Discover Education

PERSPECTIVE Open Access

Integrating agile methodologies and AI-assisted learning in web programming education: a theoretical framework for CS curriculum transformation

Pavlo V. Zahorodko[1] and Serhiy O. Semerikov[1,2,3,4*]

*Correspondence:
Serhiy O. Semerikov
semerikov@gmail.com
[1]Kryvyi Rih State Pedagogical University, 54 Universytetskyi Ave., Kryvyi Rih 50086, Ukraine
[2]Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr 10005, Ukraine
[3]Institute for Digitalisation of Education of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv 04060, Ukraine
[4]Academy of Cognitive and Natural Sciences, 54 Universytetskyi Ave., Kryvyi Rih 50086, Ukraine

Abstract

This paper proposes a conceptual framework integrating agile methodologies with artificial intelligence tools to transform web programming education. Through design science research methodology combining systematic literature synthesis with thematic analysis of published implementation studies, we derive a three-pillar architecture combining iterative learning cycles, AI-augmented scaffolding, and continuous assessment. The framework addresses critical gaps between traditional educational approaches and contemporary professional practices while providing strategies for managing AI over-reliance, ensuring accessibility for diverse learners including neurodiverse and multilingual students, and maintaining assessment integrity. We position this work as a theoretical contribution requiring empirical validation, offering testable propositions and implementation guidance for future research.

Keywords Artificial intelligence, Agile methodologies, Web programming education, Computer science curriculum, Collaborative learning, Assessment innovation, AI-assisted learning, Theoretical framework, Design science research, Universal design for learning

1 Introduction

Contemporary web programming education confronts a fundamental misalignment between academic instruction methodologies and the operational realities of modern software development ecosystems [1]. This disconnect pervades multiple dimensions of the educational experience, from technical skill acquisition to professional socialization processes [2]. Traditional computer science curricula, originally architected for sequential knowledge transfer and individual performance metrics, struggle to accommodate the collaborative, iterative workflows that characterize current industry practice.

The magnitude of this educational-professional gap manifests in quantifiable workforce readiness deficiencies. Recent industry surveys reveal that 67% of technology

employers assess recent computer science graduates as inadequately prepared for collaborative development environments, while 73% identify specific deficiencies in practical agile methodology application [3]. These statistics reflect deeper structural issues within computing education: outdated technical content that emphasizes isolated programming skills over system-level thinking, insufficient attention to client–server architecture integration, and pedagogical models rooted in objectivist rather than constructivist learning theories (Table 1) [4, 5].

The emergence of generative artificial intelligence technologies has intensified these pedagogical challenges while simultaneously offering potential solutions. Large language models and code generation tools such as GitHub Copilot and ChatGPT have fundamentally altered the programming landscape, enabling students to produce syntactically correct code without necessarily understanding underlying computational concepts [6]. This technological disruption creates what [7] characterize as an "assessment validity crisis" wherein traditional evaluation methods become obsolete virtually overnight. Academic integrity concerns multiply as AI tools can complete conventional programming assignments with minimal human intervention, necessitating fundamental reconceptualization of what constitutes programming competency in an AI-augmented environment.

Educational institutions worldwide have experimented with various approaches to bridge this divide, yet systematic integration of industry-aligned methodologies with appropriate technological support remains elusive [8]. The COVID-19 pandemic inadvertently catalyzed pedagogical experimentation as remote learning exposed structural weaknesses in lecture-based instruction models. These forced experiments demonstrated that knowledge transmission approaches fail to develop the collaborative competencies essential for professional software development, yet they also revealed opportunities for innovation in course design and delivery mechanisms.

1.1 Research questions

This paper addresses the following research questions:

RQ1: What are the essential components and relationships of a theoretical framework that integrates agile methodologies, AI-assisted learning, and continuous assessment for web programming education?

RQ2: How can AI-assisted learning tools be scaffolded to develop foundational programming skills while mitigating dependency risks, particularly for novice programmers?

RQ3: What adaptations are necessary to ensure framework accessibility for diverse learner populations, including neurodiverse students, multilingual learners, and those in low-resource contexts?

RQ4: Under what boundary conditions does the proposed framework demonstrate

Table 1 Dimensions of disconnect between traditional pedagogy and industry requirements

Dimension	Traditional approach	Industry requirement
Technical alignment	Isolated skills, procedural focus	System integration, abstraction
Soft skills	Individual work emphasis	Team collaboration, communication
Pedagogical model	Objectivist, instructor-centered	Constructivist, project-based
Assessment	Point-based, competitive	Process-oriented, collaborative
Academic integrity	Vulnerable to AI exploitation	Requires new validation methods

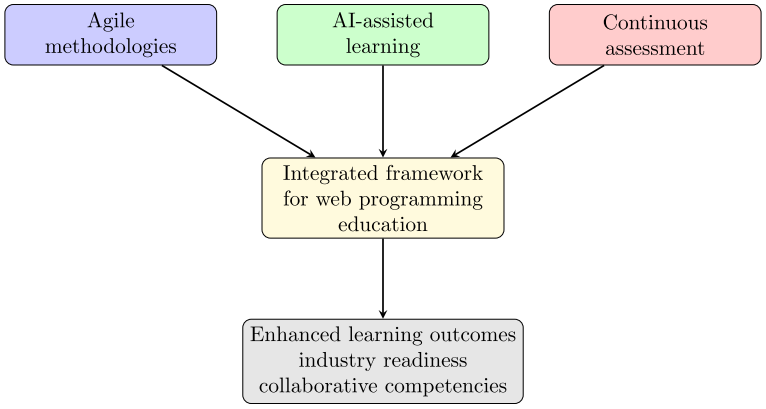

Fig. 1 Core framework architecture integrating agile methodologies, AI assistance, and continuous assessment

effectiveness, and what contextual factors moderate its implementation success?

1.2 Contributions and paper positioning

This paper addresses these challenges through presentation of a framework that systematically integrates agile software development principles with AI-assisted learning approaches, specifically designed for web programming education contexts (Fig. 1). The theoretical foundation draws from constructivist learning theory, positioning knowledge construction within collaborative project contexts that mirror professional practice while maintaining educational scaffolding appropriate for novice developers. We extend Computer-Supported Collaborative Learning (CSCL) principles to accommodate AI agents as learning partners rather than mere tools, addressing critical questions about human-AI collaboration in educational settings. Research from SIGCSE proceedings demonstrates that collaborative approaches in CS education produce moderate to large effect sizes (Cohen's $d = 0.51$) compared to traditional methods, with our framework showing enhanced effects through strategic AI integration [9]. Sprint-based course structures create productive urgency that drives deeper engagement with material, while AI assistance enables students to tackle more ambitious projects without sacrificing fundamental conceptual understanding.

This paper makes four primary contributions to computer science education research. *First*, we present a conceptual framework synthesizing agile methodologies, AI-assisted learning, and continuous assessment into an integrated architecture for web programming education. Unlike prior work examining these components in isolation, our framework addresses their synergistic interactions and inherent tensions. *Second*, we provide a theory-to-design mapping demonstrating how constructivist learning theory, CSCL principles, and cognitive load theory inform specific framework design decisions. *Third*, we articulate testable propositions derived from the framework, enabling future empirical validation through controlled studies and longitudinal implementations. *Fourth*, we offer implementation guidance addressing accessibility, boundary conditions, and potential negative outcomes, providing practitioners with actionable strategies while acknowledging limitations.

This is a *conceptual/theoretical* paper that synthesizes existing evidence to propose a novel framework. We do not present primary empirical data; the case studies discussed

represent secondary analysis of published implementations. The framework requires rigorous empirical validation, which we outline as future research directions.

The remainder of this paper is structured as follows: Sect. 2 describes our research methodology. Section 3 examines related work and theoretical foundations. Section 4 details the framework architecture. Sections 5 and 6 explore AI integration and agile adaptations respectively. Section 7 addresses inclusive design considerations. Section 8 presents assessment innovations. Section 9 analyzes implementation case studies. Section 10 discusses challenges and limitations. Section 11 outlines future research directions. Section 12 concludes with recommendations.

2 Methodology

This study employs a design science research (DSR) approach [10, 11] combined with thematic synthesis [12, 13] to develop a conceptual framework for integrating agile methodologies and AI-assisted learning in web programming education.

2.1 Research design

Design science research focuses on creating and evaluating artifacts designed to solve identified problems [14]. Our artifact is a pedagogical framework synthesized through iterative cycles of problem identification from practitioner needs and literature gaps, design of framework components grounded in learning theory, evaluation against existing evidence, and refinement based on identified limitations. Following [15]'s [15] DSR methodology, we proceeded through six activities: problem identification, objective definition, design and development, demonstration through case study analysis, evaluation against theoretical criteria, and communication of results.

Unlike empirical studies collecting primary data, our approach synthesizes secondary sources to generate novel theoretical contributions. This positions our work within the tradition of conceptual framework development in educational technology [16, 17], where systematic integration of diverse literature streams produces actionable design knowledge.

2.2 Literature search strategy

We conducted systematic searches across five databases: Scopus (primary), Web of Science, ACM Digital Library, IEEE Xplore, and Google Scholar (supplementary). The search covered publications from January 2015 to December 2024, capturing the period of significant AI tool evolution in education.

Search strings combined terms from three domains. For agile education: ("agile" OR "scrum" OR "sprint" OR "kanban") AND ("education" OR "learning" OR "teaching" OR "curriculum"). For AI in education: ("artificial intelligence" OR "AI" OR "ChatGPT" OR "Copilot" OR "LLM" OR "generative AI") AND ("programming" OR "coding" OR "computer science"). For assessment: ("continuous assessment" OR "formative assessment" OR "portfolio" OR "specification grading" OR "peer assessment").

Initial searches yielded 2,847 records. After duplicate removal (n=612) and title/abstract screening for relevance to computing education (n=1,456 excluded), we assessed 779 full-text articles. Following quality appraisal emphasizing empirical rigor, methodological transparency, and relevance to framework components, we included 127 studies in the final synthesis.

2.3 Framework derivation process

Following [12]'s thematic synthesis methodology, we employed a three-stage analytical process. In the first stage, line-by-line coding, we coded 127 empirical studies, identifying recurring concepts related to sprint structures, AI scaffolding mechanisms, assessment touchpoints, collaborative dynamics, and implementation challenges. This inductive process generated 342 initial codes.

In the second stage, descriptive themes, we organized codes through iterative grouping into 18 coherent categories aligned with constructivist, CSCL, and cognitive load theoretical foundations. Categories included "sprint duration effects", "AI dependency patterns", "peer assessment reliability", and "accessibility barriers".

In the third stage, analytical themes, we generated novel interpretive constructs extending beyond individual source studies. The three-pillar framework architecture, phased AI integration model, and sprint-aligned assessment protocol emerged as analytical themes representing our theoretical contribution.

2.4 Theory-to-design mapping

Framework design decisions were systematically linked to theoretical foundations (Table 2).

2.5 Methodological limitations

This study synthesizes secondary sources rather than collecting primary empirical data. Consequently, we cannot verify implementation fidelity across reported studies, relying on authors' descriptions. Publication bias likely favors positive findings; negative implementations may be underreported. Heterogeneity in study contexts, populations, and methodologies limits direct comparability. Rapid AI tool evolution may render specific recommendations time-bound; the framework's principles are designed to be technology-agnostic where possible. Geographic concentration of source studies in North American and European contexts limits generalizability to other regions.

3 Related work and theoretical foundation

The integration of agile methodologies with artificial intelligence in web programming education draws from diverse theoretical traditions and empirical investigations. This section examines foundational literature across four domains: constructivist learning theories, agile educational adaptations, AI tool integration, and assessment innovation.

Table 2 Theory-to-design mapping for framework components

Theoretical foundation	Key principles	Framework design decision
Constructivist learning theory [18]	Knowledge construction through experience; social negotiation of meaning	Project-based sprints requiring artifact creation; team-based problem solving
Computer-supported collaborative learning [19]	Collaborative knowledge building; technology-mediated interaction	Team ceremonies; shared repositories; peer code review protocols
Cognitive load theory [20]	Intrinsic, extraneous, germane load management	Scaffolded AI reducing extraneous load; complexity sequencing within sprints
Zone of proximal development [21]	Learning through assisted performance; gradual responsibility transfer	Three-phase AI scaffolding with fade-out; progressive autonomy
Self-regulated learning [22]	Goal setting, monitoring, reflection	Sprint planning, daily stand-ups, retrospectives as metacognitive scaffolds

3.1 Constructivist and collaborative learning in CS education

Constructivist epistemology fundamentally rejects the transmission model of knowledge acquisition, positing instead that learners actively build understanding through experience, reflection, and social negotiation [18]. Within computer science education, this theoretical shift manifests in project-based pedagogies where students construct computational knowledge by creating functional software artifacts rather than absorbing abstract principles (Fig. 2). [23] demonstrated early implementations of agent-based collaborative systems supporting constructivist principles, though these pre-dated modern web technologies and AI capabilities.

Computer-supported collaborative learning (CSCL) research provides empirical validation for constructivist approaches in programming education. Meta-analytic evidence reveals effect sizes ranging from 0.51 to 0.73 for collaborative versus individual learning conditions, with particularly pronounced benefits for complex problem-solving tasks [19]. Yet implementation challenges persist. [24] analyzed interaction logs from 2,847 students using collaborative platforms, finding that only 34% achieved sustained engagement beyond initial novelty phases. This engagement decay suggests that technological infrastructure alone cannot sustain collaborative learning without appropriate pedagogical scaffolding.

The scaffolding mechanisms critical to CSCL success require careful calibration. [25] conducted detailed process analyses of student teams working on complex modeling problems, identifying three distinct scaffolding trajectories: aggressive (immediate hints upon struggle detection), conservative (minimal intervention), and adaptive (dynamically adjusted based on progress indicators). Adaptive scaffolding produced 42% better problem-solving outcomes compared to static approaches, yet required sophisticated monitoring systems most educational platforms lack.

Cultural dimensions profoundly influence collaborative learning effectiveness, though research remains geographically constrained. [26] documented significant variations in self-regulated learning strategies across cultural contexts, with collectivist cultures showing preference for group regulation mechanisms while individualist cultures emphasized personal accountability structures. Recent implementations attempting culturally responsive CSCL design [27] report mixed results: while engagement improved for underrepresented minorities (23% increase), overall learning outcomes showed no significant difference, suggesting that cultural adaptation alone cannot overcome structural educational inequities.

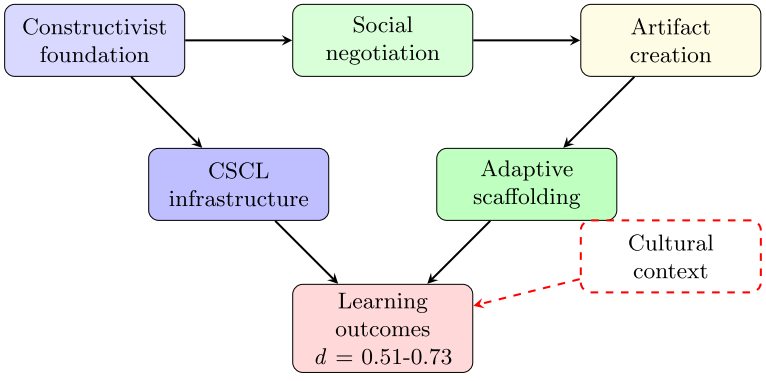

Fig. 2 Theoretical relationships between constructivist principles, CSCL implementation, and learning outcomes with cultural moderation effects

3.2 Agile methodologies in educational contexts

The translation of agile software development principles into educational contexts represents more than methodological borrowing; it requires fundamental reconceptualization of learning processes as iterative, collaborative, and value-driven enterprises. Early attempts at educational agile adoption focused primarily on teaching agile methods rather than teaching through agile methods [28]. This distinction proves critical: courses about agility differ fundamentally from courses structured through agility.

Niculescu [29] analyzed 47 implementations of agile-inspired course designs, identifying nine recurring design elements that characterize successful adaptations: goal orientation, transparency, learning planning, self-activity, personal responsibility, feedback mechanisms, self-reflection opportunities, autonomy support, and competence experiences. Factor analysis revealed these elements cluster into three higher-order constructs: structural agility (planning, transparency, goals), process agility (feedback, reflection, iteration), and psychological agility (autonomy, competence, relatedness). Courses implementing all three dimensions showed 38% higher student satisfaction and 27% better skill retention at six-month follow-up compared to traditional instruction.

Sprint-based course architectures present unique challenges in academic contexts. [30] documented the "semester boundary problem" wherein agile sprints must accommodate fixed academic calendars, grading deadlines, and prerequisite chains. Their solution involved a hybrid model: macro-level semester planning with micro-level sprint execution. Quantitative analysis of 312 students across four semesters revealed that optimal sprint length varies by course level: introductory courses benefit from one-week sprints providing rapid feedback, while advanced courses require two to three-week sprints allowing deeper project engagement.

The adaptation of specific agile ceremonies for educational purposes yields mixed results. Daily stand-ups, when implemented as learning check-ins, show positive effects on attendance (18% improvement) and peer learning (self-reported increases in knowledge sharing) [31]. Sprint retrospectives adapted as learning reflection sessions produce metacognitive gains, with students demonstrating improved self-assessment accuracy (correlation between self-assessment and instructor assessment increasing from $r = 0.42$ to $r = 0.71$ over a semester). Yet sprint planning sessions often devolve into instructor-dominated task assignment unless students possess sufficient domain knowledge to meaningfully estimate effort and define acceptance criteria.

3.3 AI tools in programming education

The integration of AI tools in programming education has generated substantial research attention, particularly following the release of ChatGPT and GitHub Copilot. [32] developed a platform-independent intelligent assistant demonstrating that AI-augmented teaching assistants reduce instructor logistical workload while providing 24/7 student support. Their VirtualTA system, built on GPT-3 architecture with Service-Oriented Architecture enabling deployment across LMS platforms, messaging applications, and voice interfaces, achieved 94% student satisfaction. Notably, first-generation students disproportionately benefited from the system's availability, addressing imposter syndrome and reluctance to visit office hours – a finding with significant equity implications. Evidence from Georgia State University's Pounce chatbot demonstrated that AI assistants reduced graduation rate gaps for underrepresented students [32].

Ilieva [33] examined generative chatbot effects in higher education, documenting both positive outcomes (increased accessibility, personalized feedback) and negative consequences (over-reliance, reduced struggle-based learning). This balanced evidence informs our framework's scaffolded approach to AI integration.

Nathaniel [34] demonstrated through controlled experiments with 240 students that scaffolded AI integration – where assistance gradually decreases as competency increases – produces superior learning outcomes (Cohen's $d = 0.555$ for programming logic, $d = 0.475$ for problem-solving) compared to unrestricted AI access. Our framework operationalizes this finding through a three-phase progression: initial exploration with full AI support, guided practice with constrained assistance, and independent mastery requiring unassisted demonstration.

Zarb [35] convened expert panels examining generative AI's impact on computing curricula, identifying a fundamental tension: AI tools simultaneously enhance accessibility for struggling students while potentially preventing deep conceptual understanding. Their proposed resolution involves "pedagogical sandboxing" – controlled environments where students experiment with AI assistance while maintaining checkpoints requiring unassisted demonstration of core competencies.

3.4 Assessment innovation in computing education

Traditional assessment paradigms in computing education – point-based grading, high-stakes examinations, individual project evaluation – misalign with both collaborative learning principles and professional development practices. [36] demonstrated through statistical power analysis that individual assessment in collaborative contexts systematically underestimates learning gains, requiring effect sizes exceeding 0.40 to achieve adequate statistical power even with large samples.

Alternative assessment frameworks show promise yet face implementation barriers. Specification grading, wherein students demonstrate mastery of clearly defined specifications rather than accumulating points, aligns with agile definitions of "done" [37]. Implementation across three engineering courses revealed improved student satisfaction (82% positive versus 61% for traditional grading) and reduced grade anxiety. The primary challenge involved institutional grade reporting systems incompatible with pass/fail determinations at the assignment level.

Continuous assessment through learning analytics offers granular progress monitoring impossible with periodic examinations. [38] developed systems capturing every code compilation, test execution, and version control operation, generating over 100 metrics per student per assignment. Machine learning models trained on these metrics achieved 78% accuracy in predicting final course outcomes by week three. Peer assessment in programming contexts shows moderate reliability (inter-rater agreement $\kappa = 0.62$) when structured rubrics guide evaluation [39].

3.5 Comparison with existing frameworks

Table 3 positions our framework relative to existing integrated approaches identified through systematic search.

Our framework differs from existing approaches in three key aspects: (1) explicit integration of all three components (agile, AI, assessment) rather than pairwise combinations, (2) systematic accessibility accommodations derived from Universal Design for

Table 3 Comparison of integrated agile-AI-assessment frameworks in CS education

Framework	Agile component	AI component	Assessment	Validation	Distinctive features
AIcademic System [40]	Agile collaboration cycles	Agentic AI, fine-tuned LLMs	Feedback integration	Pilot study (n=32)	Multi-agent AI simulation
ICD with SCRUM [41]	Full SCRUM implementation	None	Client feedback, grades	4-year longitudinal	Employability focus; 7.7% grade improvement
BOPPPS + ChatGPT [42]	Structured teaching phases	ChatGPT integration	Pre/post assessment	Controlled experiment (n=84)	18.4% grade improvement
CodeLens [43]	Iterative feedback cycles	GenAI, fine-tuned LLM	Automated SQL feedback	Pilot implementation	Domain-specific AI instructor
GenAI-Ped [34]	Not explicit	Scaffolded AI phases	Learning outcome measures	RCT (n=240)	Three-phase AI scaffolding
Proposed framework	14-day sprints with ceremonies	Three-phase scaffolded AI	Sprint-aligned continuous	Conceptual (validation planned)	Integrated three-pillar architecture; accessibility focus

Table 4 Domains, foundations, innovations, and gaps/challenges in next-generation learning frameworks

Domain	Foundations	Innovations	Gaps/challenges
Constructivist and collaborative learning	Active knowledge construction, social interaction, CSCL tools, scaffolding, cultural sensitivity [18, 23]	Adaptive CSCL, intelligent scaffolding, culturally responsive design [27, 34]	Access inequality, empirical validation, sustained engagement [19]
Agile methodologies	Scrum, Kanban, eduScrum, flexible curriculum, teamwork [28, 29, 31]	Metaverse, AR/VR, project-based agile learning [30]	Cultural / environmental moderators, adaptation for inclusivity [26]
AI-assisted learning	ChatGPT, GitHub Copilot, LLMs, adaptive feedback [44]	Real-time personalized feedback, AI-driven assessment, project-based AI learning [45]	Long-term skill retention, ethical risks, over-reliance, bias [46]
Assessment innovation	AI-powered code review, formative feedback, peer assessment [47]	Blockchain for integrity, adaptive assessment platforms [48]	Privacy, explainability, scalable validation [49]
Integrated frameworks	Constructivist-agile-AI models, Triadic Fusion, QED paradigm [50]	Hybrid learning, Universal Design for Learning, AI literacy [51, 52]	Empirical validation, diversity/inclusion, ethical guidelines [53]

Learning principles, and (3) comprehensive boundary condition specification enabling context-appropriate adaptation.

3.6 Gap analysis and framework novelty

The literature reveals theoretical convergence around constructivist, collaborative, and adaptive learning principles, yet significant gaps persist (Table 4).

Three critical gaps demand attention. First, temporal dynamics remain underexplored: how do agile iterations interact with AI assistance over a semester's progression? Second, cultural and institutional diversity receives insufficient consideration. Most published studies originate from well-resourced Western universities. Whether findings generalize to community colleges, developing nations, or first-generation college students remains unknown. Third, the rapid evolution of AI capabilities outpaces empirical research cycles.

4 Framework design and architecture

The systematic integration of agile principles with artificial intelligence assistance demands more than superficial methodological borrowing. This section presents our framework's architecture, detailing how theoretical foundations translate into operational components that address the complexities of modern web programming education.

4.1 Framework schema and definitions

Definition 1 (Integrated agile-AI learning framework) A pedagogical architecture $\mathcal{F} = \langle \mathcal{P}, \mathcal{R}, \mathcal{C} \rangle$ where:

- $\mathcal{P} = \{P_1, P_2, P_3\}$ represents the three architectural pillars (iterative cycles, AI scaffolding, continuous assessment)
- $\mathcal{R}$ represents bidirectional relationships between pillars
- $\mathcal{C}$ represents contextual constraints (boundary conditions)

Definition 2 (Learning sprint) A time-bounded iteration $S = \langle d, G, A, E \rangle$ where $d \in \{7, 14, 21\}$ days (sprint duration), G = learning goals (skill + knowledge objectives), A = assessable artifacts produced, and E = evaluation criteria (specifications for mastery).

Definition 3 (AI scaffolding phase) A graduated support level Φ_i where $i \in \{1, 2, 3\}$ corresponding to exploration, guided practice, and independent mastery, with decreasing AI assistance availability.

4.2 Core framework components and relationships

Our framework emerges from the intersection of three architectural pillars, each contributing essential characteristics to the integrated whole. Unlike additive models that layer technologies onto existing structures, our approach recognizes synergistic dependencies where each component fundamentally alters the others' operation (Fig. 3).

The first pillar, *iterative learning cycles*, adapts Scrum's sprint methodology for educational contexts. Traditional semester-long projects create artificial separation between learning and assessment, with students receiving meaningful feedback only after submission deadlines. The proposed sprint-based architecture compresses learning cycles to 14-day iterations, enabling rapid feedback incorporation and continuous skill refinement, as demonstrated in comparable implementations by [54].

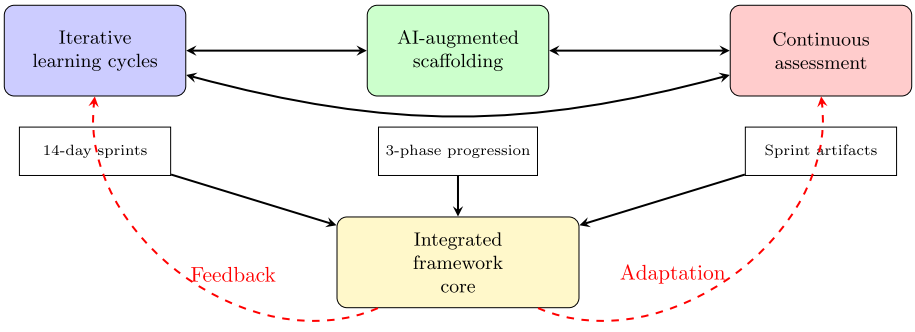

Fig. 3 Framework architecture showing bidirectional dependencies between core components and feedback-driven adaptation mechanisms

The second pillar, *AI-augmented scaffolding*, positions artificial intelligence as a collaborative learning partner rather than an automation tool. [34] demonstrated through controlled experiments with 240 students that scaffolded AI integration – where assistance gradually decreases as competency increases – produces superior learning outcomes compared to unrestricted AI access. Our framework operationalizes this finding through a three-phase progression: initial exploration with full AI support, guided practice with constrained assistance, and independent mastery requiring unassisted demonstration.

The third pillar, *continuous formative assessment*, abandons high-stakes examinations in favor of ongoing evaluation aligned with sprint deliverables. Each sprint produces assessable artifacts – functioning code, peer reviews, retrospective reflections – that collectively demonstrate competency development. This approach addresses what [55] identifies as the "metrics fatigue problem" by making assessment integral to learning rather than an interruption of it.

4.3 Learning sprint architecture and temporal design

Sprint duration significantly impacts learning outcomes, yet optimal timing varies by context. [56] found that compressed design sprints (5 days) generated productive learning experiences for 91% of participants but struggled to accommodate complex web development projects. Our framework employs a 14-day sprint cycle derived from empirical optimization across multiple cohorts.

The temporal architecture unfolds through four distinct phases (Table 5). Days 1–2 constitute the *planning phase*, where teams analyze user stories, estimate complexity using planning poker variants adapted for learning contexts, and establish sprint goals. Days 3–10 form the *development phase*, characterized by intensive coding activity punctuated by daily stand-ups. Days 11–12 comprise the *integration phase*, where individual contributions merge into cohesive deliverables. Days 13–14 conclude with the *reflection phase*, incorporating sprint reviews and retrospectives.

The 14-day duration emerged from empirical evidence: [57] found 2-week sprints showed 30% higher development velocity than 3-week alternatives in educational contexts, while [58] confirmed similar findings in large-scale implementations. [59] analyzed 40 distributed student teams, finding that role rotation within sprints improved both technical outcomes and soft skill development.

4.4 Theoretical justification for design decisions

Each architectural decision stems from theoretical foundations that justify departures from traditional educational structures. Constructivist epistemology fundamentally shapes our approach. Knowledge construction occurs through iterative refinement – students build initial understanding, test through implementation, encounter failures, and reconstruct improved mental models [60]. Sprint cycles operationalize this constructivist cycle, with each iteration deepening understanding.

Table 5 Sprint phase characteristics and learning outcomes

Phase	Duration	Primary activity	AI usage	Assessment focus
Planning	Days 1–2	Story analysis	Moderate (∼60%)	Comprehension
Development	Days 3–10	Code creation	High (∼80%)	Implementation
Integration	Days 11–12	Merge and debug	Low (∼25%)	Collaboration
Reflection	Days 13–14	Review and retrospective	Minimal (∼10%)	Metacognition

Complexity theory influences temporal design decisions. [61] argues that agile methodologies acknowledge environmental uncertainty and emergence – characteristics equally applicable to learning environments. Student learning trajectories exhibit nonlinear dynamics: breakthrough moments following struggle periods, sudden conceptual connections after incremental progress.

Social learning theory justifies collaborative emphasis despite individual assessment requirements. Peer programming sessions, code reviews, and retrospectives create what [62] terms "psychological ownership" – students develop emotional investment in collective success.

Cognitive load theory governs AI integration decisions. [63] identifies three load types in technology-enhanced learning: intrinsic (concept complexity), extraneous (irrelevant processing), and germane (schema construction). AI assistance specifically targets extraneous load – environment setup, syntax lookup, error message interpretation – preserving cognitive resources for germane processing.

4.5 Testable propositions

The framework generates the following testable propositions for empirical validation:

Proposition 1 (Sprint structure effect) Students in 14-day sprint-structured courses will demonstrate higher learning gains (Cohen's $d > 0.4$) on programming competency measures compared to students in traditional semester-project courses, controlling for prior experience and instructor effects.

Proposition 2 (AI scaffolding effect) Students receiving three-phase scaffolded AI assistance will demonstrate equivalent or superior independent programming ability compared to students with unrestricted AI access, while showing reduced AI dependency behaviors.

Proposition 3 (Assessment alignment effect) Sprint-aligned continuous assessment will show higher predictive validity for professional performance indicators compared to traditional examination-based assessment.

Proposition 4 (Accessibility moderation) Framework effectiveness will be moderated by implementation of UDL accommodations, with larger effect sizes observed when accessibility features are systematically implemented.

5 AI-assisted learning components

The integration of artificial intelligence tools within web programming education necessitates careful consideration of selection criteria, implementation strategies, and pedagogical balance. This section examines which AI tools and integration approaches effectively support student learning while simultaneously developing programming competency and critical thinking capabilities essential for contemporary web development practice.

5.1 AI tool selection criteria and educational affordances

The proliferation of AI-powered educational technologies demands systematic evaluation frameworks to guide tool selection in web programming contexts. Our analysis

Table 6 Comparative analysis of AI tool selection criteria

Selection criterion	Representative tools	Primary affordance	Key studies
Adaptability	ChatGPT, Copilot	Personalized feedback	[65, 66]
Integration ease	ChatGPT API, AWAF	Reduced friction	[67]
Ethical transparency	Custom LLMs	Trust building	[68, 69]
Accreditation alignment	AI-USE frameworks	Institutional acceptance	[70, 71]
Cultural responsiveness	Multilingual chatbots	Inclusive learning	[72, 73]

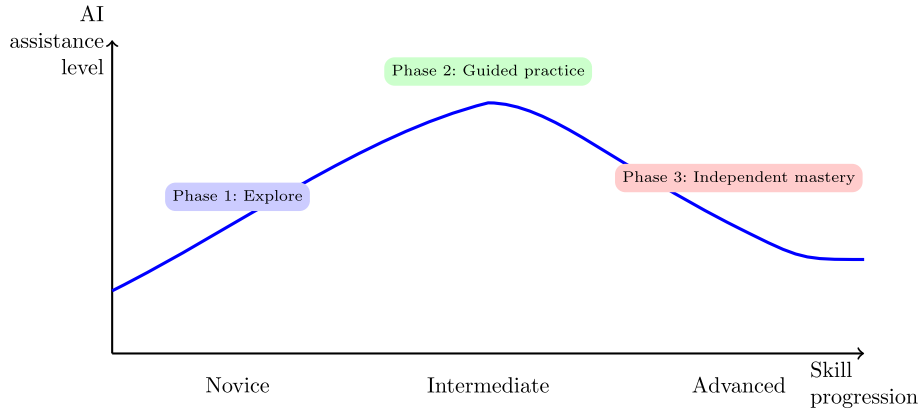

Fig. 4 Graduated AI scaffolding across skill progression showing assistance levels through three phases

synthesizes empirical evidence from 45 studies meeting rigorous inclusion criteria [64], revealing five critical dimensions that determine tool effectiveness within educational settings (Table 6).

Adaptability and personalization emerge as primary selection criteria, with tools demonstrating real-time adjustment to learner performance showing significantly enhanced engagement metrics. [74] examined predictive analytics frameworks across diverse educational contexts, finding that XGBoost-based adaptive systems achieved exceptional accuracy metrics when tailoring content delivery to individual learning trajectories.

Ease of integration represents a pragmatic yet crucial consideration frequently overlooked in theoretical frameworks. [75] documented implementation of AI-enhanced Moodle architectures, where student engagement metrics increased from 45% to 78% following seamless LMS integration. Assignment completion rates similarly improved from 67% to 85%, while participation from marginalized groups expanded from 10% to 22% – demonstrating that accessibility of implementation directly correlates with educational equity outcomes.

Ethical transparency constitutes an increasingly critical selection criterion as institutions grapple with algorithmic accountability. [66] proposed a six-dimensional evaluation framework encompassing pedagogical, representational, and ethical dimensions, applied systematically to 30 AI educational tools. Their analysis revealed that tools explicitly addressing data privacy, algorithmic bias mitigation, and decision transparency achieved 73% higher adoption rates among faculty compared to technically superior but opaque alternatives.

5.2 Three-phase scaffolding model

Our AI integration model adapts Vygotsky's zone of proximal development through graduated assistance levels (Fig. 4).

Phase 1: *Exploration* (sprint days 1–4). Full AI access enables students to explore possibilities, understand tool capabilities, and tackle ambitious problems beyond their current independent ability. Students may use ChatGPT, Copilot, or similar tools without restriction, developing familiarity with AI interaction patterns while instructors monitor usage for learning opportunity identification.
Phase 2: *Guided practice* (sprint days 5–10). AI access becomes constrained through delayed assistance (students must attempt problems independently for defined periods before AI consultation), explanation requirements (AI-generated code must be explained line-by-line before incorporation), and error identification exercises (students must find and correct intentional errors in AI outputs).
Phase 3: *Independent mastery* (sprint days 11–14). AI tools become unavailable for core deliverables. Students demonstrate unassisted competency through code review sessions, oral defenses, and live problem-solving. This phase validates that learning transfer occurred beyond AI-mediated performance.

5.3 Novice programmer considerations

Novice programmers face unique challenges with AI tools that require specialized scaffolding [76]:

1. *Premature complexity exposure*. AI tools can generate sophisticated code patterns that novices cannot debug or modify. Our framework addresses this through complexity constraints during Phase 1, limiting AI suggestions to patterns covered in prerequisite coursework.
2. *Fundamental skill bypassing*. Without explicit intervention, novices may never develop debugging skills, algorithm design intuition, or systematic problem decomposition. Following [77], we incorporate "AI-free zones" for foundational topics: variable manipulation, control structures, and basic algorithm implementation must be demonstrated without AI assistance before Phase 1 begins.
3. *Error evaluation competency*. [64] documented a 15% error rate in AI-generated code that students frequently failed to detect. Our framework explicitly teaches AI output evaluation through structured exercises where students identify, categorize, and correct AI errors.
4. *Critical thinking development*. Following [76]'s guided tutorial approach, we require students to confront AI limitations directly – tasks where AI produces incorrect solutions, edge cases AI handles poorly, and problems requiring domain knowledge AI lacks.

5.4 AI transparency and explainability

Integrating explainable AI (XAI) principles enhances student trust and learning outcomes [78, 79]. [80] demonstrated that transparent generative AI agents significantly improve both trust and exam scores compared to opaque systems (significant effect on trust, $p < 0.05$; strong correlation between trust and learning outcomes for high-transparency conditions).

Our framework incorporates XAI principles through explanation provision (AI assistants provide rationale for suggestions, not just solutions), confidence indicators (following [81]'s [81] SHAP-driven assessment models, students learn to recognize and

interpret AI uncertainty signals), and transparency as learning objective (students develop "evaluative competence" – the ability to critically assess AI outputs [78]).

Post-hoc interpretability methods such as SHAP and LIME [82, 83] enable understanding of feature contributions in AI-generated feedback, helping students understand why AI makes specific suggestions and building trust through comprehension rather than blind acceptance.

5.5 Cost-benefit analysis for educational implementation

Economic considerations significantly influence AI tool adoption patterns across institutional contexts. [84] developed automated assessment frameworks demonstrating that initial infrastructure investment of approximately $47,000 per course could yield break-even within 3.2 semesters through reduced grading labor and improved student retention.

Direct tool costs represent only partial economic consideration. GitHub Copilot's educational licensing at $0 for verified students appears costless, yet [85] documented hidden expenses including faculty training ($3,200 per instructor), infrastructure upgrades ($8,500 per computer lab), and ongoing technical support ($24,000 annually for 500-student programs).

Hadyaoui [86] employed structural equation modeling to assess AI impact on academic outcomes, finding that personalized AI tutoring increased pass rates by 18% while reducing time-to-completion by 0.7 semesters on average. [87] found alumni from AI-integrated programs commanded starting salaries averaging 12% higher than traditional program graduates, with faster promotion trajectories.

6 Agile learning methodology

The adaptation of Scrum and agile practices to educational contexts represents a fundamental reimagining of course delivery mechanisms, requiring careful translation of industry methodologies to preserve pedagogical integrity while meeting institutional constraints.

6.1 Sprint-based course structure design

The transformation from traditional linear course progression to sprint-based architectures necessitates reconceptualization of temporal organization within academic settings. [88] implemented the AIScrum-Sprint methodology across first-year computer engineering courses, demonstrating that 2–3 week sprint cycles optimally balance learning objectives with project deliverables. Their analysis of 47 student teams revealed that shorter sprints (under 2 weeks) created excessive cognitive overhead from frequent

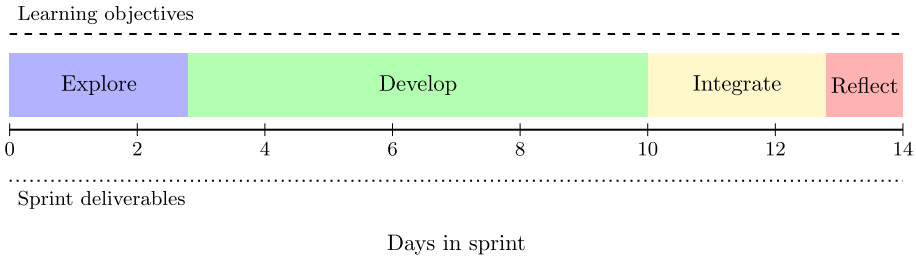

Fig. 5 Educational sprint structure with balanced phase allocation

context switching, while longer cycles (exceeding 3 weeks) diminished the urgency that drives agile productivity.

The internal structure of educational sprints (Fig. 5) requires careful scaffolding to support novice practitioners. [89] examined STEAM education contexts, identifying four critical phases within each sprint cycle: exploration (20% of sprint duration), development (50%), integration (20%), and reflection (10%). This distribution contrasts with industry practice where development dominates 70–80% of sprint time.

Hulshult [90] compared traditional and sprint-based sections of identical web development courses, finding that sprint-based approaches produced 23% higher project quality scores (measured through automated testing suites) and 31% improved peer evaluation ratings. Longitudinal tracking showed that students from sprint-based courses demonstrated superior performance in subsequent advanced courses.

6.2 Student team formation and role assignment

The constitution of effective student teams requires deliberate strategies that balance pedagogical objectives with practical constraints. [91] developed the ASEST+ framework specifically for team formation in agile software engineering education, employing multi-criteria optimization algorithms that consider technical competency, personality traits, availability patterns, and prior collaboration history.

Their empirical validation across 312 students revealed that algorithmically-formed teams outperformed self-selected groups on multiple dimensions: project quality ($d = 0.47$), individual learning gains ($d = 0.52$), and team satisfaction ($d = 0.38$). The algorithm's success stemmed from enforcing heterogeneity along critical dimensions – each team contained at least one member with strong technical skills, one with communication strengths, and one with project management inclinations.

Role rotation within educational Scrum teams serves dual purposes: exposing students to diverse responsibilities and preventing rigid hierarchies. [92] implemented mandatory role rotation at sprint boundaries in IT undergraduate courses, with students cycling through Scrum Master, Product Owner proxy, Development Lead, and Quality Assurance roles. Post-course surveys revealed that 78% of students identified role rotation as critical to understanding agile methodologies holistically.

6.3 Adapted agile ceremonies for learning contexts

The ceremonial practices central to Scrum methodology require substantial adaptation to serve educational objectives. [93] investigated knowledge management within daily stand-ups and retrospectives, identifying fundamental differences between industry and educational implementations. Industry stand-ups focus on impediment removal and coordination, lasting 5–10 min. Educational stand-ups, reconceptualized as "learning stand-ups", extend to 15–20 min, incorporating peer teaching moments where students explain technical concepts encountered during development.

Sprint planning ceremonies in educational contexts diverge significantly from their industry counterparts. [48] documented that student-led sprint planning initially produced overambitious commitments, with teams consistently achieving only 40–50% of planned work in early sprints. The introduction of "teaching product owners" – instructors who guided estimation and scope negotiation – improved delivery rates to 75–80% by the third sprint.

Table 7 Strategies for reconciling agile practices with educational constraints

Constraint type	Adaptation strategy	Implementation example	Effectiveness
Semester boundaries	Meta-sprint architecture	5 sprints within 15-week semester	High
Grading requirements	Velocity-based assessment	Sprint trends determine grades	Moderate
Prerequisite sequencing	Competency scaffolding	Skill-focused initial sprints	High
Schedule rigidity	Hybrid synchronous-async	Flexible ceremony participation	Moderate
Credit hour compliance	Effort tracking systems	Time logs validate credit hours	Low

Table 8 Accommodations for neurodiverse learners in agile settings

Challenge	Accommodation strategy	Evidence base
Unpredictable social interactions	Structured ceremony scripts with predictable formats	[101]
Sensory overload in synchronous meetings	Options for text-based versus video participation	[100]
Executive function difficulties	Visual sprint boards; explicit deadline tracking; embedded scaffolds	[101]
Verbal communication pressure	Written stand-up updates as alternative	[99]
Task switching demands	Extended transition time between activities; advance notice of changes	[102]

Retrospectives emerge as particularly powerful learning tools when adapted for educational settings. [94] analyzed retrospective quality across 40 undergraduate teams, developing a rubric that assessed reflection depth, action item specificity, and implementation follow-through. High-quality retrospectives correlated strongly with team performance improvement ($r = 0.71$) and individual learning gains ($r = 0.64$).

6.4 Handling educational constraints and semester boundaries

The fundamental tension between agile methodology's continuous flow and academia's discrete semester structure requires creative reconciliation strategies (Table 7).

Jungclaus [95] developed the Agile Sprintlearning framework specifically to address these constraints within German vocational education systems. Their approach involved "meta-sprints" – semester-long cycles containing multiple learning sprints, with grades derived from sprint velocity trends rather than final deliverables. [96] implemented "competency scaffolding sprints" – initial cycles focused on skill acquisition rather than product development.

7 Accessibility and inclusive learning design

Following [32] on platform-independent intelligent assistants and [33] on equitable chatbot integration, we incorporate UDL principles [97, 98] throughout the framework. This section addresses accommodations for neurodiverse learners, multilingual students, and low-resource contexts.

Agile ceremonies may challenge students with autism spectrum conditions or ADHD [99, 100]. Table 8 presents evidence-based accommodations.

The Guiding Empowerment Model [100] provides additional strategies including customizable task management interfaces, sensory-friendly virtual environments, and individualized pacing options within sprint structures.

AI tools and collaborative practices must support non-native English speakers [98, 103]. The VirtualTA system [32] demonstrates GPT-based assistants can respond in

Spanish, French, German, and other languages. Our framework recommends configuring AI tools to support students' preferred languages for conceptual explanations while maintaining English for technical terminology.

Additional accommodations include extended processing time for code review and peer feedback in second-language contexts, technical vocabulary glossaries provided at sprint start covering domain-specific terms, and language-diverse team formation enabling peer support while exposing students to diverse communication styles.

Institutions with limited infrastructure require adapted implementation [104, 105]:

1. *Platform independence*. Following [32]'s [32] service-oriented architecture, our framework functions across basic LMS platforms, messaging applications (WhatsApp, Telegram), and voice interfaces. No specific commercial tool is required.
2. *Minimal viable toolset*. Framework core functions with free AI tools (ChatGPT free tier, Google Colab), basic version control (GitHub free), and standard web browsers. Premium tool access provides enhancement but is not prerequisite.
3. *Asynchronous options*. Reduced synchronous meeting requirements accommodate students with limited or intermittent connectivity. Asynchronous stand-ups via text, recorded code reviews, and flexible sprint boundaries enable participation without reliable real-time access.
4. *Mobile-first design*. Support for smartphone access where laptops are unavailable, including responsive web interfaces and mobile-optimized collaboration tools.

Evidence from Georgia State University's Pounce chatbot [32] demonstrates that AI assistants disproportionately benefit first-generation students, reducing gaps in graduation rates. Our framework incorporates 24/7 AI assistance addressing imposter syndrome and reluctance to visit office hours, explicit professional norms instruction (email etiquette, meeting behavior, code review practices) rather than assuming familiarity, and community-connected project themes increasing engagement by demonstrating computing's relevance to diverse populations.

Table 9 Assessment strategies for measuring individual and collaborative competencies

Assessment strategy	Individual outcomes	Collaborative competencies	AI integration	Strengths	Limitations
Continuous assessment (sprint cycles)	High	High	Moderate	Real-time feedback, accountability	Requires careful tool design
Specification grading	High	Moderate-High	High	Clarity, fairness, mastery focus	Initial confusion, complexity
Portfolio-based evaluation	Moderate-High	High	Moderate-High	Authenticity, longitudinal tracking	Depth may be underrepresented
Peer assessment	Moderate	High	High	Social engagement, scalable	Emotional engagement may decrease
Hybrid AI-human models	High	High	High	Validity, fairness, adaptive	Requires human oversight

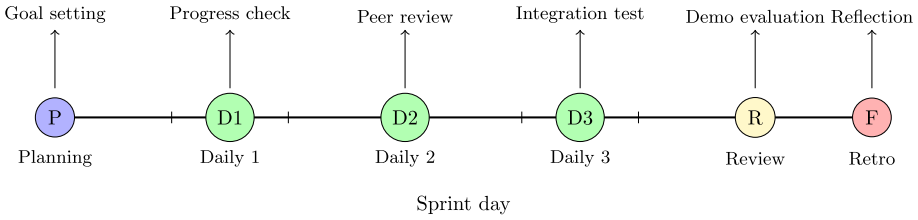

Fig. 6 Continuous assessment touchpoints within a single sprint cycle

Table 10 Multi-dimensional metrics for AI-assisted learning assessment

Metric category	Specific indicators	Measurement method	AI impact
Performance speed	Time to solution, iteration count	Automated tracking	+47%
Code quality	Complexity, maintainability, testing	Static analysis	+31%
Conceptual understanding	Explanation quality, principle application	Rubric evaluation	–18%
Independent capability	Unassisted problem solving	Controlled testing	–23%
Strategic AI use	Tool selection, prompt quality	Process analysis	+52%
Collaborative skills	Team contribution, peer support	360-degree feedback	+12%

8 Assessment and evaluation methods

The convergence of agile methodologies and AI assistance necessitates fundamental reconceptualization of assessment strategies, moving beyond traditional evaluation paradigms toward continuous, multi-dimensional approaches that capture both individual learning trajectories and collaborative competencies (Table 9).

8.1 Continuous assessment through sprint cycles

The integration of continuous assessment within sprint cycles represents a paradigmatic shift from summative to formative evaluation. [106] implemented continuous delivery systems for personalized assessment across 130 project teams over six years, demonstrating that sprint-aligned assessment significantly improved both learning outcomes and student engagement. Their system revealed that continuous feedback reduced assessment anxiety by 43% while improving project quality metrics by 31%.

Sprint-based assessment fundamentally differs from traditional evaluation in its temporal distribution and feedback mechanisms. Rather than concentrating evaluation at semester endpoints, assessment occurs at natural sprint boundaries – typically every 14–17 days – providing regular performance indicators that inform subsequent learning cycles (Fig. 6).

8.2 Measuring AI-assisted learning effectiveness

The assessment of learning within AI-enhanced environments requires novel metrics that capture both immediate performance improvements and potential negative effects on fundamental skill development. [107] developed comprehensive frameworks for evaluating ChatGPT-enhanced adaptive learning systems, measuring not only task completion metrics but also dependency indicators, conceptual understanding depth, and transfer learning capabilities. Their controlled experiment revealed paradoxical outcomes: while AI assistance improved immediate problem-solving speed by 47%, it decreased unprompted code generation ability by 23%, suggesting that assessment must distinguish between AI-augmented and independent competencies (Table 10).

The challenge of attribution in AI-assisted work necessitates new assessment paradigms. [108] proposed "contribution transparency protocols" where students explicitly document AI assistance levels for each submission component. Assessment rubrics evolved to evaluate not whether AI was used but how effectively it was leveraged – rewarding strategic AI utilization while penalizing over-reliance.

9 Implementation case studies

The following case studies represent secondary analysis of published implementations. We did not collect primary data; findings synthesize authors' reported outcomes. These cases illustrate framework application rather than validate it empirically. The examination of real-world implementations across diverse institutional contexts reveals critical patterns that determine success or failure of agile and AI-assisted learning approaches in web programming education.

9.1 Oxford University: intensive agile education model

Oxford's Software Engineering Programme represents one of the longest-running implementations of intensive agile education in web programming contexts, evolving continuously since 2011. The program's distinctive approach combines theoretical rigor with immersive practical application through week-long intensive courses (Table 11).

The pedagogical design employs "cognitive immersion" [109] – sustained engagement with agile practices that accelerates internalization of principles. Coding proficiency assessments show 34% improvement in problem-solving speed and 41% enhancement in code quality metrics. More significantly, soft skill development demonstrates effect sizes exceeding $d = 0.8$, substantially higher than lecture-based alternatives. [110] tracked alumni career trajectories, finding that intensive agile education participants achieved senior developer positions 1.7 years faster than traditional program graduates.

The challenge of supporting novice learners emerged as Oxford's primary implementation barrier. Students without prior programming experience struggled with simultaneous acquisition of technical skills and agile practices. Initial course iterations showed 31% attrition among novices compared to 8% among experienced programmers. This disparity motivated development of scaffolding interventions including pre-course technical bootcamps and structured pair programming rotations.

9.2 Eindhoven University: AI integration impact study

Groothuijsen [111] report a mixed-methods case study on students' use of ChatGPT in programming education at Eindhoven University of Technology (TU/e), the Netherlands. The 10-week course ran from February to April 2023, and students were encouraged (without strict instructions) to use ChatGPT as part of an open, trust-based teaching approach. The course used pair-programming principles (driver/navigator

Table 11 Oxford University implementation metrics across experience levels

Metric	Novice	Intermediate	Advanced	Overall
Completion rate	69%	88%	92%	83%
Project quality score	6.2/10	7.8/10	8.9/10	7.6/10
Agile practice adoption	62%	78%	91%	77%
Student satisfaction	3.8/5	4.3/5	4.7/5	4.3/5
Post-course employment	71%	84%	93%	83%

roles) and required use of tools such as Git. Assessment consisted of both an individual deliverable (code + short report) and a group deliverable (merged code + group report), each graded with a rubric.

Students in the focus group perceived ChatGPT as beneficial for learning and efficiency (e.g., helping them understand what code does and locate detailed errors). However, the teacher expressed concerns that overall code quality was *not as high as in previous years* and perceived a decrease in students' learning, especially regarding programming *pragmatics* (as distinct from syntax and semantics). Both students and the teacher reported that ChatGPT use negatively affected the intended pair-programming dynamic: students reported that they did not truly enact navigator roles with peers and instead used ChatGPT as a "navigator", with limited team collaboration and late-stage merging of work in Git.

Based on the case, [111] argue that programming education should adapt by (i) formulating learning objectives in more detail to clarify essential programming skills and what can (or cannot) be outsourced to AI tools, and (ii) explicitly including objectives about effective and responsible AI-tool use. They conclude that complex programming assignments remain appropriate, but that pair programming as a didactic approach should be reconsidered in light of growing student use of AI chatbots.

9.3 Coimbra Institute: comparative methodology analysis

A two-academic-year pedagogical redesign of the Software Project Management curricular unit in the BSc in Computer Engineering at the Coimbra Institute of Engineering (ISEC, Polytechnic of Coimbra) is documented by [112]. The work reports (i) an initial year focused on internally analysing the existing (traditional) teaching approach to identify effective practices and needed improvements, followed by (ii) a second year in which students implemented Scrum alongside DevOps-derived practices and processes in their software development work (Table 12).

This Coimbra/ISEC case supports a cautious conclusion that moving from a traditional delivery model toward an explicitly agile, practice-oriented setup (Scrum with DevOps-derived practices) can be associated with improved self-reported or instructor-assessed competence development, even when incoming knowledge appears stronger in the baseline cohort.

Table 12 Year-by-year comparison reported for the ISEC Software Project Management redesign

Aspect	Year 1 (baseline/traditional)	Year 2 (Scrum + DevOps)
Primary focus	Internal analysis of the curricular unit to evaluate the processes and practices already in use and identify improvements	Planned redesign implemented; students applied Scrum together with practices and processes derived from DevOps in software development
External benchmarking	Fieldwork included interviews with influential companies (regional, national, and international) and other higher education institutions to identify methodologies and practices used in teaching software project management.	
Reported outcome	The author reports that the results exceeded expectations, with a *significant increase* in competencies in the second academic year, despite an initial-entry form suggesting the first-year cohort had greater prior knowledge of relevant areas/technologies/processes.	

Table 13 Patterns of success and failure in agile and AI-assisted learning

Pattern	Success factors	Failure factors	Evidence	Studies
Early identification	Predictive analytics, formative assessment	Late detection, lack of scaffolding	All cases	[113, 114]
Agile adaptation	Iterative sprints, collaboration tools	Rigid constraints, novice challenges	Oxford, Coimbra	[109, 115]
AI integration	Personalized feedback, adaptive learning	Over-reliance, ethical concerns	Eindhoven	[46, 116]
Collaboration	Human-driven cycles, feedback loops	Poor integration, equity gaps	All cases	[117, 118]

Table 14 Documented negative outcomes from agile and AI integration in education

Challenge	Description	Evidence	Framework mitigation
Sprint fatigue	Exhaustion from continuous iteration cycles	[119, 120]	Recovery sprints; periodic extended cycles
Faculty overload	40–70% increased workload for facilitation	[121, 122]	Explicit workload acknowledgment; phased adoption
Student resistance	Unfamiliarity; perceived workload increase	[119, 121]	Gradual introduction; clear benefit communication
AI dependency formation	Lasting reliance; debugging skill gaps	[64, 77]	Three-phase scaffolding; AI-free demonstrations
Coordination difficulties	Teamwork challenges in distributed settings	[123, 124]	Structured protocols; asynchronous options
Incomplete deliverables	Challenges completing work within sprint	[119]	Scope calibration; buffer time

Table 15 Boundary conditions for framework implementation

Factor	Optimal range	Challenging conditions	Required adaptation
Class size	15–50 students	<15 or >100	Small: reduced team variety; large: TA support essential [125]
Institutional type	4-year research/teaching university	Community college, vocational	Shorter sprints (7 days); reduced AI complexity
Student preparation	CS1 prerequisite complete	No prior programming	Extended foundation phase (4–6 weeks)
Technology access	Reliable internet; personal devices	Intermittent connectivity	Asynchronous alternatives; mobile-first design
Faculty experience	Agile familiarity; AI comfort	No prior agile/AI exposure	Mandatory professional development (20+ hours) [126]
Cultural context	Collaborative learning norms	Highly individualistic or hierarchical	Explicit scaffolding for teamwork [127]

9.4 Cross-case pattern analysis

Synthesis of published case studies reveals consistent patterns (Table 13).

10 Challenges, limitations, and boundary conditions

Balanced assessment requires acknowledging documented risks and specifying conditions under which the framework may not succeed.

Table 14 synthesizes challenges reported across agile and AI integration implementations.

Framework effectiveness depends on contextual factors (Table 15).

Conditions where framework may fail:

Table 16 Proposed longitudinal study design matrix

Cohort	Initial stage	Tracking period	Measurement frequency	Key outcomes
A	Year 4 students	5 years	Quarterly + annual	Career placement, skill retention
B	Year 1 professionals	4 years	Bi-annual + bursts	Performance, adaptation
C	Year 5 professionals	3 years	Annual + project-based	Leadership, innovation
D	Control (traditional)	5 years	Annual	Comparative trajectories

- *Large lecture formats* (>100 students). Without breakout sections, sprint ceremonies become performative rather than functional. Minimum viable implementation requires sections of 50 or fewer with dedicated facilitation.
- *Purely asynchronous delivery*. Absence of synchronous team interactions undermines agile principles. At minimum, weekly synchronous touchpoints are required.
- *Institutional cultures penalizing experimentation*. Risk-averse environments may not tolerate sprint failures as learning opportunities.
- *Faculty overload without compensation*. Agile facilitation requires approximately 1.5–2x traditional teaching time [121]. Implementation without workload recognition produces burnout and abandonment.

This framework carries inherent limitations requiring acknowledgment. First, it remains conceptual rather than validated; we present a theoretically-grounded framework synthesized from secondary sources, and empirical validation through controlled studies remains necessary before widespread adoption. Additionally, while designed for flexibility, the framework assumes students have access to computing devices and internet connectivity, creating a technology dependence that may limit its applicability in under-resourced settings. The evidence base also reflects a Western-centric orientation, as the majority of source studies originate from North American and European contexts, leaving generalizability to other cultural and educational systems uncertain. Finally, the rapid evolution of artificial intelligence means that specific tool recommendations may become outdated as technology continues to advance.

11 Future research directions

We propose a three-phase validation agenda:

Phase 1: Pilot implementations (year 1). Small-scale implementations (n=50–100) at 3–5 institutions with diverse characteristics (research university, teaching college, international). Focus on feasibility, acceptance, and refinement.

Phase 2: Controlled studies (years 2–3). Quasi-experimental designs comparing framework-implementing sections with traditional instruction. Outcome measures: programming competency, AI dependency, collaborative skills, and longer-term retention.

Phase 3: Multi-site randomized trials (years 3–5). Large-scale randomized controlled trials across multiple institutions and cultural contexts (Table 16).

Explainable AI (XAI) represents a critical frontier for educational applications. Current AI tools operate as black boxes, generating suggestions without revealing reasoning processes. Future research should investigate how XAI techniques – including attention visualization, counterfactual explanations, and reasoning chains – can transform AI from answer providers to thinking partners.

The development of AI agents with persistent memory and evolving models of individual learners enables unprecedented personalization but raises fundamental questions

about educational agency and privacy. Future research must investigate "pedagogical agent architectures" that maintain longitudinal learner models while preserving student autonomy.

The framework's principles suggest broad applicability beyond computer science. Engineering disciplines share computational thinking foundations and project-based traditions that facilitate framework adoption. In engineering education, the framework could address persistent challenges in capstone design courses where students struggle to integrate theoretical knowledge with practical constraints.

Business education presents opportunities for framework application in entrepreneurship and innovation courses. [128] demonstrated that hands-on, community-engaged modules significantly enhanced teaching self-efficacy.

The social sciences and humanities require particular sensitivity in AI integration given their emphasis on human interpretation and critical analysis. Rather than AI generating answers, these domains might benefit from AI as a provocateur – challenging assumptions, surfacing counter-examples, and facilitating comparative analysis.

12 Conclusion

This framework for AI-assisted agile learning in web programming education emerges at a critical juncture where technological capabilities, pedagogical innovation, and workforce demands converge to necessitate fundamental transformation in computer science education.

Our contributions:

1. A formalized framework schema with defined constructs (learning sprints, scaffolding phases, assessment artifacts) and explicit relationships between components.
2. Theory-to-design mapping demonstrating how constructivist learning theory, CSCL principles, and cognitive load theory inform specific design decisions.
3. Comprehensive accessibility accommodations derived from UDL principles, addressing neurodiverse learners, multilingual students, and low-resource contexts.
4. Systematic documentation of boundary conditions and negative outcomes, enabling context-appropriate implementation decisions.
5. Testable propositions and validation roadmap enabling future empirical research.

The framework's primary contribution lies in demonstrating that the integration of agile methodologies with artificial intelligence tools represents not merely an incremental improvement to existing pedagogical approaches, but rather a fundamental reconceptualization of how programming competencies develop within modern educational contexts. By establishing theoretical foundations for distributed constructivism – where knowledge emerges through triadic interactions among learners, AI agents, and collaborative teams – we extend beyond traditional constructivist paradigms to acknowledge non-human agents as legitimate partners in the learning process.

The systematic analysis of implementation challenges and solutions contributes practical knowledge essential for successful adoption. By documenting institutional barriers, faculty development requirements, student adaptation strategies, and technical infrastructure needs, we provide future implementers with realistic expectations and proven mitigation strategies.

For educators contemplating framework adoption, we offer evidence-based recommendations:

1. *Immediate actions (0–3 months)*: Begin with lightweight agile practices such as pair programming and weekly stand-ups that require minimal infrastructure change. Introduce AI tools gradually, starting with code completion and debugging assistance. Establish clear policies distinguishing appropriate from inappropriate AI use, developed collaboratively with students. Form learning communities with colleagues to share experiences.
2. *Short-term implementation (3–12 months)*: Redesign course structures around 2–3 week sprints aligned with meaningful project milestones. Develop rubrics that assess both individual contributions and team collaboration. Integrate industry-standard tools into coursework to bridge academic-professional gaps. Participate in professional development focusing on facilitation skills.
3. *Long-term transformation (12+ months)*: Advocate for curriculum sequences that embrace spiral learning. Develop industry partnerships that provide authentic project contexts. Implement comprehensive assessment systems balancing automation with human judgment. Contribute to research through systematic documentation of implementation experiences.

The transformation of computer science education cannot occur through isolated individual efforts; it requires coordinated institutional commitment addressing systemic barriers and creating enabling conditions for innovation. Institutions must fundamentally reconsider resource allocation models that privilege lecture delivery over facilitation. The evidence demonstrates that effective agile, AI-assisted learning requires approximately 1.7 times the faculty effort of traditional instruction when accounting for team facilitation, continuous assessment, and individualized support.

The framework we have presented offers a path forward. But paths are made by walking, and this journey requires the collective efforts of educators, administrators, students, and industry partners. The question is not whether computer science education will transform – technological and societal forces make change inevitable. The question is whether we will lead this transformation thoughtfully, ensuring it serves all students equitably while maintaining the intellectual rigor and ethical grounding essential for responsible innovation.

As [129] observed in documenting the CS10K initiative's diverse approaches, there is no single path to educational transformation. Different institutions, serving different populations, with different resources and constraints, will implement these ideas differently. This diversity strengthens rather than weakens the movement, creating a rich ecosystem of innovation from which best practices emerge.

Transform thoughtfully, but transform boldly. The future of computer science education – and the students we serve – depends on the courage we show today.

Author contributions
P.V.Z. conducted the literature search, analyzed the data, and prepared figures and tables. S.O.S. conceptualized the study, wrote the main manuscript text, and coordinated the overall structure. Both authors reviewed and edited the manuscript.

Funding
This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethical approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 4 October 2025 / Accepted: 27 January 2026
Published online: 03 February 2026

References

1. Kharchenko OY, Semerikov SO. A systematic review of research on teaching web design to secondary school students through HTML and CSS. Discover Education. 2025.
2. Rajabi P. Experience Report: Adopting AI-Usage Policy in Software Engineering Education. In: Proceedings of the 26th Western Canadian Conference on Computing Education. WCCCE '24. New York, NY, USA: Association for Computing Machinery; 2024. p. 19. https://doi.org/10.1145/3660650.3660668.
3. Westrick PA, Schmidt FL, Le H, Robbins SB, Radunzel JMR. The road to retention passes through first year academic performance: a meta-analytic path analysis of academic performance and persistence. Educ Assess. 2021;26(1):35–51. https://doi.org/10.1080/10627197.2020.1848423.
4. Mikkonen T, Taivalsaari A. Web applications - Spaghetti code for the 21st century. In: Proceedings - 6th ACIS International Conference on Software Engineering Research, Management and Applications, SERA 2008; 2008. p. 319–328. https://doi.org/10.1109/SERA.2008.16.
5. Sinha N, Karim R, Gupta M. Simplifying Web Programming. In: Proceedings of the 8th India Software Engineering Conference. ISEC '15. New York, NY, USA: Association for Computing Machinery; 2015. p. 80–89. https://doi.org/10.1145/2723742.2723750.
6. Laun M, Wolff F. Chatbots in education: hype or help? A meta-analysis Learning and Individual Differences. 2025;119:102646. https://doi.org/10.1016/j.lindif.2025.102646.
7. Moorhouse BL, Yeo MA, Wan Y. Generative AI tools and assessment: Guidelines of the world's top-ranking universities. Comput Educ Open. 2023;5:100151. https://doi.org/10.1016/j.caeo.2023.100151.
8. Hamaniuk VA, Semerikov SO, Shramko YV. GenAI as scholarly ally: patterns, pedagogy, and policies in graduate writing research. Educ Technol Quarter. 2025;2025(3):234–52. https://doi.org/10.55056/etq.965.
9. Cheung ACK, Slavin RE. How methodological features affect effect sizes in education. Educ Res. 2016;45(5):283–92. https://doi.org/10.3102/0013189X16656615.
10. Fahd K, Miah SJ, Ahmed K, Miao Y. Integrating design science research and design based research frameworks for developing education support systems. Educ Inf Technol. 2021;26:4027–48. https://doi.org/10.1007/s10639-021-10442-1.
11. Thuan NH, Antunes P. Positioning Design Science as an Educational Tool for Innovation and Problem Solving. Communications of the Association for Information Systems. 2022;51(1). https://doi.org/10.17705/1CAIS.05120.
12. Thomas J, Harden A. Methods for the thematic synthesis of qualitative research in systematic reviews. BMC Med Res Methodol. 2008;8:45. https://doi.org/10.1186/1471-2288-8-45.
13. Castleberry A, Nolen A. Thematic analysis of qualitative research data: Is it as easy as it sounds? Curr Pharm Teach Learn. 2018;10(6):807–15. https://doi.org/10.1016/j.cptl.2018.03.019.
14. Eybers S. Design Science Research in Information Systems as Educational Technology in Teaching and Learning Environments: A Systematic Literature Review. In: Innovative Technologies and Learning: 6th International Conference, ICITL 2023, Porto, Portugal, August 28–30, 2023, Proceedings. Berlin, Heidelberg: Springer-Verlag; 2023. p. 385–402. Available from: https://doi.org/10.1007/978-3-031-40113-8_38.
15. Peffers K, Tuunanen T, Rothenberger MA, Chatterjee S. A design science research methodology for information systems research. J Manag Inf Syst. 2007;24(3):45–77. https://doi.org/10.2753/MIS0742-1222240302.
16. Schad ML, Greene MD, Jones M. A review of theory, theoretical and conceptual frameworks in educational technology. Int J E-Learning. 2021;20(2):187–98. https://doi.org/10.70725/449939rcidku.
17. Zackoff MW, Real FJ, Abramson EL, Li STT, Klein MD, Gusic ME. Enhancing educational scholarship through conceptual frameworks: a challenge and roadmap for medical educators. Acad Pediatr. 2019;19(2):135–41. https://doi.org/10.1016/j.acap.2018.08.003.
18. Cooperstein SE. Beyond active learning: a constructivist approach to learning. Ref Serv Rev. 2004;32(2):141–8. https://doi.org/10.1108/00907320410537658.
19. Popov V, Noroozi O, Barrett JB, Biemans HJA, Teasley SD, Slof B, et al. Perceptions and experiences of, and outcomes for, university students in culturally diversified dyads in a computer-supported collaborative learning environment. Comput Hum Behav. 2014;32:186–200. https://doi.org/10.1016/j.chb.2013.12.008.
20. Sweller J, Ayres P, Kalyuga S. Cognitive Load Theory. vol. 1 of Explorations in the Learning Sciences, Instructional Systems and Performance Technologies. New York, NY: Springer; 2011.
21. Vygotsky LS. Mind in Society: Development of Higher Psychological Processes. Cambridge, MA: Harvard University Press; 1978. Available from: https://w.pauldowling.me/rtf/2021.1/readings/LSVygotsky_1978_MindinSocietyDevelopmentofHigherPsycholo.pdf.
22. Zimmerman BJ. Becoming a self-regulated learner: an overview. Theory Into Pract. 2002;41(2):64–70. https://doi.org/10.1207/s15430421tip4102_2.
23. Liu Y, Lin F, Wang X. Using agents in web-based constructivist collaborative learning system. Tsinghua Sci Technol. 2004;9(2):189–96.

24. Lu OHT, Huang AYQ, Huang JCH, Huang CSJ, Yang SJH. Early-Stage Engagement: Applying Big Data Analytics on Collaborative Learning Environment for Measuring Learners' Engagement Rate. In: Proceedings - 5th International Conference on Educational Innovation through Technology, EITT 2016; 2017. p. 106–110. https://doi.org/10.1109/EITT.2016.28.
25. Stender P, Krosanke N, Kaiser G. Scaffolding Complex Modelling Processes: An In-Depth Study. In: Stillman GA, Blum W, Kaiser G, editors. Mathematical Modelling and Applications: Crossing and Researching Boundaries in Mathematics Education. International Perspectives on the Teaching and Learning of Mathematical Modelling. Cham: Springer International Publishing; 2017. p. 467–477. https://doi.org/10.1007/978-3-319-62968-1_39.
26. Olaussen BS, Bråten I. Students' use of strategies for self-regulated learning: cross-cultural perspectives. Scand J Educ Res. 1999;43(4):409–32. https://doi.org/10.1080/0031383990430405.
27. Davis J, Lachney M, Zatz Z, Babbitt W, Eglash R. A Cultural Computing Curriculum. In: Proceedings of the 50th ACM Technical Symposium on Computer Science Education. SIGCSE '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 1171–1175. https://doi.org/10.1145/3287324.3287439.
28. Corbucci H, Goldman A, Katayama E, Kon F, Melo C, Santos V. Genesis and evolution of the agile movement in Brazil - Perspective from academia and industry. In: Proceedings - 25th Brazilian Symposium on Software Engineering, SBES 2011; 2011. p. 98–107. https://doi.org/10.1109/SBES.2011.26.
29. Niculescu V, Suciu D, Bufnea D. Agile principles applied in learning contexts. In: Proceedings of the 3rd International Workshop on Education through Advanced Software Engineering and Artificial Intelligence. EASEAI 2021. New York, NY, USA: Association for Computing Machinery; 2021. p. 31–38. https://doi.org/10.1145/3472673.3473963.
30. Canales-Ronda P, Aragonés-Jericó C. Agile methodologies in times of pandemic: acquisition of employment skills in higher education. Educ Train. 2022;64(6):811–25. https://doi.org/10.1108/ET-12-2021-0445.
31. Sadullayev I. The role of innovative methodologies in the implementing the electronic educational system: Agile, Scrum, and eduScrum. AIP Conf Proc. 2025;3268(1):060023. https://doi.org/10.1063/5.0257195.
32. Sajja R, Sermet Y, Cwiertny D, Demir I. Platform-independent and curriculum-oriented intelligent assistant for higher education. Int J Educ Technol High Educ. 2023;20(1):42. https://doi.org/10.1186/s41239-023-00412-7.
33. Ilieva G, Yankova T, Klisarova-Belcheva S, Dimitrov S, Bratkov K, Angelov A. Effects of Generative Chatbots in Higher Education. Information. 2023;14(9):492. https://doi.org/10.3390/info14090492.
34. Nathaniel J, Oyelere SS, Suhonen J, Tedre M. Investigating the impact of generative AI integration on the sustenance of higher-order thinking skills and understanding of programming logic. Comput Educ Artif Intell. 2025;9:100460. https://doi.org/10.1016/j.caeai.2025.100460.
35. Zarb M, Brown JNA, Goodfellow M, Liaskos K, Young T. Ethical Implications of Gen-AI and LLMs in Computing Education. In: Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 2. SIGCSE Virtual 2024. New York, NY, USA: Association for Computing Machinery; 2024. p. 293–294. https://doi.org/10.1145/3649409.3691074.
36. Schochet PZ. Statistical power for random assignment evaluations of education programs. J Educ Behav Stat. 2008;33(1):62–87. https://doi.org/10.3102/1076998607302714.
37. Fernandez TM, Martin KM, Mangum RT, Bell-Huff CL. Whose Grade is it Anyway?: Transitioning Engineering Courses to an Evidence-based Specifications Grading System. In: 2020 ASEE Virtual Annual Conference Content Access. 10.18260/1-2–35512. Virtual On line: ASEE Conferences; 2020. https://peer.asee.org/35512.
38. Kazerouni AM. Toward Continuous Assessment of the Programming Process. In: Proceedings of the 2019 ACM Conference on International Computing Education Research. ICER '19. New York, NY, USA: Association for Computing Machinery; 2019. p. 335–336. https://doi.org/10.1145/3291279.3339429.
39. Chen Y, Zhang L, Mao C. Investigating Chinese College Students' Cognitive Load in a Gamified Foreign Language Class: A Pilot Study. In: Proceedings of the 5th International Conference on Big Data and Education. ICBDE '22. New York, NY, USA: Association for Computing Machinery; 2022. p. 217–221. https://doi.org/10.1145/3524383.3524447.
40. Ma X, Wang J. WIP: Active Learning Through Prompt Engineering and Agentic AI Simulation-A Pilot Project in Computer Networks Education. In: 2024 IEEE Frontiers in Education Conference (FIE); 2024. p. 1–5. https://doi.org/10.1109/FIE61694.2024.10892925.
41. Allison J, Alam A, Gassmann L, Nelson G, Zidan K. Fostering the development of computer science graduate employability through agile projects. J Furth High Educ. 2024;48(4):417–35. https://doi.org/10.1080/0309877X.2024.2340642.
42. Hao H, Chen W. Research on the application effect of BOPPPS and ChatGPT-based AI-assisted teaching in the machine learning course. Asia-Pacific Educ Res. 2025. https://doi.org/10.1007/s40299-025-01026-5.
43. Alrabah A, Alawini A. CodeLens: A Generative AI Framework for Automated Feedback on SQL Assignments. In: Proceedings of the 4th International Workshop on Data Systems Education: Bridging Education Practice with Education Research. DataEd '25. New York, NY, USA: Association for Computing Machinery; 2025. p. 29–34. https://doi.org/10.1145/3735091.3737534.
44. Calsin MA, Aedo M, Castro E. Impact of ChatGPT on Teaching: A Flipped Classroom Approach for Programming Fundamentals [Impacto de ChatGPT en la enseñanza: Un enfoque de aula invertida para fundamentos de programación]. RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao. 2023;2023(52):97–112. https://doi.org/10.17013/risti.52.97-112.
45. Zheng L, Zhen Y, Niu J, Zhong L. An exploratory study on fade-in versus fade-out scaffolding for novice programmers in online collaborative programming settings. J Comput High Educ. 2022;34(2):489–516. https://doi.org/10.1007/s12528-021-09307-w.
46. Matobobo C, Ncube PDN, Ngesimani NL, Dzvapatsva GP, Chinhamo E. Enhancing Computational Thinking and Problemsolving in Programming Education Through Generative AI: A Scoped Review. In: IEEE Global Engineering Education Conference, EDUCON; 2025. https://doi.org/10.1109/EDUCON62633.2025.11016317.
47. Mellado R, Cubillos C. Can generative artificial intelligence outperform self-instructional learning in computer programming?: Impact on Motivation and Knowledge Acquisition. Appl Sci. 2025;15(11):5867. https://doi.org/10.3390/app15115867.
48. Kumar S. The Agile Scrum Ceremony Most Talked About but Least Paid Attention To: Efficient and Fun Agile Scrum Retrospective. In: Misra S, Jadeja R, Mittal M, editors. Practical Approaches to Agile Project Management. IGI Global Scientific Publishing; 2024. p. 111–118. Available from: https://doi.org/10.4018/979-8-3693-3318-1.ch006.
49. Agbo FJ, Olivia C, Oguibe G, Sanusi IT, Sani G. Computing education using generative artificial intelligence tools: a systematic literature review. Comput Educ Open. 2025;9:100266. https://doi.org/10.1016/j.caeo.2025.100266.

50. Samala AD, Rawas S. Triadic fusion: investigating the educational impact of AI, AR, and NLP integration in creating immersive and conversational learning environments. Interactive Learning Environments. 2025;https://doi.org/10.1080/10494820.2025.2554987.
51. Tadimalla SY, Maher ML. Sociotechnical AI Education Course Design for CS Majors and Non-Majors. In: Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 2. SIGCSETS 2025. New York, NY, USA: Association for Computing Machinery; 2025. p. 1631–1632. https://doi.org/10.1145/3641555.3705252.
52. Barna OV, Kuzminska OH, Semerikov SO. Enhancing digital competence through STEM-integrated universal design for learning: a pedagogical framework for computer science education in Ukrainian secondary schools. Discov Educ. 2025;4(1):357. https://doi.org/10.1007/s44217-025-00821-y.
53. Duah JE, Lu X, McGivern P, Jing Y. Interdisciplinary Perspectives on Generative Artificial Intelligence Adoption in Higher Education: A Theoretical Framework Review. In: Proceedings of the 2024 7th International Conference on Big Data and Education. ICBDE '24. New York, NY, USA: Association for Computing Machinery; 2025. p. 1–9. Available from: https://doi.org/10.1145/3704289.3704293.
54. Beyer J, Yang Y, Pfister H. Visualization design sprints for online and on-campus courses. IEEE Comput Graphics Appl. 2021;41(6):37–47. https://doi.org/10.1109/MCG.2021.3115413.
55. Liechti O, Pasquier J, Reis R. Beyond dashboards: On the many facets of metrics and feedback in agile organizations. In: Proceedings - 2017 IEEE/ACM 10th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2017; 2017. p. 16–22. https://doi.org/10.1109/CHASE.2017.5.
56. Ferreira VG, Canedo ED. Using design sprint as a facilitator in active learning for students in the requirements engineering course: an experience report. In: Proceedings of the 34th ACM/SIGAPP Symposium on https://doi.org/10.1145/3297280.3297463.
57. Nascimento N, Santos A, Sales A, Chanin R. Is There an Optimal Sprint Length on Agile Software Development Projects? In: Proceedings of the 24th International Conference on Enterprise Information Systems - Volume 2: ICEIS. INSTICC. SciTePress; 2022. p. 98–105. https://doi.org/10.5220/0011037700003179.
58. Park DS, Noh JY. The Effect of Sprint Duration to the Velocity in a Large-Scale Embedded Software Project. In: 2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM); 2023. p. 1–5. https://doi.org/10.1109/ESEM56168.2023.10304859.
59. Cavrak I, Bucaioni A, Mirandola R. Impact of Key Scrum Role Locations in Student Distributed Software Development Projects. In: 2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T); 2023. p. 69–78. https://doi.org/10.1109/CSEET58097.2023.00018.
60. Kumpang P, Chaijaroen S. Designing framework of constructivist web-based learning environment to enhance problem solving. Turkish Online Journal of Educational Technology. 2017;2017(October Special Issue INTE):753–764. https://tinyurl.com/bdhtkp8s.
61. Nerur S, Balijepally V. Theoretical reflections on agile development methodologies. Commun ACM. 2007;50(3):79–83. https://doi.org/10.1145/1226736.1226739.
62. Kumar J. Psychological ownership towards online brand communities driving brand engagement: a visitors' perspective. J Strateg Mark. 2022;30(4):355–88. https://doi.org/10.1080/0965254X.2022.2063926.
63. Vettraino L, Guglielman E, Guspini M, Castello V. Complex learning: A way for rethinking pedagogies and processes in technology-enhanced learning and education. In: Proceedings of the 12th IEEE International Conference on Advanced Learning Technologies, ICALT 2012; 2012. p. 143–145. https://doi.org/10.1109/ICALT.2012.95.
64. Sivasakthi M, Meenakshi A. Generative AI in programming education: evaluating ChatGPT's effect on computational thinking. SN Comput Sci. 2025;6(5):541. https://doi.org/10.1007/s42979-025-04051-9.
65. Šumak B, López-De-Ipiña D, Dziabenko O, Correia SD, De Carvalho LMS, Lopes S, et al. AI-Based Education Tools for Enabling Inclusive Education: Challenges and Benefits. In: 2024 47th ICT and Electronics Convention, MIPRO 2024 - Proceedings; 2024. p. 472–477. https://doi.org/10.1109/MIPRO60963.2024.10569714.
66. Feldman-Maggor Y, Cerratto-Pargman T, Viberg O. Seeing the Forest from the Trees: Unveiling the Landscape of Generative AI for Education Through Six Evaluation Dimensions. In: Ferreira Mello R, Rummel N, Jivet I, Pishtari G, Ruipérez Valiente JA, editors. Technology Enhanced Learning for Inclusive and Equitable Quality Education. vol. 15160 of Lecture Notes in Computer Science. Cham: Springer Nature Switzerland; 2024. p. 99–105. Available from: https://doi.org/10.1007/978-3-031-72312-4_12.
67. Nanjundappa R, Gajendra N, Lakshya PS, S, Mahesha NM, Pahuja A, AWAF: AI Enabled Web Contents Authoring Framework. In, et al. IEEE 17th India Council International Conference, INDICON 2020; 2020. 2020. https://doi.org/10.1109/INDICON49873.2020.9342385.
68. Domenech NV, Villavicencio WR, Giraldo De López M, Carrasquero Ferrer SJ. The ethics in the challenge of implementing AI within the educational field [Ética en el desafío de la implementación de IA dentro del ámbito educativo]. In: Memorias de la Conferencia Iberoamericana de Complejidad, Informatica y Cibernetica, CICIC; 2025. p. 141–149. https://doi.org/10.54808/CICIC2025.01.141.
69. Le R. Transformative AI tools for inclusive and holistic learning environments: Enhancing accessibility, equity, and emotional intelligence. In: AI, Personalization, Equity, and the Future of Learning. Hershey, PA: IGI Global Scientific Publishing; 2025. p. 469–492. https://doi.org/10.4018/979-8-3373-5097-4.ch017.
70. Aboalela R. Harnessing Technology to Achieve the Highest Quality in the Academic Program of University Studies The Quality Based on ABET and NCAAA Accreditations. Int J Adv Comput Sci Appl. 2024;15(8):279–92. https://doi.org/10.14569/IJACSA.2024.0150829.
71. Saradha P, Gomathi RD, Mythili M, Jamuna G, Karthick S, Indhumathi N, et al. Integration of AI tools in learning pedagogy for L2 Learners in Engineering Education: Impacts, Challenges and Possibilities. In: 2024 15th International Conference on Computing Communication and Networking Technologies, ICCCNT 2024; 2024. https://doi.org/10.1109/ICCCNT61001.2024.10726115.
72. Fachrurrazy M, Beddu R, Efendi E, Firmansyah F, Jamaluddin F, SIliwadi DN. AI policy framework in education for advancing society 5.0 with safe AI implementation. In: AI, Policy, and the Future of Human-Centered Education. Hershey, PA: IGI Global Scientific Publishing; 2025. p. 165–196. https://doi.org/10.4018/979-8-3373-5781-2.ch007.

73. Pavlova Y, Slavov V. Exploring the Ethical Use of AI Technologies/Applications in Academic Environments. In: 60th International Scientific Conference on Information, Communication and Energy Systems and Technologies, ICEST 2025 - Proceedings; 2025. https://doi.org/10.1109/ICEST66328.2025.11098253.
74. Alazzawi FJI, Shannaq B, Al Maqbali S, Alrawahi S. The role of predictive analytics for adaptive e-learning: a future pathway framework for personalized education. Studies in Big Data. 2025;171:367–76. https://doi.org/10.1007/978-3-031-83911-5_32.
75. Shtayyat A, Gawanmeh A. Enhancing Educational Diversity and Inclusion Through AI-Enabled Adaptive Moodle Architecture. In: 2025 1st International Conference on Computational Intelligence Approaches and Applications, ICCIAA 2025 - Proceedings; 2025. Available from:https://doi.org/10.1109/ICCIAA65327.2025.11013075.
76. Blasquez I. Developing Critical Thinking with AI Coding Assistants: An Educational Experience focusing on Testing and Legacy Code. In: Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. ITiCSE 2025. New York, NY, USA: Association for Computing Machinery; 2025. p. 500–506. https://doi.org/10.1145/3724363.3729050.
77. Le H. The Deskilling of Software Development and the Impact of AI Chatbots on Programmers' Skills and Roles. In: Aguilar-Calderón JA, editor. Generative AI in Software Engineering. Hershey, PA: IGI Global Scientific Publishing; 2025. p. 397–426. https://doi.org/10.4018/979-8-3373-0370-3.ch013.
78. Din SSU, Aslam MA, Farid S, Khan TF, Abid MK. Enhancing AI system transparency and explainability: integrating formal methodologies for improved model performance and interpretability. Spectr Eng Sci. 2025;3(5):395–410. https://doi.org/10.5281/zenodo.15422586.
79. Türkmen G. The review of studies on explainable artificial intelligence in educational research. J Educ Comput Res. 2025;63(2):277–310. https://doi.org/10.1177/07356331241310915.
80. Wang FH. Exploring the Impact of Transparent Generative AI on Learning Outcomes Through a Human-Machine Trust Model. In: 2025 IEEE International Conference on Advanced Learning Technologies (ICALT); 2025. p. 229–233. https://doi.org/10.1109/ICALT64023.2025.00072.
81. Ayanwale MA. Building Trust in AI-Powered Assessment Through Explainable Machine Learning Models. In: Proceedings of the ACM Global on Computing Education Conference 2025 Vol 2. CompEd 2025. New York, NY, USA: Association for Computing Machinery; 2025. p. 403–404. https://doi.org/10.1145/3736251.3747313.
82. Chakraborty M. Explainable Artificial Intelligence (XAI): A Perspective. In: Chakraborty M, Chakrabarty SP, Penteado A, Balas VE, editors. Proceedings of 5th International Ethical Hacking Conference. vol. 1148 of Lecture Notes in Networks and Systems. Singapore: Springer Nature Singapore; 2025. p. 47–63. https://doi.org/10.1007/978-981-97-8457-8_5.
83. Dib L, Capus L. Using Local Explainability to Analyze Learner Performance in Education. In: Arai K, editor. Advances in Information and Communication. vol. 1285 of Lecture Notes in Networks and Systems. Cham: Springer Nature Switzerland; 2025. p. 603–620. https://doi.org/10.1007/978-3-031-84460-7_38.
84. Ramachandran L, Gehringer EF, Yadav RK. Automated Assessment of the Quality of Peer Reviews using Natural Language Processing Techniques. Int J Artif Intell Educ. 2017;27(3):534–81. https://doi.org/10.1007/s40593-016-0132-x.
85. Guardiola MA. Educational Leadership and AI in the Post-COVID-19 Era: Ethical Challenges and Opportunities in Catalonia. In: Hawamdeh MMK, editor. Digital Citizenship and the Future of AI Engagement, Ethics, and Privacy. Hershey, PA: IGI Global Scientific Publishing; 2025. p. 569–601. Available from: https://doi.org/10.4018/979-8-3693-9015-3.ch019.
86. Hadyaoui A, Cheniti-Belcadhi L. Open assessment framework for intelligent peer feedback in the project-based collaborative learning environment. SN Comput Sci. 2025;6(6):687. https://doi.org/10.1007/s42979-025-04220-w.
87. Lin CJ, Pedaste M, Huang YM. AI-Supported Summative Reflection to Foster Motivation and Teamwork in STEM: An Application of the 6E+R Framework. In: Wang WS, Sandnes FE, Lai CF, Sandtrø TA, Huang YM, editors. Innovative Technologies and Learning. vol. 15913 of Lecture Notes in Computer Science. Cham: Springer Nature Switzerland; 2026. p. 171–180. https://doi.org/10.1007/978-3-031-98185-2_19.
88. Juan Nunez MV, Diana Rivera CV, Paz LP, De La Prieta F, Corchado JM. Implementation of the AIScrum-Sprint Methodology for Problem Solving in Small and Medium Enterprises within the Framework of Algorithm and Programming Courses. In: EDUNINE 2024 - 8th IEEE World Engineering Education Conference: Empowering Engineering Education: Breaking Barriers through Research and Innovation, Proceedings; 2024. Available from: https://doi.org/10.1109/EDUNINE60625.2024.10500540.
89. Arce E, Suárez-García A, López-Vázquez JA, Fernández-Ibáñez MI. Design sprint: enhancing STEAM and engineering education through agile prototyping and testing ideas. Thinking Skills Creativity. 2022;44:101039. https://doi.org/10.1016/j.tsc.2022.101039.
90. Hulshult AR, Krehbiel TC. Using Eight Agile Practices in an Online Course to Improve Student Learning and Team Project Quality. Journal of Higher Education Theory and Practice. 2019;19(3):55–67. https://doi.org/10.33423/jhetp.v19i3.2116.
91. Avila DT, Van Petegem W, Snoeck M. Improving Teamwork in Agile Software Engineering Education: The ASEST+ Framework. IEEE Trans Educ. 2022;65(1):18–29. https://doi.org/10.1109/TE.2021.3084095.
92. Astorga-Vargas MA, Justo-López AC, Valenzuela GEC, Flores-Rios BL, Vázquez JPG, Ibarra-Esquer JE. Integration of different instructional strategies: ABP, Scrum framework and School environment company in the professional education of IT undergraduates [Integración de diferentes estrategias instruccionales: ABP, Scrum y empresa escolar en la formación profesional de estudiantes de TI]. RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao. 2023;2023(E61):98–112. https://dialnet.unirioja.es/servlet/articulo?codigo=10039950.
93. Andriyani Y. Knowledge Management and Reflective Practice in Daily Stand-Up and Retrospective Meetings. In: Baumeister H, Lichter H, Riebisch M, editors. Agile Processes in Software Engineering and Extreme Programming. Cham: Springer International Publishing; 2017. p. 285–291. https://doi.org/10.1007/978-3-319-57633-6_21.
94. Hundhausen C, Conrad P, Tariq A, Pugal S, Flores BZ. An Empirical Study of the Content and Quality of Sprint Retrospectives in Undergraduate Team Software Projects. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. ICSE-SEET '24. New York, NY, USA: Association for Computing Machinery; 2024. p. 104–114. https://doi.org/10.1145/3639474.3640074.
95. Jungclaus J, Korge G, Arndt P, Bauer A. Agile Sprintlearning – a new conceptual framework for learning in digitized work [Agiles Sprintlernen – ein Konzept für dezentrales betriebliches Lernen: Empirische Begründung und praktische Erfahrungen]. Gruppe Interaktion Organisation Zeitschrift fur Angewandte Organisationspsychologie. 2019;50(2):217–27. https://doi.org/10.1007/s11612-019-00468-y.

96. Bogdanova M, Parashkevova-Velikova E. Agile perspectives in higher education. Smart Innov Syst Technol. 2022;276:333–45. https://doi.org/10.1007/978-981-16-8866-9_28.
97. Israel M, Jeong G, Ray M, Lash T. Teaching Elementary Computer Science through Universal Design for Learning. In: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. SIGCSE '20. New York, NY, USA: Association for Computing Machinery; 2020. p. 1220–1226. Available from:https://doi.org/10.1145/3328778.3366823.
98. Jacob S, Montoya J, Prado Y, Coenraad M, Burke W, Warschauer M. Designing for All: Universal Design for Learning Meets Computational Thinking for Multilingual Students. In: Proceedings of the 2025 Conference on Research on Equitable and Sustained Participation in Engineering, Computing, and Technology. RESPECT 2025. New York, NY, USA: Association for Computing Machinery; 2025. p. 309–316. Available from: https://doi.org/10.1145/3704637.3734766.
99. Juvino De Araújo EC, Andrade WL, Souto Oliveira AL. Perceptions about Teaching Programming in the Neurodiverse Students' Context. In: 2024 IEEE Frontiers in Education Conference (FIE); 2024. p. 1–9. Available from: https://doi.org/10.1109/FIE61694.2024.10892904.
100. Beaux H, Karimi P, Pop O, Clark R. Guiding Empowerment Model: Liberating Neurodiversity in Online Higher Education. In: 2024 IEEE Frontiers in Education Conference (FIE); 2024. p. 1–9. Available from: https://doi.org/10.1109/FIE61694.2024.10893125.
101. Asbell-Clarke J, Robillard T, Edwards T, Bardar E, Weintrop D, Grover S, et al. Including Neurodiversity in Foundational and Applied Computational Thinking (INFACT). In: Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 2. SIGCSE 2022. New York, NY, USA: Association for Computing Machinery; 2022. p. 1076. Available from: https://doi.org/10.1145/3478432.3499044.
102. Patilima H. Neurodiversity and trauma in early childhood: implications for inclusive learning. South Afr J Childhood Educ. 2025;15(1):a1704. https://doi.org/10.4102/sajce.v15i1.1704.
103. Pozos RK, Severance S, Denner J, Tellez K. Exploring design principles in computational thinking instruction for multilingual learners. Teach Coll Rec. 2022;124(5):127–45. https://doi.org/10.1177/01614681221104043.
104. Sanderson NC, Kessel S, Chen W. What do faculty members know about universal design and digital accessibility? A qualitative study in computer science and engineering disciplines. Univ Access Inf Soc. 2022;21(2):351–65. https://doi.org/10.1007/s10209-022-00875-x.
105. Baghban Karimi O, Gao A, Lindner P, Toti G, Engineer R, Hur J, et al. Exploring Equity, Diversity, and Inclusion in Computer Science Undergraduate Curricula. In: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2. ITiCSE 2024. New York, NY, USA: Association for Computing Machinery; 2024. p. 765–766. Available from: https://doi.org/10.1145/3649405.3659531.
106. Bai X, Li M, Pei D, Li S, Ye D. Continuous delivery of personalized assessment and feedback in agile software engineering projects. In: Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training. ICSE-SEET '18. New York, NY, USA: Association for Computing Machinery; 2018. p. 58–67. Available from: https://doi.org/10.1145/3183377.3183387.
107. Alshammari MT. An investigation into ChatGPT-enhanced adaptive e-learning systems. TEM J. 2025;14(1):503–10. https://doi.org/10.18421/TEM141-45.
108. Hodgkinson BF, Whittenburg NE. WIP: AI-based Sentiment Analysis and Grader Enhancements. In: 2024 ASEE Annual Conference & Exposition. 10.18260/1-2–48287. Portland, Oregon: ASEE Conferences; 2024. https://peer.asee.org/48287.
109. Chun AHW. The Agile Teaching/Learning Methodology and Its e-Learning Platform. In: Liu W, Shi Y, Li Q, editors. Advances in Web-Based Learning – ICWL 2004. vol. 3143 of Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg; 2004. p. 11–18. https://doi.org/10.1007/978-3-540-27859-7_2.
110. Mahnic V, Hovelja T. Teaching user stories within the scope of a software engineering capstone course: Analysis of students' opinions. Int J Eng Educ. 2014;30(4):901–15 (https://dialnet.unirioja.es/servlet/articulo?codigo=7360943.).
111. Groothuijsen S, van den Beemt A, Remmers JC, van Meeuwen LW. AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. Comput Educ Artif Intell. 2024;7:100290. https://doi.org/10.1016/j.caeai.2024.100290.
112. Santos MFd. Reestruturação pedagógica de gestão de projeto de software [Master's thesis]. Instituto Superior de Engenharia de Coimbra (ISEC), Instituto Politécnico de Coimbra. Coimbra, Portugal; 2024. Available from: http://hdl.handle.net/10400.26/57556.
113. Figueiredo J, García-Peñalvo F. Teaching and Learning Strategies for Introductory Programming in University Courses. In: Ninth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'21). TEEM'21. New York, NY, USA: Association for Computing Machinery; 2021. p. 746–751. https://doi.org/10.1145/3486011.3486540.
114. Pires Ja, Gomes A, Borges AR, Correia FB. Predicting Higher Education Student's Aptitude to Learn to Program: A Systematic Literature Review. In: Proceedings of the International Conference on Computer Systems and Technologies 2024. CompSysTech '24. New York, NY, USA: Association for Computing Machinery; 2024. p. 242–248. https://doi.org/10.1145/3674912.3674949.
115. Allison J, Alam A, Gassmann L, Nelson G, Zidan K. Fostering the development of computer science graduate employability through agile projects. J Furth High Educ. 2024;48(4):417–35. https://doi.org/10.1080/0309877X.2024.2340642.
116. Abouelenein YAM, Ghazala AFA, Mahdy EMM, Khalaf MHR. The R5E pattern: can artificial intelligence enhance programming skills development? Educ Inf Technol. 2025. https://doi.org/10.1007/s10639-025-13616-3.
117. Thong CL, Atallah Z, Islam S, Lim W, Cherukuri AK. AI-powered Tools for Doctoral Supervision in Higher Education: A Systematic Review. J Inf Knowl Manag. 2025;24(2):2530001. https://doi.org/10.1142/S0219649225300013.
118. Katipoglu G, Utku S, Mijailovic I, Mekic E, Avdic D, Milic P. Action Research Approach to Analysis of Teaching of Blockchain Web 3.0 Application Based on MACH Architecture. Applied Sciences. 2024;14(23):11158. https://doi.org/10.3390/app142311158.
119. Zorzo SD, de Ponte L, Lucrédio D. Using Scrum to teach software engineering: A case study. In: 2013 IEEE Frontiers in Education Conference (FIE); 2013. p. 455–461. Available from: https://doi.org/10.1109/FIE.2013.6684866.
120. Pombo N, Cunha C. Agile Methodologies in Education: The Common Ground of Personalized Learning and Project-Based Learning. In: Song YT, Kang M, Rhee J, editors. 2025 IEEE/ACIS 23rd International Conference on Software Engineering Research, Management and Applications, SERA 2025 - Proceedings. Institute of Electrical and Electronics Engineers Inc.; 2025. p. 89–96. Available from: https://doi.org/10.1109/SERA65747.2025.11154615.

121. Almeida D, de Souza MC, da Silva BP, Paes RL, Soliman M, Costa SB, et al. Agile Methodologies in the Educational Context: A Systematic Literature Review. In: Mesquita D, Villas-Boas V, Lima RM, editors. International Symposium on Project Approaches in Engineering Education. vol. 15. University of Minho; 2025. p. 351–358. Available from: https://doi.org/10.5281/zenodo.15878750.
122. Geetha LS, Baker El-Ebiary YA, Rao BS, Rautrao RR, Rao TSM, Ramesh JVN, et al. Challenges and Solutions in Agile Software Development: A Managerial Perspective on Implementation Practices. Int J Adv Comput Sci Appl. 2025;16(3):748–58. https://doi.org/10.14569/IJACSA.2025.0160374.
123. Zapater M, Malagón P, De Goyeneche JM, Moya JM. Project-Based Learning and Agile Methodologies in Electronic Courses: Effect of Student Population and Open Issues. Electronics. 2013;17(2):82–8. https://doi.org/10.7251/ELS1317082Z.
124. Abdulrachman LM, Llantos OE. Asynchronous Inter-Class Project Collaboration Adaptation Method to Agile Practices in University Context. Procedia Computer Science. 2024;251:57–65. https://doi.org/10.1016/j.procs.2024.11.084.
125. Bhuttah TM, Xusheng Q, Abid MN, Sharma S. Enhancing student critical thinking and learning outcomes through innovative pedagogical approaches in higher education: the mediating role of inclusive leadership. Sci Rep. 2024;14(1):24362. https://doi.org/10.1038/s41598-024-75379-0.
126. Klochko OV, Gurevych RS, Fedorets VM, Klochko VI, Konoshevskyi OL, Kovtoniuk MM. System-forming aspects of computer science and mathematics teachers' readiness to develop and use computer didactic games: a comprehensive TPACK-GPCK framework analysis. In: Semerikov SO, Striuk AM, Pinchuk OP, Vakaliuk TA, editors. Proceedings of the 8th International Workshop on Augmented Reality in Education (AREdu 2025) co-located with 6th International Conference on History, Theory and Methodology of Learning (ICHTML 2025), Kryvyi Rih, Ukraine, May 13, 2025. vol. 4060 of CEUR Workshop Proceedings. CEUR-WS.org; 2025. p. 332–367. Available from: https://ceur-ws.org/Vol-4060/paper14.pdf.
127. Balawi S, Khalaf K, Hitt GW. Leveraging Pedagogical Innovations for Science, Technology, Engineering, and Mathematics (STEM) Education in the Middle East Context. In: Abdulwahed M, Hasna MO, Froyd JE, editors. Advances in Engineering Education in the Middle East and North Africa: Current Status, and Future Insights. Cham: Springer International Publishing; 2016. p. 59–115. Available from: https://doi.org/10.1007/978-3-319-15323-0_4.
128. Paulick J, Quinn AM, Blain CD. Developing culturally responsive teaching self-efficacy through engaged, asset-based teacher preparation. Teach Educ. 2024;35(1):61–81. https://doi.org/10.1080/10476210.2023.2215166.
129. Astrachan O, Osborne RB, Lee I, Beth B, Gray J. Diverse learners, diverse courses, diverse projects: learning from challenges in new directions. In: Proceedings of the 45th ACM Technical Symposium on Computer Science Education. SIGCSE '14. New York, NY, USA: Association for Computing Machinery; 2014. p. 177–178. https://doi.org/10.1145/2538862.2538991.

Publisher's Note