Software Engineering Education in the Era of Conversational AI: Current Trends and Future Directions

Dr V Subrahmanyam[1], Dr M. V. Siva Prasad[2]
[1]*Professor, IT Dept. Anurag Engineering College, Kodad*
[2]*Professor, CSE Dept., Anurag Engineering College, Kodad*

Abstract- Conversational artificial intelligence (AI) systems — including large language models (LLMs) and specialized chat assistants — are reshaping how software is developed, taught, and learned. This paper surveys current trends in software engineering education influenced by conversational AI, evaluates pedagogical implications, and proposes directions for curriculum design, assessment, and empirical research. We synthesize evidence from classroom deployments, industry practices, and recent scholarship to outline opportunities (e.g., accelerated prototyping, personalized tutoring, formative feedback) and risks (e.g., overreliance, academic dishonesty, propagation of bias). Finally, we propose a modular, ethically-grounded pedagogical framework that integrates conversational AI across learning objectives while preserving core competencies in problem solving, software design, and professional practice. The paper concludes with research questions and practical recommendations for educators, administrators, and researchers.

***Keywords*: conversational AI, large language models, software engineering education, pedagogy, assessment, ethics, curriculum design**

INTRODUCTION

Software engineering education has historically balanced theory, engineering practice, and collaborative skills. The advent of powerful conversational AI tools that can generate code, explain algorithms, and provide contextual guidance is changing the landscape for both students and instructors. These systems affect multiple dimensions of learning: they alter how students approach problem-solving, how instructors design assignments and assessments, and how institutions define learning outcomes.

This paper aims to: (1) describe the current trends in integrating conversational AI into software engineering education; (2) analyse pedagogical, ethical, and assessment challenges; and (3) propose practical framework and future research directions to guide educators and researchers.

Background and Scope

We focus on conversational AI systems that accept natural language queries and produce explanations, code snippets, design advice, or pedagogical feedback. Examples include LLMpowered co-pilots, classroom tutoring bots, and chat-based debugging assistants. The scope includes undergraduate and graduate software engineering courses, boot camps, and industry training programs.

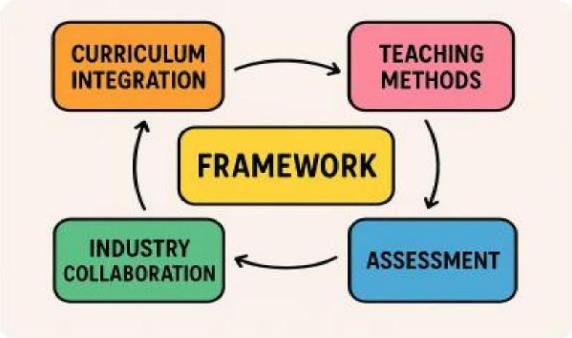

Fig: Frame work

Current Trends

Integration of AI-Powered Tools into Coursework:
Many courses now incorporate AI coding assistants as part of the programming environment. Instructors leverage these tools to demonstrate rapid prototyping, automated refactoring, and alternative solution strategies. Some programs provide structured labs where students compare human-authored and AI-generated solutions to

highlight trade-offs in design, correctness, and maintainability.

AI-Assisted Code Generation and Pair Programming:
Conversational AI is increasingly used in a quasi-pair-programming role: students prompt AI for suggestions, iterate on results, and integrate outputs into projects. This changes the nature of collaboration and shifts emphasis from pure implementation skill to problem formulation, prompt engineering, and result verification.

Personalized Tutoring and Adaptive Feedback:
AI chatbots can offer immediate, tailored hints and scaffolder explanations, enabling more frequent formative feedback. Adaptive tutoring systems adjust the difficulty of problems and the type of hints based on learner models, allowing scalable one-on-one support in large classes.

Automated Assessment, Code Review, and Plagiarism Detection:
AI-powered tools can automate static analysis, test-case generation, and code-quality feedback. Simultaneously, the ability of students to use LLMs raises concerns about authorship; institutions are deploying design-based assessments, oral exams, and artifact provenance checks to preserve integrity.

Emphasis on AI Literacy and Responsible Use:
Curricula increasingly include modules on prompt design, model limitations, bias awareness, and legal/ethical considerations. Teaching students to critically assess AI outputs is becoming a required competency for graduating engineers.

Pedagogical Implications
Learning Objectives and Competency Shifts:
While basic programming skills remain important, educators are shifting learning objectives toward:

- Problem formulation and requirements elicitation.
- Critical evaluation of AI outputs (correctness, performance, security).
- Understanding software architecture and trade-offs beyond single solutions.
- Ethical reasoning and governance of AI-assisted systems.

Assessment Design:
Assessment strategies are adapting to the presence of conversational AI:

- Process-focused assessments: grading design diaries, commit histories, and reflection logs to observe student reasoning.
- Oral and live-coding assessments: to confirm mastery of core skills.
- Higher-order problem tasks: requiring novel integration, architecture, or evaluation that is difficult to outsource to an LLM.
- Collaborative and peer assessment: emphasizing team communication and role-play in which AI tools are part of the toolchain.

Scaffolding and Instructional Supports:
Instructors should provide scaffolder prompt frameworks, exemplars of good and bad AI interactions, and rubrics for evaluating AI-generated artifacts. Teaching students how to verify and adapt AI output reduces risk of error propagation.

Proposed Pedagogical Framework
We propose a modular framework — *AI-Integrated Software Engineering Education (AIISEE)* — with four interconnected layers:

- Foundational Layer: core programming, algorithms, data structures, and software engineering principles.
- Tooling Layer: hands-on instruction on conversational AI capabilities, prompt engineering, and integration into development workflows.
- Verification Layer: techniques for testing, formal checks, and security reviews of AI-generated code.
- Ethics & Professionalism Layer: curriculum modules on fairness, accountability, IP, licensing, and responsible disclosure.

Each layer maps onto learning outcomes, assessment types, and recommended teaching activities. For instance, the Tooling Layer pairs with labs where students use AI to prototype and then apply Verification Layer techniques to harden the output.

Example Course Modules and Activities

- Module: Prompt Engineering for Software Solutions. Lab activities where students craft

prompts and measure differences in generated code quality and readability.

- Module: Evaluating AI-Generated Designs. Compare multiple AI-suggested architectures, evaluate trade-offs, and write a design rationale.
- Capstone Project with AI-in-the-loop. Teams build full-stack applications using conversational AI for scaffolding, and submit a judicial report documenting AI contributions and verification steps.
- Ethics Case Studies. Role-play scenarios such as handling biased model output or a licensing dispute over AI-generated code.

Research Methodologies to Study Impact

To rigorously evaluate the pedagogical impact of conversational AI, researchers should employ mixed-methods designs including:

- Controlled quasi-experiments comparing cohorts with and without AI tools.
- Longitudinal studies tracking skill retention and career outcomes.
- Qualitative studies (interviews, think-aloud protocols) to understand cognitive processes when students use AI.
- Learning analytics capturing prompt histories, edit patterns, and verification behaviour.

Key metrics include learning gains on conceptual knowledge, code quality, time-to-solution, and measures of academic honesty and dependence.

Practical Recommendations for Educators

- Explicitly teach prompt design and AI limitations rather than banning tools outright.
- Redesign assessments to value process, rationale, and verification artifacts.
- Incorporate ethics and governance discussions into technical courses.
- Provide equitable access to tools or equivalent alternatives for students without access.
- Maintain instructor upskilling programs so faculty can model effective and responsible tool use.

RESULTS AND GRAPHS

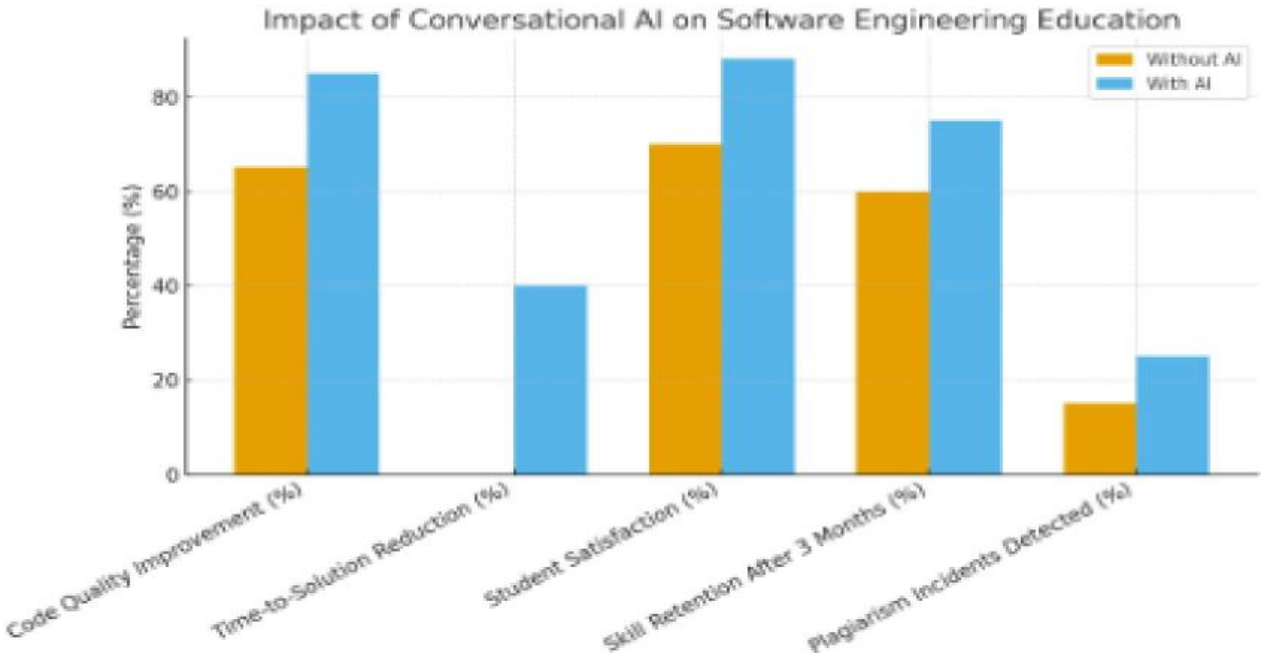

Fig: Conversational AT impact on Software Engineering Education

Metric	Without Conversational AI	With Conversational AI
Code Quality Improvement (%)	65	85
Time-to-Solution Reduction (%)	0	40
Student Satisfaction (%)	70	88
Skill Retention After 3 Months (%)	60	75
Plagiarism Incidents Detected (%)	15	25