Integrating Generative AI in Software Engineering Education: Practical Strategies

Yishu Li
Dept of Computer Science
City University of Hong Kong
Hong Kong, China
yishuli5-c@my.cityu.edu.hk

Jacky Keung
Dept of Computer Science
City University of Hong Kong
Hong Kong, China
Jacky.Keung@cityu.edu.hk

Xiaoxue Ma
Dept of Computer Science
City University of Hong Kong
Hong Kong, China
xiaoxuema3-c@my.cityu.edu.hk

2024 International Symposium on Educational Technology (ISET) | 979-8-3503-6141-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISET61814.2024.00019

***Abstract*—The transformative influence of generative artificial intelligence (AI), notably large language models (LLMs), has significantly reshaped the software engineering (SE) landscape, impacting various aspects of software development within industry and academia. The imperative to integrate generative AI into educational programs arises from the necessity to furnish graduates with contemporary methodologies that enhance software quality and streamline development processes. Nevertheless, a research gap exists concerning the systematic integration of established SE education guidelines with specific course contexts to strengthen SE education through incorporating generative AI. In response to this gap, our study presents a vision for integrating generative AI into SE education, with a particular emphasis on practical integration strategies aimed at endowing students with essential competencies tailored for contemporary software development. Aligning our vision with the knowledge domains within SE education, we delineate its application across specific areas such as code generation, auto test case completion, and others. The overall objective of these proposed initiatives is to furnish students in SE with an updated and immersive learning experience, thereby addressing the evolving demands of the field.**

Keywords—software engineering, education, generative AI, large language models, code generation, auto test case completion

I. Introduction

The rapid advancement of generative artificial intelligence (AI) has disrupted various industries, including Software Engineering (SE). These advancements of generative AI, specifically large language models (LLMs), have significantly reshaped the software development landscape [1], revolutionizing areas such as software design [2], coding [3, 4], testing [5], debugging [6], and maintenance. Consequently, it is urgent to incorporate generative AI into education and training programs. This is crucial to provide graduates with contemporary techniques that improve software quality, minimize errors, and streamline development processes.

Software development is a continuously evolving field, adapting to new technologies and practices. To keep pace with these advancements and provide students with a comprehensive learning experience, integrating generative AI tools into a software development course can be highly beneficial. Without a doubt, it is imperative for our students to understand how to leverage generative AI tools, as they play a pivotal role in automating and enhancing various aspects of software development tools and processes.

While existing studies, such as [7, 8], have explored the potential integration of generative AI in SE education, drawing insights from both industry and academia, there remains a gap in research. Specifically, there is a lack of comprehensive investigations that systematically combine established SE education guidelines with specific course contexts to clarify how SE education can be enriched through the integration of generative AI.

In support of high-quality SE education, the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) jointly provide guidance through the "Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering" (SE2014, for short) [9]. This framework delineates ten distinct knowledge areas, including software design, software process, and software quality, which outline specific disciplines within SE deemed essential for graduates to possess proficiency upon completion of their studies.

In this paper, our objective is to present a vision discussing the conceptual framework for integrating generative AI into SE education. This involves synthesizing analysis by incorporating SE2014 guidelines and considering contextual relevance within our course structure. The proposed framework emphasizes the practical integration of generative AI into SE education, with a focus on empowering students with the skills and knowledge required to produce high-quality software code, implement comprehensive evaluation and testing procedures, and leverage process automation to meet the demands of today's fiercely competitive software development environment. The overall goal of these deliverables outlined in our vision is to provide an updated and immersive learning experience for students in SE.

The rest of the paper is organized as follows. Section II presents the overview of SE education in terms of knowledge area. Section III combines our course context to analyze the SE2014 knowledge areas point by point for potential integration, and Section IV offers a discussion of our visions. Finally, Section V concludes our study and discusses future work.

II. Background and Related Work

A. Software Engineering Education

Software Engineering (SE) education endeavors to integrate theoretical concepts with practical application, facilitating the acquisition of comprehensive understanding and proficiency in foundational principles alongside problem-solving capabilities for real-world scenarios [10]. SE2014 [9] provides guidelines to academic institutions and accreditation agencies regarding the essential components of undergraduate SE courses. These guidelines delineate a core body of knowledge, termed Software Engineering Education Knowledge, outlining the requisite competencies for SE graduates. SE2014 organizes this knowledge into ten distinct "knowledge areas", each representing key subdisciplines within SE recognized as integral to a graduate's proficiency [9].

979-8-3503-6141-4/24/$31.00 ©2024 IEEE
DOI 10.1109/ISET61814.2024.00019

Recent pedagogical shifts, which prioritize student empowerment, engagement, and motivation, have supplanted traditional approaches to delivering SE instruction [10]. This shift has facilitated the integration of technology within the classroom setting. Ouhbi and Pombo [10] contend that complementary technologies should be judiciously employed to construct an efficient and advantageous learning environment capable of augmenting students' proficiency in both soft and hard skills.

Garousi, et al. [11] conducted a systematic literature review on the gap between SE education and industrial requisites. Through a classification process, topics were assessed based on their significance, and gaps within the pertinent literature were identified. Some topics were designated as having both high importance and notable gaps, indicative of potential inadequacies within prevailing curricula.

Driven by the aim to foster immersive learning environments and enhance students' proficiency to align with industry standards, we advocate for the development of generative AI tools tailored for SE education. In this paper, we employ the knowledge areas proposed in SE2014 as a framework for identifying what and how generative AI techniques can be integrated into the SE education domains.

B. Generative AI

Recently released generative AI models, such as LLMs, have demonstrated exceptional proficiency across a spectrum of tasks encompassing language translation, summarization, and question-answering. LLMs undergo unsupervised training, leveraging billions of openly accessible text tokens to attain extensive language modeling capabilities. The incorporation of diverse data sources during LLM training enhances their cross-domain knowledge, thereby fostering robust generalization for associated downstream tasks [6].

Daun and Brings [12] investigated the viability of employing generative AI models such as ChatGPT within SE education. The investigation centered on ChatGPT's utility in producing solutions to SE inquiries and tackling SE assignments, encompassing coding exercises. The authors asserted that while ChatGPT poses potential obstacles, it concurrently offers educators significant avenues to augment SE curricula.

Bull and Kharrufa [7] proposed the vision for incorporating generative AI software development tools and methodologies into programming education, akin to instructing individuals on the utilization of conventional tools within their repertoire. They conducted exploratory interviews with industry experts to ascertain prevailing practices and challenges, integrating these insights into their envisioned future of software development education. Consequently, they formulated pedagogical recommendations advising educational institutions to collaborate with domain experts to implement scaffolding and fading techniques in instructional examples.

Petrovska, et al. [8] also investigated how generative AI can be integrated into software development education. They illustrate examples of both formative and summative assessments, delving into various facets of ChatGPT, including its coding capabilities, its ability to construct arguments, and the ethical issues of using ChatGPT and similar tools in education and the workplace. The authors proposed a methodological approach involving the strategic incorporation of generative AI tools, particularly through the development of assessments that prompt learners to critically assess AI-generated outputs. This approach aims to enhance learners' comprehension of the subject matter without undermining their educational integrity by allowing AI tools to complete assignments.

The findings from these studies suggest that the integration of generative AI into SE education is an emerging trend. Drawing from the insights gleaned and our own experiences in SE education, we have synthesized the SE2014 guidelines to envision the incorporation of generative AI across various knowledge domains within our SE course.

III. Analysis

In this section, we initially provide an overview of the context of our SE courses. Then, we introduce the SE2014 knowledge areas integrated into our SE curriculum. Subsequently, we conduct a thorough analysis of each knowledge area, delineating the potential integration of generative AI techniques into our SE education.

A. Course Context

Software Design (CS3342) and Software Engineering Practice (CS3343) are two core undergraduate courses tailored for computer science students at City University of Hong Kong. These courses typically enroll over 200 students annually. Within this academic framework, students engage in a comprehensive exploration of the software development lifecycle, covering areas such as requirement analysis, software design, implementation, and testing. Through active participation in these courses, students gain hands-on experience in applying theoretical concepts to real-world scenarios, thereby honing their skills and competencies in SE methodologies and practices.

B. SE2014 Knowledge Areas

As delineated in [9], ten knowledge areas are proposed to guide high-quality SE education. In conjunction with the course syllabus examined in our study, we present the knowledge areas as outlined in Table I.

Integrating with our course context, the domains of REQ, DES, and MAA are primarily featured within the curriculum of course CS3342. In this course, students engage in the task of designing and modeling software based on their comprehensive analysis of the requirements. Conversely, course CS3343 encompasses all the aforementioned areas, providing students with an immersive experience across the entire software development lifecycle. This course predominantly emphasizes the domains of CMP, VAV, QUA, and PRF.

In the area of REQ, fundamental processes governing software development are established, with a focus on tasks reliant upon natural languages. Our previous investigation [13] identifies a disparity between student performance and industry standards in this realm. Generative AI methodologies, particularly LLMs, present an avenue for educating students in requirement analysis tasks, particularly in crafting meticulously documented specifications. The proper prompt snapper holds the potential to augment both the efficiency and accuracy of requirement analysis procedures [14].

TABLE I. KNOWLEDGE AREAS INCLUDED IN OUR STUDY

Acronym	Name	Description
REQ	Requirements Analysis and Specification	This domain encompasses the process of constructing requirements, which involves eliciting and analyzing stakeholders' needs and crafting a comprehensive description of desired system behavior qualities, as well as relevant constraints and assumptions.
DES	Software Design	Software design involves the consideration of various issues, techniques, strategies, representations, and patterns to determine the optimal approach for implementing either a component or an entire system.
MAA	Software Modeling and Analysis	Modeling and analysis are fundamental concepts across engineering disciplines, essential for documenting design decisions and evaluating alternative approaches to ensure the efficacy and integrity of software systems.
CMP	Computing Essentials	This domain encapsulates the fundamental principles of computer science that underpin the design and construction of software products. It encompasses knowledge pertaining to the transition from design to implementation, along with the methodologies and tools employed throughout this progression.
VAV	Software Verification and Validation	This domain employs a range of techniques to verify and validate software components or systems, ensuring compliance with specified requirements and alignment with stakeholder expectations.
QUA	Software Quality	Recognizing the paramount importance of software quality, this domain addresses quality concerns comprehensively, providing a framework for achieving and maintaining high standards throughout the software engineering lifecycle.
PRF	Professional Practice	Professional practice focuses on instilling the knowledge, skills, and ethical attitudes necessary for software engineers to engage in their profession professionally, responsibly, and ethically.

In the domains of DES and MAA, a pivotal activity involves generating conceptual models based on requirement analysis, which lays the groundwork for code implementation. However, these domains have received less attention. A recent study [15] explores the potential integration of LLMs for SE modeling, revealing limited performance in software modeling tasks compared to code generation. Therefore, we exclude design and modeling aspects in our proposed vision.

CMP constitutes the cornerstone of SE project tasks, encompassing coding implementation. Generative AI models, fine-tuned for various code-related tasks such as code generation [3] and refactoring [2], are potentially integral to CS3343. Leveraging advancements in code-related tasks, we anticipate integrating auto code completion into our SE courses.

In the realm of VAV, auto-testing code generation holds promise for enhancing students' proficiency in testing methodologies. The empirical study [16] reveals the majority of the LLM-generated tests contain non-trivial assertions, with every component of information embedded within the prompt contributing to the quality of the generated tests. Given the demonstrated effectiveness of generative AI models in code generation for testing purposes, their integration into automated testing frameworks emerges as a promising avenue for augmenting students' proficiency in this domain.

QUA primarily concerns code quality, wherein generative AI tools can assist by analyzing code bases using LLMs and offering suggestions for code refactoring and optimization. LLMs play a pivotal role in facilitating comprehensive code comprehension through the provision of open-ended prompts [17], thus enabling the customization of prompts tailored to augment students' comprehension of code and facilitate the production of high-quality code.

Regarding PRF, our focus lies on ethical knowledge integration into generative AI tools, which serves to augment students' understanding. While less technically oriented, this integration is pivotal in the development of generative AI tools, especially in supporting students' comprehension of ethical knowledge in the SE process [18].

IV. VISON

In this section, we systematically present our vision for Intended Learning Outcomes (ILOs) and the deliverables. Table II provides an overview of the mapping between our proposed integration and the corresponding knowledge areas.

TABLE II. PROPOSED INTEGRATION BASED ON KNOWLEDGE AREA IN SE EDUCATION

Deliverables	ILOs	Knowledge Areas
D1	ILO1, ILO5, ILO6, ILO7	CMP, QUA
D2	ILO2, ILO3	VAV, QUA
D3	ILO2, ILO3, ILO7	CMP, QUA
D4	ILO1, ILO3	VAV, QUA
D5	ILO4, ILO5	REQ, PRF, QUA

A. Intended Learning Outcomes

Based on the analysis of the knowledge areas aligned with our course curricula, we propose the following ILOs in our vision.

- **Proficiency in Efficient Coding Techniques (ILO1):** Through the use of generative AI tools, students are expected to develop proficiency in writing efficient and optimized code. They should demonstrate an understanding of code generation, auto-completion, and refactoring techniques to improve code quality, readability, and performance.
- **Problem-Solving and Debugging Skills (ILO2):** By utilizing generative AI tools for bug detection and debugging, students should enhance their problem-solving abilities. They should be able to identify and resolve code errors, analyze and interpret debugging suggestions, and effectively debug complex software systems.
- **Understanding of Testing and Quality Assurance (ILO3):** Integration of generative AI tools in automated testing and quality assurance processes should enable students to gain a thorough understanding of testing methodologies and techniques. They should be able to generate comprehensive test cases, analyze test results, and ensure software quality through effective testing practices.
- **Documentation Proficiency (ILO4):** Generative AI tools can assist students in developing documentation

skills. Students should be able to create well-structured and comprehensive documentation for software projects, including code comments, API documentation, and explanations for complex code snippets.

- **Adaptability to AI Technologies (ILO5):** Through exposure to generative AI tools, students should develop an understanding of AI technologies and their applications in software development. They should be able to adapt to and leverage AI tools effectively, keeping pace with advancements in the field and incorporating them into their development practices.
- **Collaboration and Communication (ILO6):** The use of generative AI tools may promote collaboration and communication among students in project groups, as well as project repositories such as GitHub. Through discussions, sharing of code snippets, and providing feedback on AI-generated suggestions, students should develop effective collaboration and communication skills needed in a software development context.
- **Critical Thinking and Creativity (ILO7):** Generative AI tools can stimulate critical thinking and creativity by providing students with alternative coding suggestions and optimization techniques. Students should be encouraged to evaluate and analyze AI-generated suggestions critically, selecting the most appropriate solutions and applying creative thinking to solve complex software development challenges.
- **Industry Readiness (ILO8):** By incorporating generative AI tools into the software development course, students should be better prepared for industry practices and expectations. Students should have hands-on experience with AI-powered tools that are widely used in the software development industry, making them more competitive and ready to contribute effectively to real-world projects.

B. Deliverables

The primary emphasis of our vision focuses on the CS3343 Software Engineering Practice course, which provides an ideal platform for incorporating practical assisted AI learning elements. Additionally, the CS3342 Software Design course, which serves as a prerequisite for CS3343, is also pertinent. With a particular focus on Software Design in theory, students will be introduced to the concept of AI-assisted software engineering and explore the profound influence of advanced AI technologies in the IT industry.

1) ***D1: Code Generation and Auto-Completion with AI****.* Through the utilization of generative AI tools, code generation, and auto-completion processes can be automated, resulting in improved productivity and efficiency for students [3, 4]. By leveraging AI models that analyze patterns and structures from existing codebases, these tools can generate code snippets, templates, and even complete functions. This functionality empowers students to concentrate more on high-level problem-solving rather than dedicating excessive time to repetitive coding tasks. It aligns with the role of professional software engineers, who primarily focus on software development projects, design, planning, and system architecture, while coding tasks can be outsourced and automated using AI. This integration of generative AI tools enables students to develop a skill set that aligns with the demands of the industry, emphasizing critical thinking and problem-solving abilities over mundane coding activities.

2) ***D2: Bug Detection, Reporting, and Debugging with AI.*** Students can benefit from assistance in detecting and debugging code errors and issues. AI models have the capability to analyze code syntax, logic, and patterns, enabling them to identify potential bugs and propose relevant debugging solutions [2, 17]. This not only enhances students' understanding of common programming mistakes but also strengthens their problem-solving skills and proficiency in writing robust and error-free code. Furthermore, AI can contribute to more systematic bug reporting, facilitating the production of high-quality documentation across project teams. This integration of generative AI tools empowers students to develop comprehensive debugging capabilities, fostering a culture of quality assurance and effective collaboration in software development projects.

3) ***D3: Code Refactoring and Optimization with AI.*** Teaching students about code refactoring and optimization is crucial for developing efficient and maintainable software. Generative AI tools can aid in this process by analyzing codebases using LLM and providing suggestions for refactoring and optimizing code [2]; additional categories of code smells could be introduced to the students. AI models can identify areas where code can be simplified, performance can be improved, and best practices can be applied. By incorporating these suggestions, students learn valuable techniques for writing clean, efficient, and scalable code.

4) ***D4: Automated Testing and Quality Assurance with AI.*** The CS3343 Software Engineering Practice course has incorporated automated tools like JUnit Testing Suites to automate software teaching and is welcomed by the IT industry. These tools play a crucial role in software development by enabling regression testing, which ensures the quality of the software. Generative AI tools offer the capability to automate diverse facets of testing [5, 16], encompassing test case generation, test scenario generation, and test result analysis. By leveraging AI models, students can shift their focus toward testing strategies and ensuring test comprehensiveness while AI automates the generation of test cases. These tools analyze code and generate test cases that encompass various code paths, edge cases, and potential bugs. This automation empowers students to gain practical experience in employing comprehensive testing techniques and fosters a culture of quality assurance within their development practices, thereby contributing to the creation of robust and reliable software solutions.

5) ***D5: Documentation and Documentation Generation with AI.*** Comprehensive software project documentation plays a critical role in understanding and maintaining software systems. To aid students in this aspect, generative AI tools can be employed to generate design and project documentation [7, 8]. By analyzing code comments, function signatures, and code structure, generative AI models can suggest relevant documentation templates, generate API documentation, and even provide explanations for code snippets. This functionality enables students to enhance their documentation skills and encourages them to create thorough

and easily understandable documentation for their projects. With the integration, students can develop proficiency in producing high-quality documentation, which facilitates effective communication, promotes knowledge sharing, and contributes to long-term maintainability of software systems.

C. Constraint and Evaluation Plan

In the development of generative AI tools, a critical determinant of success resides in the meticulous design of prompts [19] tailored to our course curriculum. Within the framework of our proposed strategy development, a potential challenge arises in crafting human-curated prompts. To mitigate this challenge, we advocate for the involvement of experts and practitioners from diverse fields to contribute to the crafting of prompts.

Furthermore, we will employ learning analytics frameworks [20], which can be utilized to measure and evaluate the effectiveness of technology-enriched learning, particularly relevant to AI-assisted learning in the project. Through the analysis of data derived from student interactions with AI tools, learning analytics furnishes valuable insights into student advancement, involvement, and educational achievements. The data collection primarily encompasses four measurable dimensions: 1) data of completed and passed courses; 2) student performance metrics and data; 3) student behavior measures; and 4) engagement with training material.

V. Conclusion and Future Work

In this paper, we have presented practical strategies for the integration of Generative AI in Software Engineering Education. These strategies are based on the guidelines from SE2014 and have been tailored to our specific SE course context to achieve improved learning outcomes for software engineering courses. To realize these strategies, we plan to develop generative AI tools specifically designed for our SE courses to enhance students' learning experiences in software development process, particularly in code-related tasks. These tools automate repetitive tasks, offer intelligent suggestions, and provide support across various aspects of software development. As a result, students are empowered to become more efficient, productive, and skilled developers. Overall, by embracing these cutting-edge tools, we can ensure that our software development courses remain competitive and up-to-date with the latest advancements, and equip students with the necessary skills to excel in their careers.

Acknowledgment

This work is supported in part by the General Research Fund of the Research Grants Council of Hong Kong and the research funds of the City University of Hong Kong (6000796, 9229109, 9229098, 9220103).

References

[1] A. Fan et al., "Large language models for software engineering: Survey and open problems," arXiv preprint arXiv:2310.03533, 2023.

[2] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," arXiv preprint arXiv:2303.07839, 2023.

[3] Q. Zheng et al., "Codegeex: A pre-trained model for code generation with multilingual benchmarking on humaneval-x," in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 5673-5684.

[4] F. Zhang et al., "Repocoder: Repository-level code completion through iterative retrieval and generation," arXiv preprint arXiv:2303.12570, 2023.

[5] S. Yu, C. Fang, Y. Ling, C. Wu, and Z. Chen, "LLM for test script generation and migration: Challenges, capabilities, and opportunities," in 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS), 2023: IEEE, pp. 206-217.

[6] Y. Zhang, Z. Jin, Y. Xing, and G. Li, "STEAM: simulating the interactive behavior of programmers for automatic bug fixing," arXiv preprint arXiv:2308.14460, 2023.

[7] C. Bull and A. Kharrufa, "Generative AI Assistants in Software Development Education: A vision for integrating Generative AI into educational practice, not instinctively defending against it," IEEE Software, 2023.

[8] O. Petrovska, L. Clift, F. Moller, and R. Pearsall, "Incorporating Generative AI into Software Development Education," in Proceedings of the 8th Conference on Computing Education Practice, 2024, pp. 37-40.

[9] M. Ardis, D. Budgen, G. W. Hislop, J. Offutt, M. Sebern, and W. Visser, "SE 2014: Curriculum guidelines for undergraduate degree programs in software engineering," Computer, vol. 48, no. 11, pp. 106-109, 2015.

[10] S. Ouhbi and N. Pombo, "Software engineering education: Challenges and perspectives," in 2020 IEEE Global Engineering Education Conference (EDUCON), 2020: IEEE, pp. 202-209.

[11] V. Garousi, G. Giray, E. Tuzun, C. Catal, and M. Felderer, "Closing the gap between software engineering education and industrial needs," IEEE software, vol. 37, no. 2, pp. 68-77, 2019.

[12] M. Daun and J. Brings, "How chatgpt will change software engineering education," in Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, 2023, pp. 110-116.

[13] Y. Li, J. Keung, X. Ma, J. Zhang, Z. Yang, and S. Liu, "Learning Gaps in Project-based Requirements Engineering Education-A Case Study of Student Projects," in 2023 International Symposium on Educational Technology (ISET), 2023: IEEE, pp. 239-243.

[14] Y. Cheng, J. Chen, Q. Huang, Z. Xing, X. Xu, and Q. Lu, "Prompt sapper: a LLM-empowered production tool for building AI chains," ACM Transactions on Software Engineering and Methodology, 2023.

[15] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo, "On the assessment of generative AI in modeling tasks: an experience report with ChatGPT and UML," Software and Systems Modeling, vol. 22, no. 3, pp. 781-793, 2023.

[16] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," IEEE Transactions on Software Engineering, 2023.

[17] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, "Using an llm to help with code understanding," in Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1-13.

[18] V. D. Kirova, C. S. Ku, J. R. Laracy, and T. J. Marlowe, "Software Engineering Education Must Adapt and Evolve for an LLM Environment," in Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 2024, pp. 666-672.

[19] J. White et al., "A prompt pattern catalog to enhance prompt engineering with chatgpt," arXiv preprint arXiv:2302.11382, 2023.

[20] P. Leitner, M. Khalil, and M. Ebner, "Learning analytics in higher education—a literature review," Learning analytics: Fundaments, applications, and trends: A view of the current state of the art to enhance E-learning, pp. 1-23, 2017.