IIAI Open Conference Publication Series
IIAI Letters on Informatics and Interdisciplinary Research
Vol.006, LIIR388
DOI: https://doi.org/10.52731/liir.v006.388

Preliminary Practices for Java Programming Tools and TDD Courses Utilizing Generative AI and Online Education

Mika Ohtsuki [*], Tetsuro Kakeshita [*]

Abstract

With the rapid advancement of generative AI, automation is increasingly being introduced across various stages of software development. In response to these changes, programming education must also evolve to incorporate the use of generative AI from the outset. In this study, we designed and implemented intermediate-level programming courses that integrate generative AI tools such as GitHub Copilot. The curriculum consisted of three subjects: Object-Oriented Programming, Test-Driven Development, and Practical Project Development. Each course combined on-demand instructional materials with AI-assisted exercises. As a result, learners reported high levels of satisfaction and frequently accessed course materials and assessments. Notably, many students demonstrated the ability to critically evaluate and adapt AI-generated suggestions rather than relying on them uncritically. A comparative survey between GitHub Copilot and Google Gemini revealed that students were also beginning to select AI tools based on purpose and context. These findings indicate the potential of educational designs that foster practical programming skills and cultivate AI literacy. This initiative highlights the promise of programming education that is both AI-integrated and personalized, offering new directions for curriculum innovation in higher education.

Keywords: Programming education, GitHub Copilot, Object-oriented programming, Test-Driven Development.

1 Introduction

The rapid advancement of generative AI has led to increased AI utilization across various stages of software development, from design to testing, driving the restructuring of development processes. As a result, programming and software engineering education at universities and technical colleges are also being compelled to reevaluate their educational frameworks, shifting from traditional education based on “human-driven programming” to education designed around “collaboration with generative AI” [1].

AI tools such as ChatGPT [2] and GitHub Copilot [3] are increasingly being used to support be-ginners' learning and provide individual feedback, attracting attention even in introductory programming education [4][5][6][7]. However, a more practical and step-by-step instructional design is required to cultivate skills that enable users to utilize AI outputs without overreliance on them while evaluating and revising proposed content.

[*] Saga University, Saga, Japan

This study aims to establish a methodology for intermediate-level programming education that assumes collaboration with generative AI. We designed a three-course curriculum incorporating a software engineering perspective, consisting of object-oriented programming, test-driven development, and practical project development.

This curriculum utilizes generative AI tools such as GitHub Copilot but emphasizes a process in which learners select, revise, and integrate AI suggestions rather than blindly accepting them. This approach differs from previous studies that focused on individual support for beginners, instead prioritizing the development of AI literacy and autonomous learning behaviors (learner agency). Additionally, it aims to cultivate metacognitive skills to compare multiple AI tools and use them appropriately for specific purposes.

In this paper, we report the results of a pilot implementation of this curriculum and a multi-faceted evaluation of its educational effectiveness based on learning histories, assignment submissions, and surveys. This paper is structured as follows. We provide an overview of related research in Section 2. Section 3 describes the educational support system and the designed course. Section 4 outlines the teaching methods. Section 5 presents analysis results using logs and assignments. We analyze the participant survey in Section 6. Then, Section 7 contains a conclusion with a summary and prospects.

2 Related Research

A ChatGPT was released to the public in November 2022, and education to utilize generative AI remains a new field [8]. Regarding programming education, several studies have been reported on the impact of generative AI on student learning [1][4][5][6][7]. These studies have shown that generative AI effectively supports the acquisition of programming concepts and provides individualized feedback while highlighting the challenge of identifying AI errors.

In terms of exercise support, Kazemitabaar et al. [6] reported on automatic support for beginners using Codex, and Leinonen et al. [9] presented a case study on error support. These studies assume that “humans create programs,” positioning generative AI as an auxiliary tool. On the other hand, Bull et al. [7] and Prather et al. [10] discuss the potential of integrating AI coding assistants such as GitHub Copilot into education, analyzing how students accept AI suggestions and experience difficulties. Additionally, Prather et al. [11] provide a comprehensive review of research, practice, and tools related to the educational use of generative AI, which is useful as background for this study.

Practical examples of integrating generative AI into educational curricula include a report on curriculum design support by Western Governors University [12] and a case study on the integration of Claude by Northeastern University and Anthropic [13].

This study builds on these previous studies and has the following unique features: (1) practical curriculum design and implementation through multiple lectures for intermediate-level learners, (2) multi-faceted evaluation using learning logs, grades, and questionnaires, and (3) a comparative study of multiple generative AI tools. In particular, by analyzing how the use of AI in exercises and assignments leads to behavioral changes in students, this study provides new insights that complement existing research.

3 Design of Intermediate Programming Education Utilizing Generative AI

3.1 Role Shift in Programming with Generative AI

Figure 1 shows (a) the conventional programming procedure and (b) a new procedure utilizing generative AI. In conventional programming, all tasks, from specification determination to maintenance, were performed by humans. The utilization of generative AI is expected to improve the efficiency and automation of program development; however, generative AI has issues such as hallucination, copyright issues, and information security risks, making the direct use of generated outputs highly risky. Therefore, since the human evaluation of the generated outputs is essential, the role of humans in programming shifts to providing instructions to the generative AI and verifying the outputs it generates. As a result, humans and generative AI will collaborate interactively, dividing tasks between them.

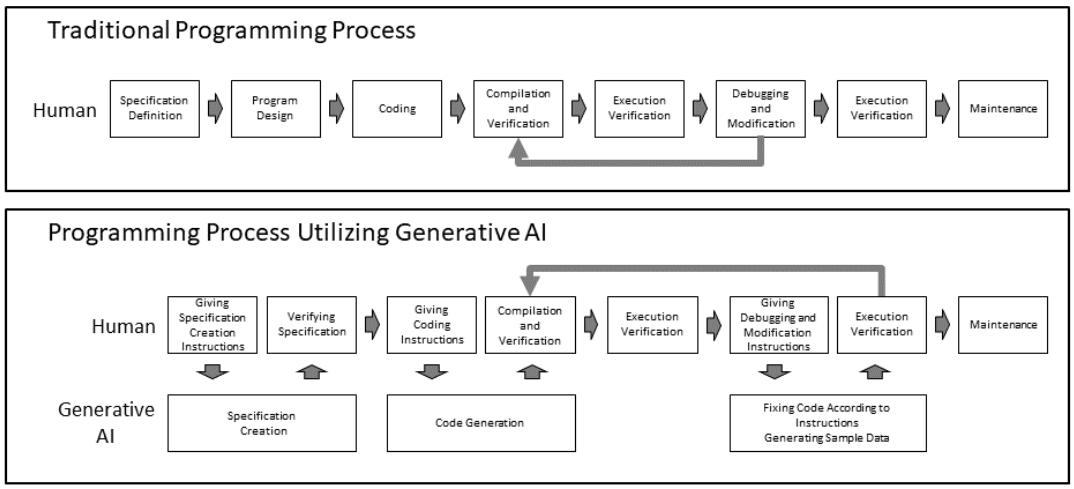

Figure 1: Conventional and new procedure for programming utilizing generative AI

3.2 Requirements for Educational Reform

Coding is the role of generative AI in programming that utilizes generative AI, but basic programming knowledge is necessary to evaluate the generated output. Traditionally, coding exercises have been used to solidify understanding of these concepts. In AI-assisted programming, understanding basic concepts and algorithms is expected to shift to using the code and explanations proposed by generative AI and further applying algorithms. Appropriate prompts are required to generate the desired code using generative AI. This necessitates the ability to correctly understand problems and break them down into smaller problems, making the education of problem-solving skills more critical.

Understanding the mechanisms and issues of generative AI is essential when utilizing generative AI to make decisions that consider ethical aspects and the responsibilities associated with its adoption. Additionally, it is essential to note that mastering code generation AI tools and programming techniques will become a new requirement.

4 Curriculum Design for Intermediate-Level Programming

At the intermediate-level of programming education, the goals are to develop the skills necessary for integrated application development based on the fundamental programming skills learned at the beginner level. Another objective is to learn how to utilize coding support provided by generative AI during this process.

In this study, we designed an intermediate programming education program that emphasizes the gradual acquisition of practical development skills and software engineering concepts. We established three courses, as shown in Table 1. Java was selected as the programming language. Java has been traditionally used in intermediate programming education at Saga University, serving as a practical target and widely adopted in business application development.

These courses are designed with generative AI (primarily GitHub Copilot) as a prerequisite, leveraging the tool's characteristics to enable experiences and efficient learning that are difficult to achieve through traditional educational methods.

Table 1: Subjects at the intermediate-level

Object-Oriented Programming（OOP）	
Objective	Learn object-oriented programming from basics to advanced concepts in a step-by-step manner. Emphasize understanding concepts over syntax, and utilize code completion and error explanations provided by generative AI to deepen your understanding.
Contents	Classes, inheritance, polymorphism / Design patterns / Exception handling / OOP exercises
Tools	GitHub Copilot, Eclipse
AI Usage	Deepen students' understanding of concepts by receiving support in reading and revising proposal codes.
Test-Driven Development（TDD）	
Objective	This course aims to understand TDD's basic concepts and development process and apply them using generative AI. Students will learn the importance of testing and design techniques by interpreting test cases automatically generated by AI and implementing them based on those cases.
Contents	What is software engineering? / Process models / Specification development / Software testing / TDD exercises
Tools	GitHub Copilot, Eclipse, JUnit
AI Usage	Read design and specifications through test code generation support.
Practical Project Development（PPDev）	
Objective	Simulate actual application development and experience in everything from design and implementation to project management. Utilize generative AI, CI/CD tools, and issue management tools to recreate a modern development environment.
Contents	Basic design/Detailed design/GitHub/UML/Web application exercises/CI tool utilization
Tools	GitHub Copilot, GitHub, CI/CD, UML Drawing Tool
AI Usage	Practical development utilizing AI support in a modern development environment

4.1 Object-Oriented Programming（OOP）

This course covers the basic concepts of object-oriented programming (OOP), program design, and implementation methods based on those concepts. OOP is essential for making software structures more transparent and extensible. Using generative AI as an auxiliary tool makes it possible to efficiently address grammatical errors and acquire practical design skills.

In this course, Java will be the programming language. Students will progressively learn concepts such as classes, inheritance, and polymorphism through coding exercises in Eclipse. Generative AI (GitHub Copilot) will assist with code examples, error messages, and suggestions for simple code syntax.

In exercises, the focus is on developing the ability to critically analyze the content while efficiently advancing development using GitHub Copilot's suggestions. The goal is to cultivate autonomous programming skills using AI as an auxiliary tool rather than excessively relying on it.

4.2 Test-Driven Development（TDD）

In this course, students will learn the basic software engineering concepts and then apply the fundamental techniques and practices of test-driven development (TDD). TDD [14] is a development methodology that involves repeating the cycle of "write tests → write code → refactor," with the support of generative AI, students can efficiently and practically learn this process.

In this course, students combine GitHub Copilot and Eclipse and execute test codes using JUnit. They will implement code based on test codes automatically generated by AI and repeatedly improve the code so that it passes the tests. The objective of this course is to cultivate the ability to read and understand AI output, consider its intent and specifications, and implement code accordingly.

The course content is designed to enable students to progressively master the construction of a development environment, generative AI, Eclipse operations, and JUnit usage. Demonstration videos and supplementary materials are provided to facilitate self-directed learning.

4.3 Practical Project Development（PPDev）

In this course, students will apply their foundational knowledge of software engineering and object-oriented programming techniques to project-based exercises designed to simulate real-world application development. The goal is to develop comprehensive software development skills by progressively experiencing each phase of the project development process.

A distinctive feature of this course is that students can experience efficient development in both team and individual development by utilizing generative AI (GitHub Copilot) for code completion and design support. In addition, modern development tools such as continuous integration (CI) [15] and issue management tools are introduced, allowing students to gain hands-on experience in a practical development environment.

This course is designed to help students understand the practical flow of software development and gain experience combining tools to advance development through a series of processes, from planning to design, implementation, and management. Generative AI functions as

an “auxiliary partner” in each phase, designed to enhance students' judgment and problem-solving skills.

4.4 Instructional Planning and Execution

As a preliminary practice, we conducted classes on three subjects prepared for students in our research lab. The class content was provided to students using Moodle. There were five to seven fourth-year students specializing in information engineering. Although the small sample size limits the generalizability of findings, it enabled close monitoring of individual learning processes and provided rich insights into how each student adapted AI-generated suggestions to their needs.

Table 2 shows the implementation schedule. Classes were held every Friday, and students worked on the content independently. Any content or assignments that could not be completed during class were to be worked on outside class time.

Table 2: Trial Course Schedule

Subject	Start Date	End Date	Assignment Deadline	# of Students
OOP	Nov. 29, 2024	Dec. 12, 2024	Jan. 10, 2025	7
TDD	Dec. 13, 2024	Dec. 25, 2024	Jan. 21, 2025	7
PPDev	Jan. 10, 2025	Jan. 31, 2025	Jan. 31, 2025	6

5 Evaluation and Analysis

In this section, we analyze learners' behavioral tendencies, learning achievement, and the actual use of AI based on Moodle access logs, quiz scores, and exercise assignment submissions.

5.1 Learning History Analysis

Table 3 shows the viewing status of teaching materials and videos and the test participation status, while Table 4 shows the utilization status of exercise using AI teaching materials and reference materials. Across all three subjects, access to lecture materials, videos, and quizzes was very high, indicating that repetitive learning behaviors had become established. In particular, in test-driven development (TDD) and practical project development (PPDev), access to AI-supported exercise materials and submission pages was active, demonstrating a high interest in practical exercises. There were significant individual differences in the use of supplementary materials (external links, explanatory videos, etc.), and improving the structure to indicate the importance of these materials is a challenge for the future.

Table 3: Viewing status of teaching materials and videos as well as the test participation status

Course	Viewing the status of teaching materials and videos	Test participation status
OOP	The lecture videos for the first 10 sessions were accessed by multiple students, with the first video viewed 13 times by 10 students and the lecture materials viewed multiple times by approximately	Access was particularly high for the sixth and tenth of the ten sessions, with the sixth session recording the highest number of accesses at 283.

	7 students. A high level of engagement was maintained throughout the course.	
TDD	All six sessions saw numerous accesses to lecture materials and videos (e.g., Session 1: 14 accesses to materials by 7 participants, 8 accesses to videos by 7 participants).	All students took the test, accessed 217 times in total in the first round, clearly showing that they repeatedly worked on it.
PPDev	Five to six people accessed lecture videos on basic design and project management each time.	They were retaken frequently, up to 201 times (basic design 3) and 194 times (PM).

Table 4: Utilization status of exercise using AI teaching materials and reference materials

Course	Utilization status of exercise using AI teaching materials	Utilization status of reference materials
OOP	The use of AI-oriented training materials was particularly notable, including setup videos for Copilot (13 videos/7 participants) and basic Java exercise materials (38 materials/8 participants).	1 to 7 people also utilized external materials such as design patterns, UML, collections, and stack traces, and access to supplementary reading materials was maintained at a relatively high level.
TDD	JUnit exercise materials (32 items/8 students), exercise videos (17 total), and submission locations (78/8 students) showed a high level of interest in implementation exercises.	Supplementary URL materials such as the Agile Development Manifesto (3 items/3 people) and the Gompertz curve (1 item/1 person) were used to a limited extent.
PPDev	Six to eight people accessed all materials related to Git exercises, task management, and UML exercises multiple times, demonstrating a strong interest in practical development tools. In particular, the UML diagram drawing and Jenkins exercises were viewed more than 20 times per file.	External links (Copilot usage, Git official website, PlantUML server, etc.) were also utilized, and the use of supplementary materials in this course was relatively consistent.

5.2 Instructional Planning and Execution

The confirmation tests conducted in each subject achieved high participation rates and average scores (generally 90 percent or higher), indicating that repeated testing contributed to a deeper understanding of the material. The exercise assignments also demonstrated high submission rates, with many participants observed critically evaluating and revising their proposals while utilizing generative AI.

For example, in the TDD exercise, all participants submitted their assignments for the test design task using JUnit and demonstrated an attitude of reading and understanding the test code generated by AI while implementing it following the specifications. In OOP, the Java exercise tasks confirmed the consolidation of design concepts such as inheritance and polymorphism through AI-assisted completion. In PPDev, high submission rates were maintained in both the UML diagram exercise and the CI tool implementation exercise, indicating that practical learning was achieved throughout the development process.

To assess the quality of these submissions—particularly in the context of AI-assisted devel-

opment—we employed a rubric designed to capture both technical correctness and the depth of learner engagement. The rubric included the following criteria: (1) correctness and completeness of the final code, (2) alignment between comments and implemented functionality (i.e., whether learners articulated and reflected on AI-generated suggestions), and (3) meaningful revision history as recorded in version control systems. This last point allowed us to observe whether students iteratively modified AI-generated outputs or engaged in reflective improvement processes.

These results suggest the potential for AI to be utilized not merely as a convenient tool but as a scaffold to facilitate understanding of learning content and to promote reflective behavioral change.

6 Editorial Policies for General Issues

In this section, we analyze evaluations of course design and the use of generative AI based on the results of a student survey conducted in three courses: “Object-Oriented Programming (OOP),” “Test-Driven Development (TDD),” and “Practical Project Development (PPDev).”

The surveys were conducted online via a form on Moodle immediately after each subject was completed. Each question was designed as a multiple-choice or open-ended format and collected opinions on participants' satisfaction, study time, evaluation of course materials, usage of generative AI, and the reasons behind their usage. The number of respondents for each subject is shown in Table 5.

Table 5: The number of respondents for each subject

Course	# of participants	# of respondents	Response rate
TDD	7	5	71.4%
OOP	7	5	71.4%
PPDev	6	5	83.3%

The response rate was over 70% for all questions, and despite the small number of respondents, we determined that the data was reliable enough to analyze trends among the participants.

The common questions are organized into the following categories:

- Course satisfaction: on-demand format, teaching materials, exercises, etc.
- Learning behavior: time spent on learning, frequency of review, etc.
- Use and evaluation of generative AI: Tools used (Copilot, ChatGPT, Gemini, etc.), usage scenarios, and evaluations
- Open-ended comments: Opinions on the course, materials, and use of generative AI, as well as improvement suggestions, etc.

Additionally, for the OOP course, a special survey titled “Comparison Survey of GitHub Copilot and Google Gemini” was conducted to collect participants' opinions on the usability of AI-assisted tools.

6.1 Satisfaction and Learning Behavior Trends (Quantitative Analysis)

For on-demand classes, 60% to 83% of respondents were “very satisfied” across all subjects, and 100% were “somewhat satisfied” or higher. This confirmed that on-demand classes are highly compatible with flexible learning styles. In OOP, 83% of respondents chose “very satisfied,” suggesting that learning at their own pace may have enhanced the learning experience.

Regarding “time spent on learning (per week),” options such as “less than 10 hours” and “10 to 20 hours” were surveyed. As a result, all respondents in OOP chose “less than 10 hours,” while in TDD, 4 out of 5 chose “less than 10 hours,” and 1 chose “10 to 20 hours.” On the other hand, in PPDev, 4 out of 5 respondents chose “10-20 hours,” and 1 chose “20-30 hours.” It is presumed that the workload was relatively high in PPDev due to the large number of exercises.

The course materials (slide materials, videos) and exercise assignments were evaluated on a 5-point scale. Regarding the clarity of the materials, three respondents in OOP, 1 in TDD, and 3 in PPDev answered that they were “very clear.” On the other hand, regarding the exercise assignments, two respondents in OOP, 1 in TDD, and 4 in PPDev answered that they were “very effective” in understanding the content of the experiments. These results suggest a certain level of understanding and satisfaction with the course content.

Regarding the use of generative AI, GitHub Copilot was widely used across all subjects, with all participants reporting experience using it in TDD and PPDev. In PPDev, Google Gemini was also used alongside Copilot, and a special survey analyzed the comparison results (see Section 6.3 for details). The primary purpose of using Copilot was “code completion,” suggesting that generative AI practically supported students' coding activities.

6.2 Analysis and Commentary on Free Responses

In this section, we analyze the open-ended feedback provided by participants in three courses (TDD, OOP, and PPDev). We present their evaluations of the courses, their opinions on using generative AI, and the challenges they faced in their learning.

Many participants commented positively about the course's on-demand format, such as “I can learn at my own pace” and “It is easy to review.”

In the free-response comments related to generative AI, the following feedback was received regarding the use of GitHub Copilot:

- The code quality suggested during input was high, which was helpful for learning.
- Since the AI's suggestions were not always correct, verifying them on my own was necessary.
- By progressing while understanding the meaning of the suggested code, my understanding deepened

Additionally, among participants who were hesitant to use AI, comments such as “I didn't feel it was necessary, so I didn't use it” were common, reflecting an autonomous learning attitude.

6.3 Comparison Results between Gemini and Copilot (Special Survey)

PPDev conducted implementation exercises utilizing multiple generative AI services. We conducted a special survey comparing GitHub Copilot and Google Gemini, two representative code generation support tools, to collect participants' subjective evaluations regarding usability and output quality.

The survey included the following two questions:

Q1. Which of GitHub Copilot and Google Gemini was superior? (Options: Copilot was better / Gemini was better / They were about the same, etc.)

Q2. Please explain your reasons (free response)

This survey received responses from five participants who had used both tools briefly. The results showed that three participants rated Copilot as better, one rated Gemini as better, and one rated them as about the same (depending on the case).

Reasons for highly evaluating Copilot included “high consistency of the proposed code” and “familiar and easy to use in an IDE,” highlighting its strengths as an IDE-integrated AI. On the other hand, participants who evaluated Gemini mentioned “ease of specifying what is needed,” highlighting the advantages of its chat-based interface.

The following insights can be drawn from these results:

- Real-time completion-type (Copilot) contributes to improving development efficiency in short periods
- Chat-based generative AI (Gemini) is useful for code understanding and design support
- The educational significance of using tools appropriately according to the learning stage or combining them is significant.

In the future, instructional design that allows students to selectively utilize AI tools (“choosing AI that suits them and using it intentionally”) will likely become important.

It should be noted, however, that this survey was exploratory and based solely on subjective impressions from a small number of participants. Since no structured evaluation criteria (e.g., usability dimensions, accuracy, context relevance) were established, the findings remain at a surface level. For these results to serve as actionable guidance for educators and tool designers, more detailed comparative studies with task-specific benchmarks and larger sample sizes are needed in the future.

6.4 Comprehensive Considerations and Future Prospects

This section presents a comprehensive analysis of student evaluations regarding the on-demand lecture format, instructional materials and exercises, and the utilization of generative AI based on surveys administered to participants in three courses.

High satisfaction with the on-demand format and a deeper understanding gained through exercise assignments indicated that the instructional design effectively balanced temporal flexibil-

ity and promoted gradual understanding. On the other hand, there were also requests for improvements in the viewing load of videos and the clarity of materials, indicating that ensuring the organization and searchability of information is key in designing materials that assume self-directed learning.

Generative AI centered on GitHub Copilot played diverse roles, including implementation support, design assistance, and test creation support, promoting learners' motivation and understanding. Furthermore, a comparative study of Copilot and Gemini suggested that the choice of AI tools depends on learning styles and objectives, indicating the need to incorporate "AI literacy" and "tool selection skills" into future training programs.

Beyond these motivational and behavioral aspects, another critical educational benefit lies in cultivating cognitive and metacognitive skills. In addition to enhancing learning motivation, using generative AI tools like Copilot and Gemini also supports the development of such skills. For instance, debugging AI-generated code promotes error identification and abstraction, while prompt refinement fosters problem decomposition. Moreover, the deliberate selection and comparison of AI tools based on context enhances self-monitoring and reflective judgment. These practices align with educational theories such as the metacognitive loop, in which learners plan, monitor, and evaluate their learning processes [16]. Thus, AI tools can catalyze deeper thinking rather than simple automation devices, particularly in intermediate-level programming education. While generative AI is not a panacea, it has the potential to serve as a "guide" that helps beginners gain a more profound and faster understanding when used appropriately.

The findings of this study are preliminary insights based on small-scale and limited practical implementation. Nonetheless, the following prospects can be identified: (1) comparative practical implementation with a broader range of AI tools (e.g., Claude, Mistral, etc.), (2) comparative analysis of learning outcomes between AI users and non-users, and integration with educational support systems (e.g., LMS or GitHub Classroom), (3) automation and personalization of AI intervention based on learning logs. In future learning environments where generative AI and human intellectual activities converge, it will become increasingly important to design systems that support learners' "choice and reflection."

Given that generative AI has shown specific effects in learning support, designing integrated support environments that enable multiple tools in a consistent context is essential. For example, introducing an environment that integrates development, design, dialogue, and metacognitive support—such as Cline [17], an integrated AI-based educational support environment—into educational settings is expected to offer the following possibilities: (1) ensuring continuity between coding, design, and feedback, (2) promoting transparency, reflection, and peer feedback through collaborative use of AI, and (3) enhancing the visibility of learning processes and supporting metacognitive activities such as planning, monitoring, and revision.

Such environments may help learners not only complete tasks more efficiently but also engage in deeper self-regulation and formative assessment. However, it is also necessary to remain aware of potential risks, including overreliance on AI suggestions or superficial engagement with code. Therefore, future research should examine how to balance AI-driven assistance with learner agency and critical thinking. Designing systems that allow users to utilize multiple AI support functions—not in isolation but within an integrated and pedagogically sound context—will be an essential direction for educational practice and technological development.

7 Conclusion and Future Work

This study explored the design and implementation of intermediate-level programming education using generative AI tools such as GitHub Copilot. We developed and conducted three courses—Object-Oriented Programming (OOP), Test-Driven Development (TDD), and Practical Project Development (PPDev)—to examine students' engagement with AI-assisted coding, testing, and project work. Although the sample size was small—comprising only six to seven students per course—this limitation enabled a more detailed observation of individual engagement patterns and tool adaptation strategies. Nonetheless, caution is warranted in generalizing these findings beyond the scope of this pilot.

Findings from learning logs, assessments, and learner feedback suggest several implications. First, generative AI supported self-regulated learning: students actively used AI tools for repeated practice and refinement, particularly in TDD and PPDev. Second, on-demand video lectures and modular resources enabled learners to study flexibly while maintaining deep engagement. Third, AI literacy emerged as a critical competency, including the ability to evaluate and choose tools like Copilot or Gemini. Finally, quizzes and assignments effectively tracked progress and encouraged practical, reflective learning through AI integration.

Future work should address scalability and sustainability through deeper LMS integration, AI-based tutoring, and analytics-driven feedback loops. Comparative studies involving AI-assisted and non-AI learner groups are needed to assess the pedagogical impact more rigorously. Longitudinal research could further explore how skills developed in AI-supported environments transfer to advanced coursework or real-world software development contexts.

Additionally, integrated environments such as Cline—which combine code generation, design scaffolding, dialogue-based interaction, and support for reflection—offer promising directions for enhancing metacognitive engagement and peer feedback. However, as such systems become more sophisticated, examining potential risks such as overreliance on AI-generated output or surface-level learning behaviors is critical. Embedding generative AI within structured, purpose-driven curricula must be accompanied by thoughtful instructional design that cultivates learner agency, critical thinking, and adaptive expertise in navigating complex development tasks.

Acknowledgment

We appreciate the students of the Faculty of Science and Engineering, Saga University, for their cooperation in this practice. This research is supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant Number 24K06418.

References

[1] M. Daun and J. Brings, "How ChatGPT Will Change Software Engineering Education," in Proc. ITiCSE, 2023, pp. 110–116.

[2] OpenAI, "ChatGPT."; Available: https://chatgpt.com/

[3] GitHub Inc., “GitHub Copilot.”; Available: https://docs.github.com/ja/copilot

[4] P. Haindl and G. Weinberger, “Students’ Experiences of Using ChatGPT in an Undergraduate Programming Course,” IEEE Access, vol. 12, pp. 43519–43529, 2024.

[5] M. Hu, T. Assadi, and H. Mahroeian, “Explicitly Introducing ChatGPT into First-year Programming Practice: Challenges and Impact,” in Proc. TALE, IEEE, 2023.

[6] M. Kazemitabaar et al., “Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming,” in Proc. CHI, ACM, 2023, pp. 1–23.

[7] B. Bull and A. Kharrufa, “Generative AI Assistants in Software Development Education,” arXiv preprint, arXiv:2303.13936, 2023.

[8] M. M. Rahman and Y. Watanobe, “ChatGPT for Education and Research: Opportunities, Threats, and Strategies,” Applied Sciences, vol. 13, no. 9, p. 5783, 2023.

[9] J. Leinonen et al., “Using Large Language Models to Enhance Programming Error Messages,” in Proc. SIGCSE, ACM, 2023, pp. 563–569.

[10] J. Prather et al., “It's Weird That it Knows What I Want": Usability and Interactions with Co-pilot for Novice Programmers,” arXiv preprint, arXiv:2304.02491, 2023.

[11] J. Prather et al., “Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools,” arXiv preprint, arXiv:2412.14732, 2024.

[12] Western Governors University, “How AI Is Reshaping Curriculum Design: Insights for Future Educators,” WGU Blog, 2025; https://www.wgu.edu/blog/how-ai-reshaping-curriculum-design-insights-future-educators2412.html

[13] Axios, “Northeastern University joins AI-higher ed experiment,” Axios Local Boston, Apr. 3, 2025; https://www.axios.com/local/boston/2025/04/03/northeastern-ai-claude-partnership

[14] K. Beck, Test-Driven Development: By Example, Addison-Wesley, 2002.

[15] K. Beck, “Embracing change with extreme programming,” IEEE Computer, Vol. 32, No. 10, pp. 70-77, 1999.

[16] J. H. Flavell, “Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry,” American Psychologist, vol. 34, no. 10, pp. 906–911, 1979.

[17] Cline Project, "Cline," GitHub; Available: https://github.com/cline