Introducing Coding Co-pilots in an Introductory Programming Course - the Beginning of the end?

Lars Michael Kristensen and Sven-Olai Høyland

Western Norway University of Applied Sciences (HVL)
Dept. of Computer Science, Electrical Engineering, and Mathematical Sciences
{lmkr,soh}@hvl.no

Abstract. Recent advances in generative AI and coding co-pilots in particular have fundamentally changed the tools available to students. For introductory programming courses this represents a challenging setting as assignments typically posed are well within reach of being automatically solved by current state-of-the-art generative AI technology. This in turn raises concerns about student learning outcomes in terms of fundamental programming knowledge and skills. At the same time, recent surveys indicate a considerable industry uptake of generative AI to boost productivity in software development. The latter suggests that in the course of their education, computer science students must learn to master these new technologies. This paper presents our initial investigations into introducing coding co-pilots in a first-year introductory programming course with a view towards embracing co-pilots and encouraging reflective use by the students.

Keywords: Teaching Programming · Coding co-pilots · Generative AI

1 Introduction

With the availability of generative AI technologies accelerated by the release of ChatGPT in the fall of 2022, students have obtained widespread access to powerful and potentially disruptive tools. This includes computer science students where coding co-pilots based on large language models are available in most software development environments. This represents a significant challenge in programming courses as AI-based tools on the one hand can be used as virtual labassistants to support learning, and on the other hand may lead to a copy-paste approach for solving programming assignments thereby diminishing learning outcomes [11]. The pervasive presence of generative AI has led some researchers [14] to revisit the fundamental question of *what is programming?* from an educational perspective, and contemplate whether there is a need to teach introductory programming in its traditional form in the future. A recent industry survey [18] on AI for software development indicates a considerable and increasing level of adoption.

The fact that it seems inevitable to address the widespread existence of co-pilots led us to investigate how co-pilots may be introduced in our DAT100 Basic

Programming course [7] at HVL. The course is a 10 ECTS course using Java as programming language and is taken by bachelor students in software engineering and information technology in their first semester. In terms of content, the course is a classical introductory programming course covering statements and expressions, control structures, methods, arrays, classes and inheritance, files and exceptions. In addition, the course covers use of an integrated development environment (IDE), debugging, unit-tests, and basic usage of Git for code collaboration and version control. The students have three medium-sized programming assignments and a programming project. These are mandatory and must be approved in order to enroll for the exam. In addition, there are a number of voluntary programming exercises each week. There is an intermediate test after four weeks to assess student learning outcomes, and towards the end of course there is an exam dry run (mini-exam) on a scaled down version of representative exam questions and assignments. The final exam is a digital four-hour written exam comprised of a multiple-choice part (25-30 %) and smaller programming assignments (70-75 %). The course grade is determined solely upon the final exam. No aids are allowed at the exam beyond the basic code editor integrated in WiseFlow which is being used for the digital exam. The basic editor supports syntax highlighting and indentation, but not compilation nor execution of code.

In the fall of 2024, we developed and introduced a coding co-pilot module in the course. The overall philosophy underpinning the module is that it is better to work *with* generative AI technology and explore its potential and limitations in programming education rather than working *against* it and attempting to ban students from using it. Hence, the aim of introducing a coding co-pilot module in the course was to openly discuss and reflect with the students on the role of generative AI in the education and in the profession of being a software developer. From an educational perspective, our aim was to address the following research questions (RQs):

RQ1 To what extent have students on their own initiative adopted generative AI as part of learning programming?
RQ2 What are the students' reflections on the challenges of using generative AI when learning programming?
RQ3 Does the availability of generative AI technologies impact student performance in terms of course grades?

The emphasis of the module was on the use of coding co-pilots in programming, not on the internal workings of generative AI. For answering RQ1 and RQ2, we conducted student surveys as an integrated part of the lectures in the module. This was followed up by a second survey after the students were encouraged to use co-pilots for the subsequent programming assignment. For answering RQ3, we have obtained grading statistics from our study administration on the past 10-years of exams. Prior to introducing the module, we did not explicitly discuss generative AI in a programming context with the students. This was a deliberate choice grounded in research questions RQ1 and RQ2, as we wanted to investigate to what extent students adopt AI by themselves (RQ1), and what

reflections they had formed on their own on the challenges of adopting AI in their learning activities.

The rest of this paper is organised as follows. Section 2 provides an overview of the content of the coding co-pilot module. Section 3 presents the results from the student surveys regarding prior use of AI (RQ1) and its challenges (RQ2). In Sect. 4 we analyse students grades from the past 10-years with a view towards assessing the impact of generative AI technologies (RQ3). Section 5 contains a discussion of related work. Finally, in Sect. 6 we sum up our conclusions.

2 The Coding Co-Pilot Module

The module consisted of a two-hour lecture given in the middle of the semester followed by a programming assignment where the students were encouraged to explore the use of co-pilots when solving the assignment. In total, the semester spanned 14 weeks of teaching.

In order not to discriminate any students up front, we did not assume any prior knowledge of generative AI and co-pilots. The two-hour lecture therefore started with a short introduction to generative AI, its foundation and capabilities. This included an overview of the history of AI in order to put the emergence of generative AI into a broader computer science historical context. This was followed up by a survey among the students on their prior use of generative AI and co-pilots in the course. The survey and results will be presented in Sect. 3. The first part was concluded with an explicit articulation of the main aim of the module in terms of discussing and having the students reflect upon the impact of generative AI in their software development education and on the profession of being a software developer.

The second part consisted of a technical tutorial on the use of co-pilots for programming. The goal was to demonstrate on concrete code examples, the many facets of programming where co-pilots are applicable. Specifically, we covered code generation from textual requirements, code completion, test-case generation, code review, refactoring, documentation, code translation, and debugging. We also demonstrated how co-pilots may be used to explain code and programming concepts as this is an area where co-pilots may help students by being a form of virtual labassistent to enhance their learning experience. In addition to showcase the power of co-pilots, we also demonstrated shortcomings related to the above coding activities such as impreciseness leading to incorrect code, issues with code quality, knowledge limitations emerging from training sets, non-determinism, and code ethics.

For the technical part, we used the Copilot4Eclipse plug-in [5] which is based on GitHub Copilot [6] - both the in-editor support as shown in Fig. 1 (left) and the chat support shown in Fig. 1(right). The reason for choosing Copilot4Eclipse was that up until 2024, we have been using Eclipse as the IDE in the course. Clearly, any other IDE with co-pilot support could have been used, and it would also have been possible to rely on stand-alone generate AI technologies such as

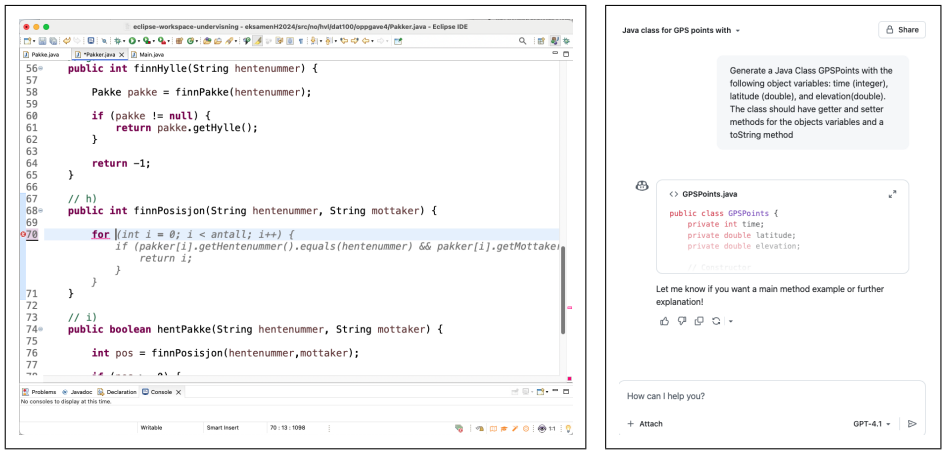


Fig. 1: Co-pilot plug-in for Eclipse: In-editor support (left) and chat (right)

ChatGPT directly or platform-based co-pilots such as the chat co-pilot of the GitHub platform.

The second part ended with a presentation of what students must master in order to effectively use co-pilots. This part included some emphasis on prompt engineering and the impact of prompt quality on the results obtained with co-pilots. The second part of the two-hour lecture consisted of a student survey with questions related to RQ2 on reflections and challenges that students see in adopting co-pilots.

Following the two-hour lecture, a mandatory assignment was introduced where the students were encouraged to explore the use of a co-pilot for solving the assignment. Following the deadline for handing in the programming assignment, we conducted a survey on how the students had used co-pilots in the assignment. The results from the survey at the end of the second part and the post-assignment survey will be presented in Sect. 3.

3 Student Adoption and Use of Co-pilots

We now present the results of the two surveys that were undertaken during the two-hour lecture, and the survey undertaken following the programming assignment where the students were encouraged to explore the use of co-pilots. A total of 74 students participated in the two in-lecture surveys while a total of 156 students participated in the post-assignment survey associated with the mandatory programming assignment.

Pre-adoption of co-pilots. The purpose of the initial survey was to obtain information on the extent to which the students had already started (on their own initiative) to use co-pilots and generative AI before it was being introduced in

Table 1: Questions and results on prior knowledge and pre-adoption of AI

Q1 Assess your level of knowledge on AI			
significant	some	little	no knowledge
6 %	57 %	34 %	3%
Q2 To what extent are you using AI technologies in the course?			
a lot	occasionally	little	not at all
31 %	44 %	12 %	13 %
Q3 Which AI technologies are you using in the course?			
Q4 What are you using AI technologies for in the course?			

the course module. The survey consisted of the four questions listed in Table 1. For questions Q1 and Q2 where the students had to choose between four alternatives, we have listed the distribution of answers in percentage below the question. For questions Q3 and Q4, we summarise the responses below.

From Q1 it can be seen that approximately 60 % of the students perceive that they have some or significant knowledge of AI technologies already. It is also noteworthy that approximately 30 % of the students were already using AI a lot in the course, and a little less than half using it occasionally. This shows that even not having touch upon the topic of co-pilots in the course until the point of the co-pilot module, students by themselves have picked up and started using the technology on their own initiative. For Q3, the vast majority of students using AI technology were using ChatGPT or GitHub co-pilot. A few students were using other alternative AI technologies such as Gemini, Claude, and AnswerAI. The free text responses from the students on Q4 indicated that primary use were for explaining code, debugging, and obtaining hints for solving a programming exercise on which they were stuck.

Reflections on the use of co-pilots. At the end of the co-pilot lecture, we conducted a survey intended to have the students reflect on their use of co-pilots. The first set of questions consisted of the following four statements S1-S4, where students were asked to indicate on a scale from 1 (Strongly Disagree) to 5 (Strongly Agree) the extent to which they agreed with the statement:

S1 Use of AI can improve the learning outcome in the course.
S2 I am motivated to learn more about the use of AI in programming.
S3 AI will play an important role in the future in courses on programming.
S4 I am confident in my ability to learn and use AI tools for programming.

The results are shown in Fig. 2, with a circle indicating the average score and the dimmed curve below each question indicating the distribution of answers. The results clearly indicate that students see a clear potential in the use of AI technologies, and that they also consider themselves capable of being able to learn how to use it.

Fig. 2: Student feedback on statements concerning role of AI

Table 2: Students assessment on potential usefulness (Lecture column) and application (Assignment column) of co-pilot functionality

Functionality	Lecture	Assignment
Code generation	14 %	12 %
Code completion	6 %	9 %
Test case generation	6 %	3 %
Code review	23 %	19 %
Refactoring	4 %	4 %
Documentation	3 %	1 %
Code explanation	19 %	16 %
Debugging	15 %	12 %
Concept explanation	10 %	10 %
Code translation	5 %	3 %
Other functionality	0 %	9 %

The next set of questions considered the use of co-pilots. Here we asked them based upon what was demonstrated in the lecture what capabilities of co-pilots they considered most useful. From the provided options, the students could choose multiple alternatives. The *Lecture response* column of Table 2 lists the relative distribution of answers. Due to rounding errors, the total of each column does not add up to exactly 100 %. We discuss the *Assignment* column of Table 2 in the next subsection. From the *Lecture* column it can be seen that code generation, code review, code explanation, and debugging receives the highest score, closely followed by concept explanation. It is interesting that code generation (from requirements) obtains a high score as this is a functionality that challenges introductory programming courses as these are courses where students traditionally are supposed to learn and are being assessed on how to write very basic code from requirements.

As the final question of the survey we asked the students using an open-ended question about the challenges they see in using AI technologies in the course.

Here, the students clearly expresssed a concern that it will negatively affect the learning outcomes when it comes to having knowledge and programming skills related to the basic concepts of programming. Many students also indicated that it may negatively impact critical thinking when it comes to code quality. Several students, however, also stated that when used properly as a learning assistant, it may in fact increase the speed of learning as it is always available. Overall the student answers to the open-ended question clearly showed that they are aware and reflect upon the pitfalls of blindly using AI technologies when it comes to learning outcomes.

Assignment use of Co-pilots. For the mandatory assignment that followed the lecture we encouraged the students to explore the use of co-pilots for solving the assignment. We followed this up with a survey on how they had used co-pilots after the deadline of the assignment. Approximately 50 % of the students chose to use co-pilots for the assignment. About 30 % indicated that the main reason for not using a co-pilot was that they were either not able to install it or found it too difficult to use. The remaining 20 % deliberately chose not to use co-pilots. That there were 50 % who did not use a co-pilot highlights the importance of not creating an AI divide in computer science education with some students mastering the technology and others not. This is in particular important in light of the increasing industry uptake and hence possible expectations from industry of being able to apply AI technologies for software development.

The detailed distribution of what the students were using co-pilots for when solving the programming assignment is shown in the *Assignment* column of Table 2. The results are largely consistent with the student responses in the *Lecture* column. It is interesting to observe that code completion scores relatively low. This may be due to the fact that this happens fully automatically in the IDE when writing code. Hence, students may not consider this to be AI-based as it does not involve any form of prompting and as such appears transparent.

The final question of the survey related to the learning outcome perceived by the students that had used co-pilots. Here those students that chose to use co-pilots for solving the assignment were asked to assess their learning outcome compared to not having used co-pilots. They were to choose between three alternative: *reduced learning outcome*, *identical learning outcome*, and *increased learning outcome*. Here the perception of 53 % of the students were that they had an increased learning outcome when using co-pilots with only 31 % perceiving a reduced learning outcome. This somewhat in contrast to common statements of AI technologies reducing learning outcomes.

4 Impact on Exam Performance

A main concern raised when it comes to the use of co-pilots in introductory programming education is the potential negative impact on learning outcome. Given that the learning outcomes of students in our course are assessed in a closed book written exam, it of relevance to investigate whether any impact on

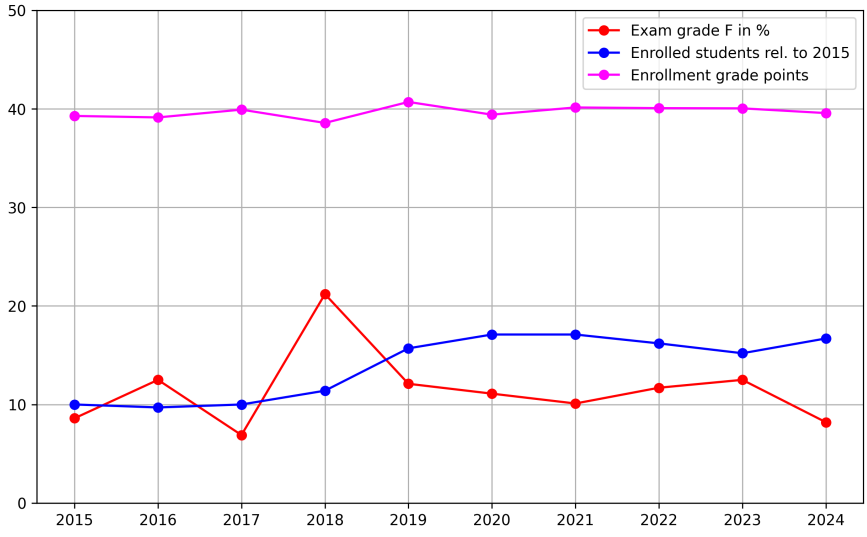
student exam performance can be observed following the widespread emergence of generative AI in recent years. We have seen from our survey that a significant fraction of the students already adopted the use of co-pilots in the autumn of 2024. We suspect this to have been the case also in 2023 and possibly in 2022.

When analysing the impact of AI on exam performance, we need to also consider other major influential developments in recent years. One is the implementation of Kunnskapsløftet in 2020 [10] which has made programming mandatory in secondary school. This implies that from 2023, students enrolled in our course potentially have a stronger programming background than earlier. A second one is the covid-pandemic in 2020 which implied a substantial amount of digital lectures and labs, and an open book home exam. A third factor is the grade point requirements for student enrollment which should be seen in conjunction with the number of enrolled students.

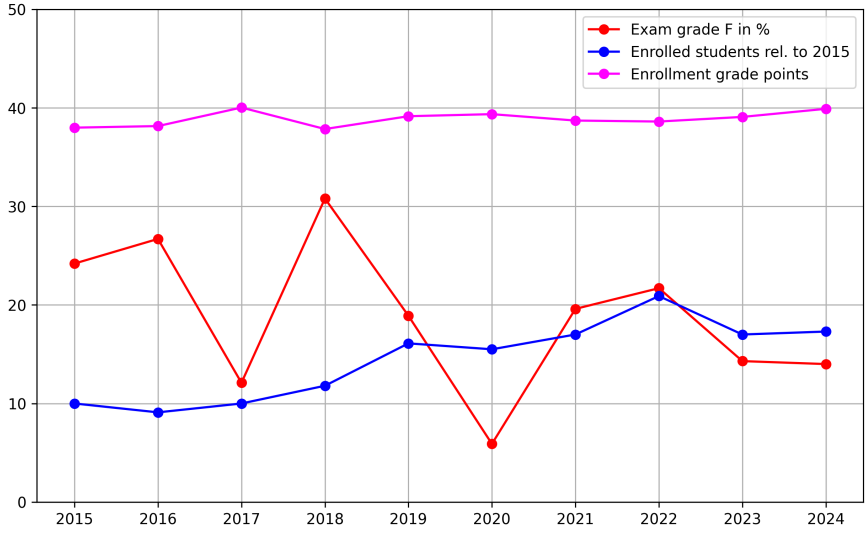
We have extracted information on student exam performance in our course from 2015 to 2024. In general, the topics of the course has been fairly stable in the past 10 years. Also, the type of exam questions and level of difficulty have been stable in the period (except for the 2020-covid exam) providing a reasonable basis on which to compare student performance. The form of teaching changed somewhat following the covid-pandemic in 2020 and 2021 where teaching was synchronous and digital, but where we in addition produced videos on the core topics. These have been made available to the students in the subsequent years. The course is taught jointly for students enrolled in the bachelor programme in software engineering and the bachelor programme in information technology. As the two study programs have different enrollment criteria (R1, R2, and Physics vs. R1 or S1+S2), we have separated them in the presentation.

Figure 3 shows the overall performance in the past 10 years in terms of percentage of students not passing the exam, number of students enrolled relative to 2015, and the average grade points for enrollment in the programme of study. Figure 4 shows the average grade (A = 5, F = 0) on the final exam for the two programs. The enrollment grade points originates from the database for higher education in Norway [2].

It can be seen that the number of enrolled students has increased since 2015, but without any significant impact on the average enrollment grade points. Information technology have in general had a slightly lower grade point average compared to software engineering as shown in Fig. 3, and also in general a higher fraction of students not passing the exam. Furthermore, from Fig. 4 it can be seen that information technology in general scores lower on average at the final exam. The 2018 exam appears to be an outlier for both study programs with a relatively high fraction of students failing the exam, and it seems correlated with the lower enrollment grade points in that particular year. We do not observe any significant negative impact since 2022 where AI technology became widely available, but we do see a small improvement in average grade and fraction of non-passing students for 2024. In terms of failing students and average grade, we cannot for 2023 observe any positive impact of programming being introduced in secondary school in 2020, unless a negative impact of AI is some-



(a) Software engineering



(b) Information technology

Fig. 3: Overall statistics for the two study programs

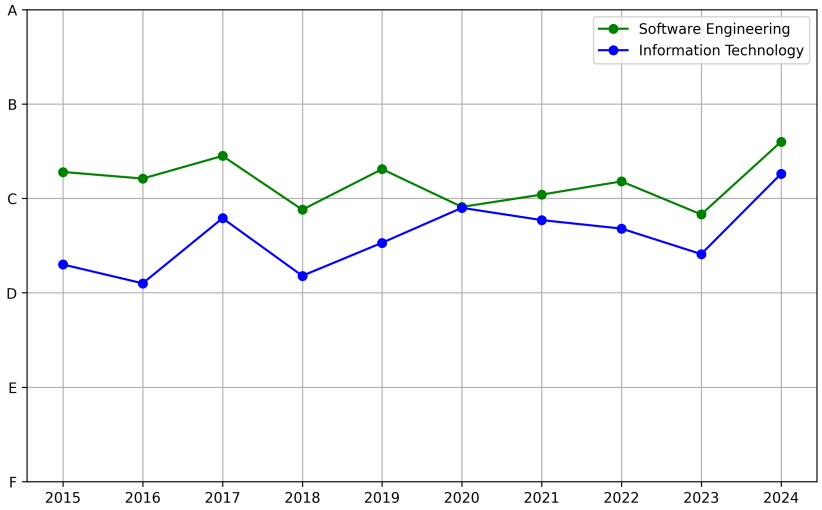


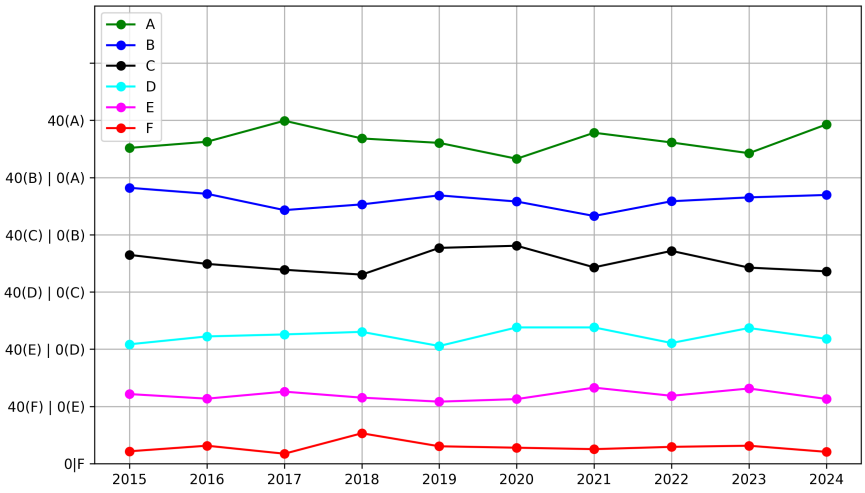
Fig. 4: Average grade on the final closed book exam

how counter-balanced by the introduction of programming in secondary school (or vice versa). This is somewhat in contrast to [1] which found that students enrolled since 2023 had a stronger programming background than others. The reason may be that the results in [1] are based on the Norwegian prior knowledge test in programming which was conducted in the beginning of the first semester, and not (unlike our results) based on the performance of the students at the final exam. For 2024 we observe an improvement (Fig. 4) on the average performance of the students.

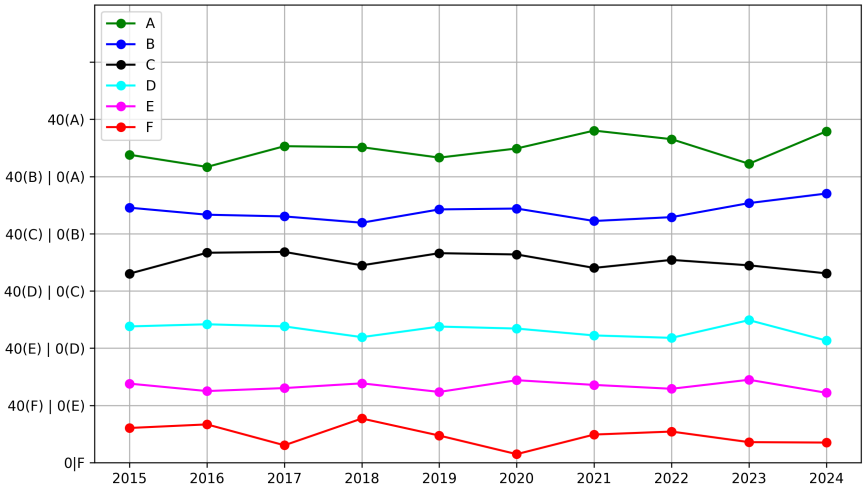
To investigate the exam performance in further detail, we have analysed the distribution of grades (A-F) for the two programs. This is shown in Fig. 5. We consider only students taking the exam for the first time. Each row between the grid-lines corresponds to a grade and development in the fraction (which are all between 0 % and 40 %) of students obtaining the corresponding grade. The grade for a given row is indicated by a label on the baseline on the form 0(X) where X is the grade. Again, we do not observe any consistent negative impact by the widespread emergence of ChatGPT in 2022. In contrast, we observe for 2024 an increase in A and B grades, and a decrease in C, D, E, and F grades which indicates an improved student performance.

5 Discussion

Generative AI and its implications for programming education [4] have become a prominent topic within educational research in recent years both from a student-centric, an educator-centric, and a technology perspective. In this paper, we have primarily taken a student-centric perspective focussing on students adoption and



(a) Software engineering



(b) Information technology

Fig. 5: Grade distribution for the two study programs

perception of AI and its impact on final exam performance. In contrast to our work, Osadcha et. al [9] studies AI from the perspective of educators and the challenges faced in adopting AI for their professional work.

In our work, we have focussed on overall student performance in terms of the final exam. We have not considered the mandatory assignments that students undertake in our course and which must be approved in order to enroll for the final exam. This aspect was investigated in [17] considering the different kinds of assignments and exams, their associated threats to integrity, and the role of AI in relation to the identified threats. Such threats are particularly imminent for courses where the assignments are being assessed and forms a basis for the final grade [12, 15]. In our co-pilot module, we focus partly on the AI technology and its use and partly on encouraging the student to think critically on their use of AI. This is in line with [16] addressing the topic of student learning in the presence of machine learning and what the fundamental differences are.

We have been using coding co-pilots directly as they are being provided as plug-ins to an IDE. As such, we use the co-pilots in a straightforward manner, where others [13] have built specific teaching material to promote responsible use, and have built teaching technology on top of generative AI to create, e.g., personalised assignments [3, 8]. Examples of this line of research also includes the use of AI to support flipped classrooms [20] and tutoring to support the learning of programming concepts [19]. Many of these approaches are highly relevant to consider also in the context of our course.

6 Conclusions and Future Work

Our survey results on pre-adoption of AI by students (research question 1) showed that a significant fraction of the students is picking up AI on their own initiative. This confirms our underlying hypothesis of embracing the technology, and the importance of covering the topic of co-pilots in an introductory programming course - in particular to encourage a responsible and sustainable use. When it comes to the challenges of using AI (research question 2) the students considered it of high importance to adopt and learn to use co-pilots. At the same time, their survey responses clearly indicate that they were well aware of the risks imposed on the learning outcomes. While the wider impact of generative AI and its potentially disruptive influence on software development is still unfolding, our analysis of the exam performance of students (research question 3) over the past 10 years does not show any clear negative impact of generative AI on the learning outcome. We conjecture that the final exam of the course being a closed book exam combined with the mini-exam organised towards the end of the semester may well be a key influencing factor on student learning behavior in terms of how they choose to use AI. In particular, the mini-exam makes it very explicit to the students what is expected and what is available in terms of aids for the final exam.

Overall we find that the results obtained with our initial introduction of co-pilots in 2024 are encouraging, and we plan to also include the topic of coding

co-pilots in 2025. Based upon the results obtained for research question 1, we plan to split the co-pilot module in two parts and have the initial introduction to the topic earlier in the course as our results indicated that students were anyway picking up the technology. This suggests the relevance of introducing it earlier. While we cannot identify a negative impact on student learning outcome by the emergence of co-pilots, it is premature to conclude that it had a positive impact even if we saw an improvement in exam performance by the students in 2024. The improved performance may also be a result on the combination of AI and programming having been introduced in secondary school in 2020.

Hence, we need to track the exam performance in the coming years to have more conclusive evidence on the impact of AI on learning outcomes. Another highly important aspect of future work is how to follow up a basic introduction to coding co-pilots in software development courses in subsequent semesters where more advanced use of AI may well be of relevance. From the overall perspective of student learning outcomes measured based on performance at the final exam, our results do not suggest that the introduction coding co-pilots somehow would represent the end of teaching an introductory programming course.

References

1. Bolland, S.: National Prior Knowledge Test in Programming - How Proficient are Incoming Higher Education Students? Proc. of Norsk konferanse for utdanning og didaktikk i IT-fagene (Nov 2023), https://www.ntnu.no/ojs/index.php/nikt/article/view/5785/5206
2. DBH: Database for statistikk om høyere utdanning (2025), https://dbh.hkdir.no/tall-og-statistikk/statistikk-meny/studenter, [Accessed: 16-September-2025]
3. Fayaz, A., Glassey, R., Baltatzis, A.: Generating Personalized Assignments with Students in the Loop. In: Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. p. 305–311. ITiCSE 2025, ACM (2025). https://doi.org/10.1145/3724363.3729070
4. Finnie-Ansley, J., Denny, P., Becker, B.A., Luxton-Reilly, A., Prather, J.: The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In: Proc. of the 24th Australasian Computing Education Conference. p. 10–19. ACE '22, ACM (2022). https://doi.org/10.1145/3511861.3511863
5. Genuitec, LLC: Copilot4Eclipse – GitHub Copilot Integration for Eclipse (2025), https://marketplace.eclipse.org/content/copilot4eclipse, [Accessed: 16-September-2025]
6. GitHub: GitHub Copilot. https://github.com/features/copilot (2025), [Accessed: 16-September-2025]
7. Høgskulen på Vestlandet: DAT100 – Grunnleggende programmering (2025), https://www.hvl.no/studier/studieprogram/emne/dat100, [Accessed: 16-September-2025]
8. Jacobs, S., Peters, H., Jaschke, S., Kiesler, N.: Unlimited Practice Opportunities: Automated Generation of Comprehensive, Personalized Programming Tasks. In: Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. p. 319–325. ITiCSE 2025, ACM (2025). https://doi.org/10.1145/3724363.3729089

9. Kateryna Osadcha, J.S., Chishti, M.S.: Using Microsoft Copilot Chat in the Work of IT Educators: Pilot Study. Proc. of Norsk konferanse for utdanning og didaktikk i IT-fagene (Nov 2024), https://www.ntnu.no/ojs/index.php/nikt/article/view/6202/5621
10. Kunnskapsdepartementet: Fornyer innholdet i skolen (2018), https://www.regjeringen.no/no/dokumentarkiv/regjeringen-solberg/aktuelt-regjeringen-solberg/kd/pressemeldinger/2018/fornyer-innholdet-i-skolen/id2606028/?expand=factbox2606064, [Accessed: 16-September-2025]
11. Lau, S., Guo, P.: From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In: Proc. of the 2023 ACM Conference on International Computing Education Research - Volume 1. p. 106–121. ICER '23, ACM (2023). https://doi.org/10.1145/3568813.3600138
12. Liu, Z., Tang, Y., Luo, X., Zhou, Y., Zhang, L.F.: No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT. IEEE Transactions on Software Engineering **50**(6), 1548–1584 (2024). https://doi.org/10.1109/TSE.2024.3392499
13. MacNeil, S., Prather, J., Nabid, R.A., Gutierrez, S., Carvalho, S., Shrestha, S., Denny, P., Reeves, B.N., Leinonen, J., Rossetti, R.L.: Fostering Responsible AI Use Through Negative Expertise: A Contextualized Autocompletion Quiz. In: Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. p. 326–332. ITiCSE 2025, ACM (2025). https://doi.org/10.1145/3724363.3729067
14. Nicolajsen, S., Brabrand, C.: What Is Programming? Commun. ACM **68**(6), 28–30 (Jun 2025). https://doi.org/10.1145/3713068, https://doi.org/10.1145/3713068
15. Ouh, E.L., Gan, B.K.S., Jin Shim, K., Wlodkowski, S.: ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course. In: Proc. of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. p. 54–60. ITiCSE 2023, ACM (2023). https://doi.org/10.1145/3587102.3588794
16. Schaathun, H.G.: Maskinlæring og studentlæring. Proc. Norsk IKT-konferanse for forskning og utdanning (Dec 2022), https://www.ntnu.no/ojs/index.php/nikt/article/view/5439/4914
17. Sindre, G.: AI Technology: Threats and Opportunities for Assessment Integrity in Introductory Programming. Proc. of Norsk konferanse for utdanning og didaktikk i IT-fagene (Nov 2023), https://www.ntnu.no/ojs/index.php/nikt/article/view/5710/5143
18. Stack Overflow: 2024 Stack Overflow Developer Survey (2024), https://survey.stackoverflow.co/2024/, accessed: 2025-09-16
19. Stienstra, L., Mohamed, A., Mohamed, M.: Exploring GenAI as a Tutoring Tool: A Case Study in First-Year Computer Programming. In: Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. p. 452–457. ITiCSE 2025, ACM (2025). https://doi.org/10.1145/3724363.3729060
20. Zhang, Y., Ouh, E.L., Ho, A., Lo, S.L., Tan, K.W., Lin, F.: PromptTutor: Effects of an LLM-Based Chatbot on Learning Outcomes and Motivation in Flipped Classrooms. In: Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1. p. 445–451. ITiCSE 2025, ACM (2025). https://doi.org/10.1145/3724363.3729095