodle. A strong positive correlation between peer assessment scores and final exam results was observed (Spearman's $\rho = 0.73$, $p < .001$, $n = 37$; Fig. 3), suggesting the activity captures meaningful aspects of exam readiness. Inter-rater agreement was moderate ($ICC = 0.55$, $n = 28$), with a slight tendency toward lenient ratings (mean peer score 68.5% vs. mean exam score 59.0%). Averaging the two peer ratings outperformed either single rating alone, suggesting that aggregating evaluations partially offsets individual leniency bias.

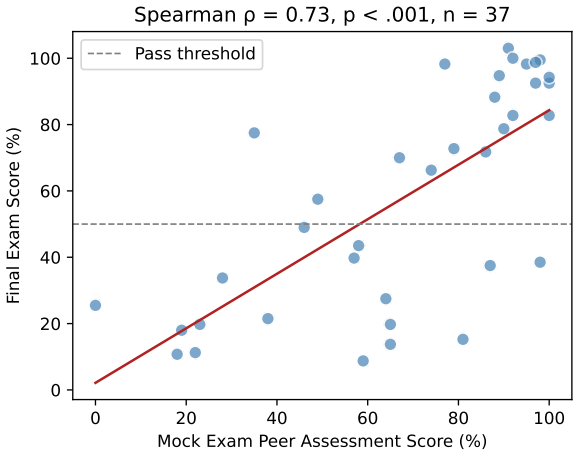


Fig. 3. Correlation between Mock-Exam Peer Assessment scores and final exam results. The dashed line indicates the passing threshold (50%).

V. Coaching At-Risk Students

In introductory programming courses, a subset of students accumulates conceptual misunderstandings over time that, if left unresolved, lead to repeated exam failure. At our institution, the course runs only in the winter semester, leaving students who fail to prepare for the summer retake largely on their own – without lectures, labs, or tutors. The coaching program was designed to provide structured support at this critical phase, grounded in the misconception repository described in Section III-C. The goal was not merely to practice programming tasks but to explicitly identify and resolve persistent misconceptions while helping students rebuild confidence and develop more effective preparation strategies.

Method. The program was offered on a voluntary basis to students who had previously failed the exam. Nine students participated; seven completed four sessions, one attended three, and one discontinued after the first, totaling 32 one-on-one online sessions of 60–100 minutes each. Before the first session, participants completed a survey on perceived topic

Fig. 4. Pre- and post-coaching comparison for self-report items (Wilcoxon signed-rank test; * $p < .05$, ** $p < .01$, n.s. = not significant). Items shown: "I have difficulty understanding the course content", "I feel confident in writing simple programs", "I did not prepare for the exam in the right way", "I enjoy programming". Lower scores indicate less agreement; higher scores indicate more agreement.

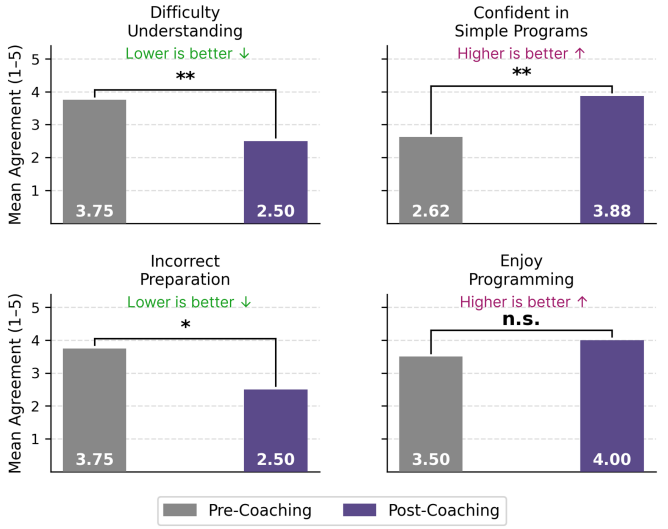
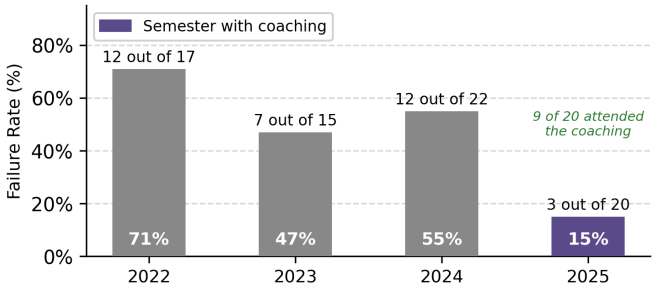


Fig. 5. Failure rates (failed/total participants) across the past four summer semesters (exam retakes)



difficulties, confidence levels, and prior preparation strategies; individual learning plans were created and refined throughout. Each session began with short oral concept-check questions, followed by programming tasks solved while thinking aloud, with the coach providing minimal but targeted guidance. Homework tasks drawn from the misconception repository were assigned at the end of each session and reviewed at the next. The most frequently addressed topics were arrays (11 sessions), classes (8), and functions (5).

Individual Outcomes. Students reported a significantly greater increase in confidence for coached topics (1.54 points on a 5-point scale) than for uncoached topics (0.63 points; Mann-Whitney $U = 746.5$, $p = .0001$). Three of five self-report items improved significantly after coaching (Fig. 4): perceived difficulty decreased by 1.25 points ($W = 28.0$, $p = .007$), agreement with having prepared incorrectly decreased by 1.25 points ($W = 19.5$, $p = .029$), and confidence in writing simple programs increased by 1.25 points ($W = 0.0$, $p = .008$). Eight of nine participants passed the retake, including three who achieved the highest possible grade. Among students who had failed in the previous semester, coached participants ($n = 7$) improved their scores by 51.9 points on average (SD = 19.4), compared to 45.0 points (SD = 19.1) among uncoached retake students ($n = 8$) – a practically meaningful difference, though not statistically significant at this sample size ($p = .11$).

Cohort-Level Impact. At the cohort level, the overall retake failure rate dropped from a three-year average of 57.7% to 15% (Fig. 5) – a 74% reduction and the lowest value in four years.

VI. Discussion

A. Formative Activities and Learning Signals

The four formative activities served complementary roles within the course design, with notable differences in their relationship to exam performance.

Mock-Exam Peer Assessment provided the strongest early indicator of exam readiness ($\rho = 0.73$). This is consistent with the principle of constructive alignment [5]: the activity closely mirrors the format, conditions, and cognitive demands of the final exam, which likely explains its predictive value. The exploratory analysis further revealed a systematic leniency bias in peer grading, which was partially offset by averaging two ratings – a finding that supports the use of multiple peer evaluators even in small-scale implementations.

Performance in **At-Home Checkups** showed no relationship with exam results, consistent with their design intention as low-threshold engagement activities. However, this creates a practical risk: students may interpret high Checkup scores as evidence of exam readiness. Explicitly communicating the limited predictive value of these activities therefore appears important to avoid overly optimistic self-assessment.

In-Lecture Peer Instruction showed no clear relationship between participation and exam performance. This replicates the pattern reported by Porter et al. [10] and Zingaro [11], whose studies found PI effects primarily on conceptual gains rather than summative performance. Our analysis only considered participation frequency; whether engagement quality or answer accuracy would show a stronger relationship remains an open question.

In-Lab Coding Challenges present a different picture. Active task completion correlated moderately to strongly with exam outcomes ($\rho = -0.58$), while attendance alone did not ($\rho = -0.05$), consistent with the broader literature on effortful retrieval practice [20]. It should be noted that task completion was measured as submission count rather than solution quality; whether correctness or depth of engagement would show a stronger relationship remains an open question analogous to the PI participation analysis.

Taken together, these findings are well explained by the ICAP framework [6]: activities that reliably engage students

at the *Constructive* or *Interactive* level – producing code under pressure, evaluating peer solutions – predict exam outcomes, while activities that can be completed at the *Active* or *Passive* level without genuine cognitive effort do not.

B. Impact and Scalability

The misconception-oriented coaching intervention showed promising results: significant gains in topic-specific confidence and general self-efficacy, and eight of nine participants passing the exam retake. The structured identification and resolution of misconceptions appears to help students address persistent conceptual difficulties that may otherwise remain unresolved during independent self-study.

These findings must be interpreted with caution. Only eight students completed both pre- and post-surveys, limiting statistical power. The voluntary nature of participation introduces a possible self-selection bias, as more motivated students may have been more likely to enroll. Furthermore, the improvement observed in the uncoached comparison group (45.0 points) suggests that students generally approach retakes better prepared regardless of coaching, possibly due to increased familiarity with the exam format or greater motivation. The additional gain observed in coached students (51.9 points) may reflect the specific benefit of misconception-targeted support, but this cannot be established conclusively from the available data. At the cohort level, the drop in failure rate from an average of 57.7% to 15% in summer 2025 – a 74% reduction – is consistent with this interpretation, but must be treated with caution for two additional reasons: the number of retake participants is considerably smaller than in the main cohort, amplifying the influence of individual cases; and while exam difficulty is kept as consistent as possible across semesters, variations cannot be fully ruled out.

While the coaching sessions proved valuable, their one-on-one format requires substantial instructor time and is difficult to scale. Two potential directions present themselves. First, trained student coaches could take over session delivery, modeled on the protocols developed here. Second, rather than a general-purpose chatbot, a purpose-built intelligent tutoring system we are currently developing [21] — which withholds model solutions and structures its feedback pedagogically — could supervise practice sessions grounded in our misconception repository, reducing instructor burden while preserving targeted conceptual support.

VII. Conclusion and Future Work

This paper describes the iterative design of a formative assessment ecosystem for an introductory programming course in the Interactive Media program, serving a heterogeneous CS1 cohort. Analyzing data from multiple semesters, we find that formative activities serve different and complementary roles, well characterized by the ICAP framework [6]: activities engaging students at the *Constructive* or *Interactive* level – In-Lab Coding Challenges and Mock-Exam Peer Assessment – showed meaningful correlations with final exam performance, while At-Home Checkups and In-Lecture Peer Instruction, which can be completed without deep cognitive engagement, did not. An exploratory analysis of peer grading further revealed a systematic leniency bias, partially offset by averaging two peer ratings.

The misconception-oriented coaching intervention for at-risk students showed promising results despite its small scale: significant gains in topic-specific confidence and self-efficacy, and eight of nine participants passing the exam retake. The overall retake failure rate dropped to 15% in the coached semester – a 74% reduction compared to the three-year average and the lowest value in four years.

These findings underscore the importance of distinguishing between formative activities that support learning processes and those that signal exam readiness. Future work will focus on scaling the coaching intervention – through student coaches trained on the protocol developed here as a near-term step, and a hybrid model incorporating LLM-based tutoring for practice sessions as a longer-term direction. We further plan to improve the diagnostic use of learning analytics within the formative assessment ecosystem.

References

[1] T. Lowe, C. Naughton, T. Struetzel, R. Adams, J. Walker, P. Kynaston, and C. Mackey, "Why Are Students Not Attending In-Person Classes Post-COVID-19? An Explorative Study in Student Engagement," *Active Learning in Higher Education*, p. 14697874251366213, Sep. 2025. [Online]. Available: https://doi.org/10.1177/14697874251366213

[2] J. Prather, B. N. Reeves, J. Leinonen, S. MacNeil, A. S. Randrianasolo, B. A. Becker, B. Kimmel, J. Wright, and B. Briggs, "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," in *Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1*, ser. ICER '24, vol. 1. New York, NY, USA: Association for Computing Machinery, Aug. 2024, pp. 469–486. [Online]. Available: https://dl.acm.org/doi/10.1145/3632620.3671116

[3] S. Rojas-Galeano, "New Kid in the Classroom: Exploring Student Perceptions of AI Coding Assistants," Sep. 2025, arXiv:2507.22900 [cs.HC]. [Online]. Available: http://arxiv.org/abs/2507.22900

[4] A. Forte and M. Guzdial, "Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses," *IEEE Transactions on Education*, vol. 48, no. 2, pp. 248–253, May 2005. [Online]. Available: https://ieeexplore.ieee.org/document/1427874

[5] J. Biggs, "Enhancing teaching through constructive alignment," *Higher Education*, vol. 32, no. 3, pp. 347–364, Oct. 1996. [Online]. Available: https://doi.org/10.1007/BF00138871

[6] M. T. H. Chi and R. Wylie, "The ICAP framework: Linking cognitive engagement to active learning outcomes," *Educational Psychologist*, vol. 49, no. 4, pp. 219–243, 2014.

[7] I. Greenberg, D. Kumar, and D. Xu, "Creative coding and visual portfolios for CS1," in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, ser. SIGCSE '12. New York, NY, USA: Association for Computing Machinery, Feb. 2012, pp. 247–252. [Online]. Available: https://dl.acm.org/doi/10.1145/2157136.2157214

[8] Z. Gao, H. Yan, J. Liu, X. Zhang, Y. Lin, Y. Zhang, X. Sun, and J. Feng, "Tracing distinct learning trajectories in introductory programming course: a sequence analysis of score, engagement, and code metrics for novice computer science vs. math cohorts," *International Journal of STEM Education*, vol. 12, no. 1, p. 27, May 2025. [Online]. Available: https://doi.org/10.1186/s40594-025-00546-2

[9] E. Mazur, *Peer Instruction: A User's Manual*, ser. Series in Educational Innovation. Prentice Hall, 1997, backup Publisher: Prentice Hall.

[10] L. Porter, C. Bailey Lee, and B. Simon, "Halving fail rates using peer instruction," in *Proceeding of the 44th ACM technical symposium on Computer science education*, ser. ACM Conferences, Mar. 2013, pp. 177–182. [Online]. Available: https://dl.acm.org/doi/10.1145/2445196.2445250

[11] D. Zingaro and L. Porter, "Peer Instruction in computing: The value of instructor intervention," *Computers & Education*, vol. 71, pp. 87–96, Feb. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0360131513002777

[12] A. K. Al-Khalifa and M. Devlin, "Evaluating a Peer Assessment Approach in Introductory Programming Courses," in *United Kingdom & Ireland Computing Education Research conference.*, ser. ACM Other conferences, Sep. 2020, pp. 51–58. [Online]. Available: https://dl.acm.org/doi/10.1145/3416465.3416467

[13] J. Sorva, "Visual Program Simulation in Introductory Programming Education," Ph.D. dissertation, Aalto University, Espoo, Finland, 2012.

[14] L. Chiodini, I. Moreno Santos, A. Gallidabino, A. Tafliovich, A. L. Santos, and M. Hauswirth, "A Curated Inventory of Programming Language Misconceptions," in *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ser. ITiCSE '21. New York, NY, USA: Association for Computing Machinery, Jun. 2021, pp. 380–386. [Online]. Available: https://dl.acm.org/doi/10.1145/3430665.3456343

[15] R. Caceffo, P. Frank-Bolton, R. Souza, and R. Azevedo, "Identifying and Validating Java Misconceptions Toward a CS1 Concept Inventory," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 23–29. [Online]. Available: https://dl.acm.org/doi/10.1145/3304221.3319771

[16] Y. Qian and J. Lehman, "Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review," *ACM Transactions on Computing Education*, vol. 18, no. 1, pp. 1–24, Mar. 2018. [Online]. Available: https://dl.acm.org/doi/10.1145/3077618

[17] B. S. BLOOM, "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring," *Educational Researcher*, vol. 13, no. 6, pp. 4–16, Jun. 1984. [Online]. Available: https://doi.org/10.3102/0013189X013006004

[18] C. List, L. Wagner, A. Prochaska, M. Müller, and M. Kipp, "Digitale Dynamik im Hörsaal: Wie die Campus App synchrone Lehre unterstützt," 2025. [Online]. Available: https://opus4.kobv.de/opus4-haw/6437

[19] C. List, M. Müller, and M. Kipp, "Yet Another Collection of Programming Misconceptions – To Help Educators Find What Matters," in *Proceedings of the 25th Koli Calling International Conference on Computing Education Research*, ser. ACM Other conferences, Nov. 2025, pp. 1–3. [Online]. Available: https://dl.acm.org/doi/10.1145/3769994.3770058

[20] H. L. Roediger III and J. D. Karpicke, "Test-Enhanced Learning: Taking Memory Tests Improves Long-Term Retention," *Psychological Science*, vol. 17, no. 3, pp. 249–255, Mar. 2006. [Online]. Available: https://doi.org/10.1111/j.1467-9280.2006.01693.x

[21] M. Müller, C. List, and M. Kipp, "The Power of Context: An LLM-based Programming Tutor with Focused and Proactive Feedback," in *Proceedings of the 6th European Conference on Software Engineering Education*. Seeon Germany: ACM, Jun. 2025, pp. 1–10. [Online]. Available: https://dl.acm.org/doi/10.1145/3723010.3723034