The Future of Learning: Teaching Software Development in the Age of AI

Robert Pucher[1], Robert Mischak[2]

University of Applied Sciences, Austria[1,2]

Abstract

Artificial Intelligence (AI) is increasingly integrated into every phase of the software engineering lifecycle, from requirements analysis to maintenance, bringing transformative impacts on development processes and educational practices alike. AI tools such as GitHub Copilot, machine learning-driven testing systems, and natural language processing applications now assist with coding, testing, design, and planning tasks. This evolution necessitates a rethinking of software engineering education to prepare students not only to use AI tools effectively but also to understand their limitations and implications.

However, AI's ability to solve complex problems instantly undermines traditional assessment formats. If students rely on AI to complete assignments, their actual learning remains unclear. Educators must therefore shift toward process-oriented evaluation, including oral exams, project audits, and reflective AI-assisted tasks. Curricula should incorporate AI as both a learning tool and an object of critical inquiry.

This paper presents the ISTQB Foundation Level – Practical Tester as a model for AI-integrated education. Structured into 15 chapters with clear learning outcomes, the program uses AI to provide immediate feedback on student responses, increasing motivation and learning efficiency. Final assessments are AI-pre-evaluated but human-graded, combining scalability with pedagogical integrity.

In conclusion, banning AI in education is not viable; instead, its deliberate integration offers powerful opportunities for deeper learning and skill development. To fully leverage AI's educational potential, institutions must invest in AI literacy, infrastructure, and faculty training, ensuring that future software engineers are equipped to thrive in a technology-driven world.

Keywords: *Artificial Intelligence in Education, Software Engineering, AI-Assisted Learning, Curriculum Development, Assessment Methods, ISTQB Practical Tester*

1. Introduction - Artificial Intelligence in Software Engineering

Artificial Intelligence (AI) is now routinely applied across many phases of the software engineering lifecycle to enhance efficiency, quality, and automation. Its integration reshapes how software is designed, developed, tested, and maintained.

Requirements Engineering and Planning: Natural Language Processing (NLP) techniques support automated requirements extraction and interpretation, improving consistency and traceability.

Software Design and Architecture: AI-driven tools assist in design decisions, architecture optimization, and the identification of reusable patterns, thereby enhancing system quality and maintainability.

Coding and Code Generation: Tools like GitHub Copilot, powered by large language models, facilitate intelligent code completion and automatic code synthesis, reducing developer workload.

Software Testing: Machine learning supports automated test generation, prioritization, and anomaly detection, leading to more efficient and robust testing processes.

Debugging and Maintenance: AI techniques such as log analysis and predictive modeling enable automated fault localization and proactive system maintenance.

Project Management: AI aids project planning through data-driven prediction of risks, timelines, and resource needs, leading to better-informed management decisions.

Given the widespread and growing use of AI across all stages of software engineering—from requirements analysis to maintenance, it is essential that education in this field evolves accordingly. Developers must understand both the capabilities and limitations of AI tools to use them effectively and responsibly. Therefore, education should at least include similar elements, or ideally incorporate experimental methods, so that students gain hands-on experience with real-world applications. Integrating AI-related topics into software engineering curricula ensures that future professionals are equipped with the skills necessary to thrive in an industry increasingly shaped by intelligent automation and data-driven decision-making.

These topics already are being extensively discussed in the literature; see, for example, references [1-8] for an overview.

2. Challenges for Teaching

From the perspective of educators, the widespread availability of AI presents significant challenges to traditional teaching and assessment methods—particularly at the undergraduate level, but increasingly in master's programs as well. At these stages, students are expected to learn and internalize foundational principles of software engineering: algorithm design, data structures, programming paradigms, software architecture, testing strategies, and requirements engineering.

2.1 When AI Solves the Task, Learning Falls Behind

However, today's AI systems—especially large language models—already "know" these principles and can apply them flawlessly. Students can now generate complete and correct solutions to typical homework or lab assignments using AI tools within seconds. As a result, submitted work often reflects the capabilities of the AI more than the student's own understanding.

This renders conventional assessment methods ineffective. If an assignment produced with AI consistently achieves full marks, grading becomes meaningless. More critically, instructors are left to wonder: What have the students actually learned? When AI handles the thinking, students risk bypassing essential cognitive and problem-solving processes. They may complete coursework without gaining the conceptual depth needed to become competent software engineers.

This situation creates a fundamental tension in teaching. On the one hand, AI can be a powerful learning aid. On the other, it undermines the purpose of exercises designed to develop skill through practice. Educators now face the urgent task of rethinking instructional design, shifting from outcome-based assessment to process-oriented evaluation, and creating learning environments where the use of AI fosters, rather than replaces, understanding.

2.2 Rethinking Curriculum and Assessment in the Age of AI

It is therefore essential to adapt *all* teaching content to reflect the presence and potential of AI—not to ignore or ban it, but to integrate it deliberately into the learning process. Students should be encouraged to use AI as a tool to deepen their understanding, explore alternative solutions, and reflect on the strengths and limitations of algorithmic assistance. In doing so, they learn not only subject matter, but also critical AI literacy—an increasingly vital competence in software engineering and beyond.

However, this alone is not sufficient. The *assessment process* must also be fundamentally redesigned to ensure that the intended skills are reliably evaluated. If traditional assignments no longer serve as valid indicators of individual competence, alternative approaches are required—such as oral exams, live problem-solving, peer-reviewed projects, process documentation, or AI-assisted design tasks with critical reflection components.

For decades, educators have observed that students adjust their learning behaviors in direct response to assessment requirements. This phenomenon is encapsulated in the principle of constructive alignment, which posits that when learning outcomes, teaching activities, and assessments are coherently aligned, students are more likely to engage in behaviors that lead to the desired learning outcomes. Biggs and Tang emphasize that aligning assessments with intended learning outcomes ensures that students' efforts are directed toward achieving those outcomes, thereby enhancing the overall effectiveness of the educational process [9]. In the context of AI-integrated education, this alignment becomes even more critical, as it guides students to use AI tools to deepen their understanding rather than circumvent the learning process [10].

When students understand how skills will be assessed, and that these assessments cannot be completed by AI alone, they adapt accordingly. They shift their learning strategies to meet the demands of the evaluation. In this way, well-designed assessments become the most powerful lever to guide meaningful learning. Ultimately, the goal is to ensure that students not only *use* AI, but also develop conceptual, analytical, and creative abilities that distinguish human expertise in a technology-driven world.

3. Important Aspects of Curriculum Development

Depending on the phase in the software development process, AI tools can have a greater or lesser impact on teaching. The impact is particularly significant in coding and testing (Table 1). It is therefore necessary to consider this in the essential framework of teaching, which is ideally defined in the curriculum description. Care should be taken to ensure that the specifications are not too narrow, to still leave sufficient scope for instructors to individually design their teaching.

Phase of Software Engineering	**Influence of AI Tools on Teaching**	**Examples of AI-Integration**
Requirements Engineering and Planning	Low	Natural Language Processing (e.g., requirement analysis, user story validation)
Software Design and Architecture	Medium	AI-assisted design suggestions, pattern recognition
Coding and Code Generation	High	GitHub Copilot, code completion, syntax correction
Software-Testing	High	Machine Learning for test case generation, test optimization
Debugging and Maintenance	Medium	AI-supported log analysis, anomaly detection, predictive maintenance
Project Management	Low	Data-driven effort estimation, risk prediction

Table 1. Influence of AI-tools on teaching

3.1 Teaching Methods

It is essential for the teaching and learning process to enable students and teachers to experience the possibilities of AI directly in class. It can be assumed that computer science students themselves bring experiences, some of which are unknown to the teachers. Therefore, support from AI tools for specific questions should, if possible, take place directly in class and be discussed sufficiently. One possible scenario is to solve a question posed in the lecture, first without and then with AI support, and then compare or evaluate the results and the required effort and skills. Inverted classroom methods could be used here. In any case, this would make teaching more enriching for both parties.

Exercise units or project-based courses should involve strong instructor participation and intensive team interaction. The extent to which this is possible in purely online courses seems questionable. The team that has found a solution presents it to the others and describes the process of finding the solution. Similar considerations on problem-based learning can be found, for example, in [11]. After all, it's not just the result that counts, but also the path to it.

AI-supported teaching places considerable demands on the technical infrastructure of the teaching environment. Suitable mobile devices (laptops, etc.) for students and presentation options via projectors, good Wi-Fi connections, powerful computers on the network, and licenses or at least free access to various AI tools must be ensured. The university may need stricter specifications regarding the required hardware and software to ensure effective teaching.

3.2 Examination Methods

Many of the assessment methods used in the past are no longer applicable due to the availability of AI tools, as examiners may no longer be able to recognize and therefore assess the students' own contribution. Examinations must also be conducted fairly and under equal conditions for students; students who have access to "better" tools should not be given higher marks because of the tools themselves.

It turns out that the role of examiners is becoming significantly more demanding, because not only the result but also the process must be assessed. Simply correcting and assessing program code, architectural concepts, data models etc., as homework or seminar papers do not address this problem. Students must be able to demonstrate their progress and explain the results to assess their results. This also trains their skills in the direction of *explainable and trustworthy AI*.

The assessment of project work thus becomes an audit. Instead of bachelor's or master's theses, much shorter but more compact papers should be submitted in a publishable format, which undergo a peer-review process. To put it bluntly, this development is the "rebirth of the oral examination."

3.3 Competencies to be Acquired

When formulating learning objectives for students, the competencies expected in the software development process, even when they rely on AI tools, must be considered. It is undisputed that the mere possession and use of AI tools does not constitute sufficient competence for software developers. To compare it with interpreting students: simply holding a dictionary and looking up vocabulary does not mean that one can work as an interpreter.

The competencies required for the meaningful application of AI tools essentially fall under methodological competence. However, a sufficient understanding of AI as a technology about its limitations, dangers, etc. (AI literacy) in the context of software development is also fundamental. Based on Bloom's taxonomy, depending on their qualification level, students should be familiar with, understand, and able to apply AI tools, or even be able to analyze complex issues and create new ones. Academic training also includes acquiring competencies regarding ethical aspects of AI [12]. These considerations should also be appropriately reflected in the European Qualifications Framework (EQF).

4. Example; ISTQB – Software Testing

Software testing plays a critical role in ensuring the quality, functionality, and reliability of modern software systems. As software becomes increasingly embedded in every aspect of society, from healthcare and finance to transportation and education, the need for robust testing methodologies becomes more urgent. Poorly tested software can lead to significant failures, financial losses, security breaches, and even threats to human life. Consequently, software testing is not just a technical task but a strategic activity that safeguards user trust and business continuity.

4.1 The Role of ISTQB in Standardizing Software Testing

To address the complexity and ensure a common understanding of testing principles, the International Software Testing Qualifications Board (ISTQB) was established in 2002. ISTQB is a globally recognized organization that defines standards for software testing certification. Its mission is to advance the testing profession by promoting a body of knowledge that is vendor-neutral, up-to-date, and aligned with industry best practices.

ISTQB offers a tiered certification scheme with multiple levels and specializations. The core levels are:

- Foundation Level: Covers basic testing concepts, life cycles, techniques, and tools.
- Advanced Level: Includes Test Analyst, Technical Test Analyst, and Test Manager modules.
- Expert Level: Focuses on strategic topics such as test automation, test process improvement, and test management.

These certifications are widely adopted by companies and institutions as benchmarks for hiring, training, and evaluating software testers.

4.2 AI-Supported Learning in Software Testing: The ISTQB Foundation Level – Practical Tester

This certification addresses the need for hands-on, practical skills in addition to theoretical knowledge. While the traditional Foundation Level exam focuses on conceptual understanding, the Practical Tester extension is designed to assess and develop applied competencies that reflect real-world testing scenarios.

The AI-supported training in the ISTQB Foundation Level – Practical Tester context involves integrating artificial intelligence tools into the practical learning and assessment process to enhance student engagement, individualize instruction, and mirror real-world testing scenarios.

4.3 Training Process

The ISTQB Foundation Level – Practical Tester syllabus is structured into 15 clearly defined chapters, each focusing on a specific aspect of software testing. Every chapter includes well-formulated learning outcomes that describe the exact knowledge and competencies a student should acquire. These learning outcomes form the foundation of the training and are directly linked to the questions posed to the students during their learning journey.

For each learning outcome, students are asked to provide answers in plain text, typically in natural language. These responses are then analyzed by an AI-driven evaluation system. The AI checks the content for correctness, relevance, completeness, and clarity. Based on this analysis, the system provides personalized feedback, highlighting what the student has done well and offering constructive suggestions for improvement where necessary.

This feedback mechanism has proven to be both effective and motivating. Students receive immediate, detailed responses to their submissions, which helps to reinforce understanding and promote deeper engagement with the material. Unlike traditional classroom settings, where feedback might take days or weeks, the AI responds within seconds, enabling a much more dynamic and interactive learning experience.

One of the key benefits of this format is its flexibility. Students can learn at their own pace and on their own schedule. Whether during a lunch break, in the evening, or on weekends, they can engage with the material whenever it suits them best. This self-paced structure is particularly appealing to professionals who wish to acquire new skills alongside their regular job responsibilities.

Moreover, the AI can answer students' follow-up questions in real time. If a student does not understand a particular concept or needs clarification on feedback received, the AI tutor is available 24/7 to provide further explanations or examples. This on-demand support drastically reduces learning barriers and increases satisfaction and motivation.

Companies also benefit significantly from this AI-supported format. Traditional training often requires scheduling, travel, and the presence of trainers, which can be costly and inflexible. With the AI-driven system, organizations no longer need to allocate time and resources for dedicated training sessions. Employees can train autonomously, and managers can monitor progress and results through dashboards and analytics tools.

4.4 The exam

All questions posed during the AI-supported training, as well as similar or derivative questions, may appear in the final examination. The exam is conducted under supervision, within a clearly defined time limit, to ensure fairness and integrity. During the exam, students respond to open-ended prompts in natural language, as they did during training. These responses are pre-evaluated by the AI system based on criteria such as correctness, completeness, and relevance.

However, the final assessment is not left solely to AI. A qualified examiner reviews the AI-generated evaluation, verifying its accuracy and making corrections if necessary. This human oversight ensures that context, nuance, and individual expression are adequately considered. Ultimately, the final grade is determined exclusively by the human examiner. This hybrid model combines the efficiency of automated evaluation with the critical judgment of experienced educators, ensuring both scalability and academic rigor in the assessment process.

5. Conclusion

Banning AI tools in the classroom is not only impractical, but also pedagogically counterproductive. This is especially true for software engineering education, where AI systems like large language models can already produce correct and sophisticated solutions to typical student assignments. As shown throughout this paper, such developments challenge traditional forms of teaching and assessment. To ensure meaningful learning, educators must fundamentally rethink curricula and examination formats. This includes reducing reliance on multiple-choice tests and increasing the use of oral exams, live coding, and process-focused assessments.

Our analysis demonstrates that when AI completes the task, learning can fall behind. However, when used strategically, AI becomes a powerful educational ally. The integration of AI into the teaching process, especially in coding and software testing, can support individualized feedback, real-time guidance, and flexible learning environments. The ISTQB Foundation Level – Practical Tester serves as a compelling case study in this regard: its AI-supported format provides immediate feedback on natural language responses, enabling students to engage deeply with defined learning outcomes across 15 structured chapters. This model not only improves motivation but also enhances accessibility and scalability.

Companies increasingly value such AI-driven formats, as they eliminate the need for dedicated training sessions while ensuring competence acquisition through automated, yet personalized, learning paths. To make this transition sustainable, teachers must adapt their roles, develop new competencies, and receive adequate institutional support.

Ultimately, AI literacy is not optional. It is a foundational skill for students, educators, and society. In the age of intelligent tools, the ability to use, interpret, and critically assess AI becomes a key pillar of professional and civic education. As this paper shows, embracing AI in education is not just about keeping pace with technology, it is about redefining how we teach, assess, and learn in a digital world.

REFERENCES

[1] P. Harman, M. Liaaen, and J. Clark, "Artificial intelligence in software engineering: Where are we now?" *IEEE Software*, vol. 36, no. 5, pp. 92–95, 2019.
[2] M. Sabetzadeh and A. Arora, "Practical Guidelines for the Selection and Evaluation of Natural Language Processing Techniques in Requirements Engineering," arXiv preprint arXiv:2401.01508, Jan. 2024. [Online]. Available: https://arxiv.org/abs/2401.01508.
[3] S. Ali, M. Yue, M. A. Babar, and A. Jaatun, "A systematic review of AI-supported software architecture," in *Proc. IEEE Int. Conf. Software Architecture (ICSA)*, 2021, pp. 141–151.
[4] H. Chen, M. Zhou, and Z. Zhang, "Evaluating the performance of GitHub Copilot's code generation capabilities," *arXiv preprint arXiv:2205.15372*, 2022.
[5] A. Panichella, A. Arcuri, and A. Zaidman, "The role of machine learning in software testing: survey, landscape and future directions," in *Proc. IEEE 26th Int. Conf. Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 385–395.

[6] Z. Li, Y. Zhou, B. Luo, and X. Ma, "DeepFL: Integrating multiple fault localization techniques using deep learning," in *Proc. IEEE Int. Conf. Software Maintenance and Evolution (ICSME)*, 2019, pp. 541–545.
[7] C. Xu, R. Sathya, A. Alshahrani, R. Alotaibi, and A. Ibrahim, "An Effective Software Risk Prediction Management Analysis of Data Using Machine Learning and Data Mining Method," arXiv preprint arXiv:2406.09463, Jun. 2024. [Online]. Available: https://arxiv.org/abs/2406.09463M.
[8] Vierhauser, I. Groher, T. Antensteiner, and C. Sauerwein, "Towards Integrating Emerging AI Applications in SE Education," *arXiv preprint arXiv:2405.18062*, 2024. [Online]. Available: https://arxiv.org/abs/2405.18062
[9] J. B. Biggs, Teaching for Quality Learning at University: What the Student Does, 2nd ed., Buckingham, UK: Open University Press, 2003.
[10] S. Druga, "Coding Isn't Dead, but How It's Taught Needs to Change," Business Insider, May 8, 2025. [Online]. Available: https://www.businessinsider.com/google-deepmind-research-scientist-coding-not-dead-ai-education-2025-5
[11] Ondrusch, Nicole; Quandt, Veronica; Majunke, Jennifer; Sperrfechter, Claudia (2024): KI-Lehre fur Informatik-Student*innen in einem sich inhaltlich und methodisch verändernden Umfeld. Proceedings of DELFI Workshops 2024. DOI: 10.18420/delfi2024-ws-25. Gesellschaft für Informatik e.V. Available: https://dl.gi.de/items/bc21ef78-e11f-45f9-8af6-cde7cbd40419
[12] Krumme, Julia; Kóvacs, László; Teynor, Alexandra (2023): Ethik in der Ausbildung für Software-Entwickler:innen. SEUH 2023. DOI: 10.18420/seuh2023_02. Gesellschaft für Informatik, Bonn. ISSN: 1617-5468. ISBN: 978-3-88579-727-2 Available: https://dl.gi.de/items/9a8b3c0b-b0e4-49a5-bd50-efb2dd0c34eb