This is the Accepted manuscript version of the following article:

D. Palacios-Alonso, J. Urquiza-Fuentes, J. Á. Velázquez-Iturbide and J. Guillén-García, "Experiences and Proposals of Use of Generative AI in Advanced Software Courses," *2024 IEEE Global Engineering Education Conference (EDUCON)*, Kos Island, Greece, 2024, pp. 1-10, doi: 10.1109/EDUCON60312.2024.10578869.

DOI: https://doi.org/10.1109/EDUCON60312.2024.10578869

Experiences and Proposals of Use of Generative AI in Advanced Software Courses

Daniel Palacios-Alonso
Department of Computing and Statistics
Universidad Rey Juan Carlos
Madrid, Spain
daniel.palacios@urjc.es

Jaime Urquiza-Fuentes
Department of Computing and Statistics
Universidad Rey Juan Carlos
Madrid, Spain
jaime.urquiza@urjc.es

J. Ángel Velázquez-Iturbide
Department of Computing and Statistics
Universidad Rey Juan Carlos
Madrid, Spain
angel.velazquez@urjc.es

Julio Guillén-García
Department of Computing and Statistics
Universidad Rey Juan Carlos
Madrid, Spain
julio.guillen@urjc.es

***Abstract*— the last year, we have witnessed the popularization of generative artificial intelligence. Its output includes text, code, image, audio, speech, voice, music, and video. Therefore, it impacts education courses where students are required to elaborate on any of these artifacts. In particular, the generation of code affects informatics courses, where assignments usually ask students to develop and deliver programming code. The impact of generative artificial intelligence on informatics courses has been mainly studied for introductory programming courses. These studies have shown that generative artificial intelligence is able to produce highly sophisticated programs, but also that its results and rationale can be inaccurate. Moreover, the impact of generative artificial intelligence has not been studied for other informatics subjects.**

In this paper, we present our preliminary experience and proposals on three advanced software courses, namely video games, advanced algorithms and language processors. For the video games course, we present the opportunities of use of generative artificial intelligence and the results of a survey conducted with students on their use to obtain different media products. For the algorithms course, we present the result of a session driven by the instructor on different design techniques, showing the merits and demerits of the answers generated. For the language processors course, a proposal of use of generative artificial intelligence is presented, broken down into the parts of a typical language processor. The paper concludes with some suggestions for instructors.

Keywords*— *informatics education, generative artificial intelligence, video games, advanced algorithms, language processors

I. INTRODUCTION

The field of artificial intelligence (AI) comprises different technologies that have been applied to education. Their uses also are multiple:

- Intelligent tutors. They provide assistance and personalized teaching to individual students.
- Predictive AI. They analyze students' interactions or grades, and try to identify patterns of behavior that are symptoms of their past or future performance, e.g., plagiarism or drop out.
- Other uses, such as augmenting grading efficiency and consistency, or assisting in administrative tasks (e.g., scheduling).

In the last year, we have witnessed the popularization of another AI technology, generative artificial intelligence (GenAI for short). These tools are trained on massive amounts of data to recognize patterns and relationships, which they use to generate outputs customized to the users' prompts. Examples of tasks these tools perform with success [1] are: answering questions, improving and summarizing text, writing essays, translating text from one language to another, generating ideas or suggestions for a given topic, and generating text with specific attributes, such as tone, sentiment, or formality.

Actually, GenAI output is not limited to text, but it may also generate code, image, audio, speech, voice, music, and video. Consequently, GenAI is impacting all areas of content generation. In particular, GenAI also is impacting informatics, as it is a discipline which heavily relies on writing code.

Some authors have predicted the end of programming as we have known it for decades [2]. They envision that programmers will be replaced by specification writers outlining to GenAI tools what code they want in natural language and obtaining the code. However, other authors are not so optimistic, arguing that GenAI will assist programmers to be more productive in some areas but not in others [3]. In particular, GenAI seems to be successful in generating pieces of code, i.e., "snippets".

GenAI also is impacting the education sector more profoundly than previous AI tools. GenAI can be used to generate different kinds of artifacts, to answer questions and to complete written tasks. It can also respond to prompts in a human-like way. GenAI can be used to solve assignments which involve these activities, thus raising issues of plagiarism. There is no study on which disciplines are more threatened, but the risk is obvious in disciplines where assignments often include essays (e.g., history) or media artifacts (e.g., arts).

Obviously, the risk affects informatics education, where different software artifacts (e.g., algorithms or database queries) are usually required in assignments. Certain forms of low-code programming [4] have been adopted in the last decades for programming education, especially block-based languages [5]. However, the risk of GenAI comes from the fact that it may not require the programmer to produce any code at all, just to state a specification of the intended code behavior, which usually is the assignment statement specified by the instructor.

The paper presents several experiences and proposals of the authors on advanced software education at the university. The authors are not aware of similar papers regarding the education of informatics subjects other than programming. Notable exceptions are Shoufan [6] and Pérez-Colado *et al.* [7]. The first research work focused on ChatGPT's ability to answer test questions on digital circuits and computer

This work was supported by a research grant to the LITE research group of the Universidad Rey Juan Carlos (ref. M3286) and the project grant of Student Observatory of the Universidad Rey Juan Carlos (ref. 2023/OBSERC-37674).

architecture. The authors of the second research demonstrate how AI techniques can streamline the prototyping of serious games by automating tasks, generating personalized content, and optimizing the creative process, allowing developers to focus on more strategic and creative aspects of game design.

The structure of the paper is as follows. In the following section, we present a summary of the impact of GenAI on education. The third section presents experiences and proposals at three courses of the authors' university, namely advanced algorithms, video games, and language processors. The fourth section discusses the issues raised by these experiences and proposals. Finally, we summarize our conclusions.

II. GenAI in Informatics Education

In this section we introduce the use of GenAI in education in general, and in programming education in particular.

A. GenAI in Education

As any digital technology, GenAI can provide both benefits and damage [8]. Thus, GenAI can help students practice and improve their language skills, e.g., by encouraging interaction in natural language or by supporting language translation. They can also adapt learning to the student's level of knowledge and may assist some minorities, such as students with disabilities. Even teachers may benefit, e.g., by assisting either in automated essay grading or by generating sample lesson plans, learning objectives, and instructional activities (which they must later review and refine).

GenAI also has limitations. Thus, its outcomes may be inaccurate, inappropriate, outdated, or biased because they are not based on an understanding but on the patterns found in the data used for training. It also raises issues of academic integrity, because students may deliver outcomes of GenAI tools without proper attribution.

Educational authorities and institutions have reacted in several ways. Some educational institutions have banned the use of GenAI, while others have not reacted at all or other governments and organizations are playing an active role in informing with warnings and recommendations on the educational use of GenAI.

The British Government [9] acknowledges opportunities to reduce teachers' workload. However, it also warns on several risks, such as the use of personal data or original products (subject to the students' intellectual property) to train AI models. Students should also be aware that the contents generated may be inaccurate, inappropriate or biased. Consequently, they should develop skills to use GenAI in a critical way.

The Joint Council for Qualifications (JCQ) has published guidance on AI use in assessments [1] to support teachers and exam centers to know how to protect the integrity of qualifications and what counts as AI misuse. The JCQ definition of AI misuse is clear [1]:

"AI tools must only be used when the conditions of the assessment permit the use of the internet and where the student is able to demonstrate that the final submission is the product of their own independent work and independent thinking."

JCQ provides examples of AI misuse, which consider any form of copying, paraphrasing or failure to acknowledge use of AI tools. In addition, JCQ advocates for responsible use of AI tools, making students aware of AI misuse and making centers and teachers aware of their responsibility in defining policies and in informing students on appropriate use and of risks of using AI.

Several organizations, coordinated by Code.org, has launched the TeachAI initiative, which is intended to provide guidance on the educational use of AI. The initiative has elaborated an open document [10] with guidelines to address AI in education. The "toolkit" identifies seven principles for AI use, including reaffirming adherence to existing educational policies, promoting AI literacy, and balancing the benefits and risks of AI.

B. GenAI in Programming Education

GenAI can be used to generate computer programs. Probably, the two most popular GenAI programming tools are ChatGPT and Github's Copilot, but many other tools exist: Alphacode, AI Code Reviewer, AI Data Sidekick, and Figstack. Typically, they generate code for a problem statement written in natural language, but they can also perform other programming tasks, such as explain program code or translate code between programming languages.

The potential of GenAI for generating programs has led to a profound concern within the informatics community, even suggesting that the very nature of informatics and its curricula should be revised [2]. Different experiences and research efforts have been conducted. We just summarize a representative sample.

Some studies focus on the success of GenAI in solving programming problems. Tran et al. [11] experimented with a large collection of exercises extracted from the Kattis online judge system. The problems selected were evenly rated with varying degree of difficulty, and their statement was verbose, including social and cultural background contexts. Their results allow concluding that success of ChatGPT solving problems was inversely related to both the perceived problem difficulty and the richness of the vocabulary used.

Puryear and Sprint [12] assumed that students will use GenAI for programming tasks, thus they inquired into the quality of the programs generated by Copilot. They found that Copilot generates code that can solve introductory assignments with human-graded scores ranging from 68% to 95%. In addition, generated code was generally not similar enough to students' code to suggest plagiarism. The authors made several educational suggestions based on these results, including promoting students' training on GenAI programming tools, teaching debugging and testing techniques, stating unfamiliar assignment statements, and promoting assessment methods that guarantee students' understanding of their solutions.

Other studies have found similar results, with some nuances. Thus, Ouh et al. [13] found that ChatGPT does not help much when there are complex instructions (e.g., producing a sophisticated representation as output), or when the exercises require students to interpret API documentation and UML diagrams. Similar difficulties (such as poor handling of exercises requiring complex chains of reasoning steps) have been reported by other authors [14]. Mixed results have also been obtained when ChatGPT has been prompted to check, critique and provide suggestions on students' submissions [15].

Some studies have focused on students' experience as programmers in use of GenAI tools. Prather et al. [16] studied the interactions and the perception of novices in an undergraduate introductory programming course who used Copilot for their first time. They observed two novel interaction patterns (complementary to patterns documented in the scientific literature for professional programmers), explored the ethical implications of the results, and presented several design implications for GenAI to be used in programming education.

There has been extensive research in the field of computing education on the relationship between a student's ability to explain code and other skills such as writing and tracing code. In particular, the ability to describe at a high-level of abstraction how code will behave over all possible inputs correlates strongly with code writing skills. Leinonen et al. [17] have explored the potential of GenAI in generating explanations that can serve as examples to scaffold students' ability to understand and explain code. They found that GenAI created explanations being significantly easier to understand and more accurate summaries of code than student-created explanations. They argue that these tools can be useful for students who are practicing code reading and explaining.

Adopting a different approach, Lehtinen [18] asked questions to ChatGPT about code generated by the system itself. The system responded to the questions better than the average novice, but it also lapsed into human-like errors producing failed reasoning about the code.

III. Experiences and Proposals

In this section, we present our initial experience using GenAI in three courses. First, we describe the results of a survey conducted with students of a video games project course on their actual use of GenAI for their project. Second, we report on the result of an experience conducted by the instructor of an advanced algorithms course to check the quality of ChatGPT answers to prompts mirroring the course assignments. Thirdly, we present a proposal of integration of GenAI in the different parts of an intensive software course, namely language processors. All the contributions are in the context of informatics degrees at the Universidad Rey Juan Carlos.

A. Use in a Video Games Project Course

The course "Games for Web and Social Networks" is a fourth-year course of the Degree in Video Game Design and Development with 80 students enrolled. The course focuses on the completion of a full video game in just fourteen weeks (the duration of the subject) by each of the "companies" or students' teams. Video games start from their most incipient idea to the development of a postmortem document once the project is finished. In addition, at the end of the course, students must present their game in a limited time (elevator pitch) with a completely professional style (like the one done in the PlayStation® Talents) in front of a multidisciplinary group of professionals in the sector. As a result of this innovative proposal, several of these groups have obtained notable results on social networks and through the feedback of the members of the expert tribunal who encouraged them to continue with the video games developed and to try to introduce them into the market. The high degree of student satisfaction with the teaching work carried out is noted.

The main motivation of this innovative practice is to bring together all the general and specific knowledge and skills acquired throughout the university career and expose a scenario as close as possible to the working world and more specifically, to the business, commercial and video game development sectors. To carry out this task, a substantial methodological change in the classroom has been necessary. This change lies in the recreation, in the form of a live role-playing game, of a video game design and development company. These companies will have to make their elaborations and try to simulate a business model that allows the members of this company to stay afloat for no less than two years in the real market.

The subject can be considered very demanding in terms of work hours and dedication on the part of the student. The development of the video game is carried out with an industrial framework of recognized prestige in the sector, Unity. The student can use all the aids provided by a professional tool.

The chosen methodology is based on six steps:

1) *Proposal of a realistic and competitive project*. Students must develop all the necessary infrastructure to establish themselves on the network of networks as an emerging video game design and development company. Therefore, they must choose a company name and register the following points: an email account, a Twitter account, an itch.io account, create a business website on GitHub and a YouTube channel.

2) *Creation of a realistic portfolio*. One of the most important concerns of students who reach the last year of their career is the fact of finishing the career without having made a complete videogame or, in other words, a creation of their own. The different courses focus on understanding algorithms, creating animations, designing agile methodologies, etc. However, all of them are disconnected and without a common thread that allows students to show everything learned so far. For this reason, when students finish this course, they have a portfolio that demonstrates the skills acquired and two tangible and saleable products made by themselves.

3) *Attractive and competitive rubrics*. An essential part of this project is to keep the student "hooked" at all times. One way to achieve this is to present an evaluation method that provides a grade on the team's effort. Given that the subject is demanding, if the project meets the minimum requirements, the team will have achieved a passing grade. However, if the team performs extra tasks, that grade can gradually rise until obtaining the maximum grade.

4) *Undertaking a challenge*. The project has a triple evaluation method, that is, it is evaluated by the subject's teachers, the students themselves and an expert tribunal. This last evaluating group generates an extra level of motivation because the students know that they are going to be evaluated by people who know and live the world of video games first-hand. For this reason, students commit to the project and their colleagues to successfully carry out each of the creations.

5) *A finished game...is a treasure*. It must be understood that the video game industry, like consulting or software companies, requires solid evidence to be able to hire individuals who could become part of the staff. For this reason, they see this proposal as a springboard to show the

work done to possible “talent scouts” in the world of video games in Spain and abroad months later.

6) *A proposal full of experts*. To carry out this innovative proposal, professionals from the sector have been counted on, but most importantly, from different areas such as digital marketing / PR and social media expert, video game magazine editors, Indie video game developers, specialists in artificial intelligence and algorithms, lawyers specializing in patents and software.

Due to the high demand for work required of students in coding, 2D/3D animation, music, texturing, sound effects, documents and more arrangements necessary for a final version of a video game; we considered the use of GenAI for the development of projects in the 2023/24 course. It should be noted that the use of these tools was completely optional and students were encouraged to use them, making it clear that it would not raise or lower the final grade.

Once the course was finished and all the project deliveries were made, the teachers conducted a detailed survey aimed at obtaining information about the use of GenAI in the course.

The survey consisted of 15 questions, of which three were personal data and twelve were directed at the different GenAI tools used for tasks such such as coding, 2D/3D design, 2D/3D audio, documentation, and some personal questions about this new paradigm. 38 students completed the questionnaire (28 of them used AI and the rest did not), the most relevant questions and answers are presented below, the percentages correspond to the students who used AI in their projects:

1. *Have you used any GenAI to make any part of the JWRS project (however minimal)?* In this question, 73% of respondents said yes, they used GenAI in some part of the project, while 26% did not use it at all.
2. *Let’s start with generalist GenAI, which tools did you use? a) ChatGPT, b) Google Bard, c) Bing chat, and d) None*. Of the 73% of respondents who used GenAI, 87% used ChatGPT and 12% used Bing Chat. However, Google Bard was not used by any respondent.
3. *Regarding coding, which tools did you use? a) CodeGPT, b) CodeDaVinci, c) Code-Bard, d) GitHub Copilot, e) Code2Vec, and f) None.* The outcomes were quite clear when it came to the use of AI for programming tasks. Only three out of the six options were chosen. 25% of the students used CodeGPT, and 10% used the well-known, and somewhat controversial, GitHub Copilot. However, interestingly, 60% of those who did use GenAI did not use them for programming tasks. This suggests that while AI tools are being utilized in project development, their application in programming tasks is not as widespread.
4. *Regarding images and 2D design, which tools did you use? a) DALL-E 2, b) VQGAN+CLIP, c) GANPaint, d) None and e) Others.* In the survey, it was found that 26% of respondents have used this type of generative AI. Only a small portion of them, 13%, used DALL-E 2 and another 13% used other unspecified tools. Interestingly, the remaining 73% did not use this type of tools for their projects. This data provides an insight into the current usage trends of GenAI in project development. It seems that while some students are exploring these advanced tools, a significant majority have yet to incorporate them into their workflow.
5. *Regarding images and 3D design, which tools did you use? a) Midjourney, b) 3DFY AI, c) DeepMing Image, and d) None.* No respondent used any GenAI tool of this category for their projects.
6. *Regarding 2D audio, which tools did you use? a) Chordana, b) Magenta, c) Amper Music, and d) None.* No respondent used any GenAI tool of this category for their projects.
7. *Regarding 3D audio, which tools did you use? a) WaveNet, b) NVIDIA Omniverse Audio2Face, c) 3Dimensions, and d) None.* No respondent used any GenAI tool of this category for their projects.
8. *Regarding the Game Design Document (GDD), which tools did you use? a) OpenAI GPT-3, b) Google AI LAMDA, c) DeepMind AlphaFold, d) YanwenAI, e) Intelligent Gaming Labs, and f) None.* In this case, the response is dichotomous. 58% of the respondents used OpenAI GPT-3. The rest of the respondents did not use any GenAI. This indicates a clear preference for OpenAI GPT-3 among those who did use GenAI tools, while a significant portion chose not to use any such tools in this category.
9. *Finally, some more general questions. Have you relied heavily on these types of tools for the development of the video game? (be honest...it’s not negative at all).* 67% of the students did not rely heavily on the use of these types of tools, while the remaining 33% found them quite beneficial. This suggests a mixed level of engagement with GenAI tools among the students, with a significant portion finding value in their use for project development.
10. *How much time do you think using these types of tools has saved you? Try to quantify it in hours, please.* Here, the responses are multiple and varied. Some students suggest that they have saved more than 20 hours of work, while others point to a maximum of just one hour. This indicates a wide range of experiences with the use of GenAI tools, with the time-saving benefits varying significantly among the students.
11. *Do you think GenAI introduces uncertainty in the video game or computer sector?* 60% of the students were concerned about these types of tools, while the remaining 40% feel confident that their jobs are not at risk. This highlights a significant divide in perceptions about the impact of AI on job security, reflecting broader debates in society about the role of automation and AI on the future of work.

B. Use in An Advanced Algorithms Course

The course "Advanced Algorithms" is an elective course in the first quarter of the 4th year, offered to students of both the Degree in Informatics and the Degree of Computer Engineering. It relies on a previous, mandatory course of the second year, "Design and Analysis of Algorithms".

The "Advanced Algorithms" course addresses advanced topics of design techniques previously studied, as well as advanced techniques, new to the students. The advanced topics refer to the greedy and backtracking techniques, whereas the novel techniques include heuristic and approximation algorithms, branch-and-bound, dynamic programming, and probabilistic algorithms. Most techniques address optimization problems, which are implicitly complex and difficult to solve.

The emphasis of the course is not on coding but on the design decisions that lead to specific code and on the analysis of the algorithms. Students are given guidelines to solve algorithms for each design technique, usually either specific design decisions (e.g., a bounding function for a branch-and-bound algorithm), development methodologies (e.g., for dynamic programming) or code templates (e.g., for backtracking). Students also are given software tools that support the analysis of either recursive behavior (namely, the visualization system SRec [19]) or experimentation with optimality and time execution (namely, the benchmarking system AlgorEx [20]).

Assessment is solely based on six assignments, each one devoted to a different algorithm design technique. No assignment but the first one is limited to code. Instead, each assignment asks several of the following elements: a key design decisions specific of the design technique; code based on the design decisions; formal analysis of the algorithm time and space complexity; experimental analysis of either time performance, redundancy, or optimality. Four of the six assignments address the same optimization problem, so that students may better compare the different techniques. For each assignment, a report template is provided that students must adhere to, and where students must include explanations, program code, formulas, or visualizations exported as graphical files from the SRec and AlgorEx tools.

For the purpose of this paper, we decided to explore the performance of ChatGPT 3.5 on solving the problem used in the academic course 2023/24. The complete statement can be found in Kleinberg and Tardos' textbook [20, pp. 321-322], as exercise 10 of the dynamic programming chapter (not solved in the book). A summary of the problem statement is given in the two following paragraphs.

Assume you have to simulate a physical system for as many discrete steps as you can. There are two computers available, but the simulation job can only run on one of the computers in any given minute. Over each of the next n minutes, you have a prevision of how much processing time is available on each computer. Thus, in minute i, you would be able to run $a_i>0$ steps of the simulation if the job is run on machine A, and $b_i>0$ steps if it is run on computer B. You might also move the job from one computer to the other, but doing this costs one minute of time in which no processing is done on the job.

Given a sequence of n minutes, a plan is specified by a choice of A, B or "move" for each minute, with the property that A and B cannot appear in consecutive minutes. Thus, given values a_0, a_1, …, a_{n-1} and b_0, b_1, …, b_{n-1}, the problem consists in finding a plan of maximum value. An example was given with values for 4 minutes, and an optimal outcome equal to 37.

The author had not used ChatGPT in advance. Thus he registered and started a session, which started warning ChatGPT that he would like to speak in Spanish (to have the same experience as his students) so as to obtain algorithms in Java. The session included different prompts on assignments 2, 3, 5, and 6. They all were based on the same optimization problem, but each assignment required the students to solve it by using one or two different techniques. The session was not strictly lineal, as some techniques were addressed at several times. In summary, the session proceeded as follows, grouping interactions on the same technique:

- *Heuristic algorithms.* A greedy-like heuristic algorithm was asked, as well as its time and space complexity analysis. Additional heuristics were also asked, without coding them, as well a local search algorithm, which should be coded.
- *Backtracking*. A search tree to solve the problem was asked, as well as a backtracking algorithm based on the tree.
- *Branch and bound*. A bounding function was asked, as well as modifying the previous backtracking algorithm to integrate the bounding function. Additional bounding functions were also asked, without coding them.
- *Dynamic programming*. A recursive algorithm was asked, as well as a recursion tree and its corresponding dependency graph. Finally, an iterative algorithm was asked, as well as its time and space complexity analysis.
- *Probabilistic algorithms*. A probabilistic algorithm was asked.

The answers of ChatGPT were of varying quality. In some cases, they were completely correct, whereas in others they were partially or completely wrong, even with wrong answers persisting despite being prompted to fix them. The complete transcription and its analysis will be available as a technical report. We summarize here ChatGPT behavior for the different techniques:

- *Heuristic algorithms*. A heuristic algorithm was developed by ChatGPT. Its rationale was correct. The style was excellent, with comments embedded into the code, good selection of method and variable identifiers, indentation, etc.

  However, we noticed that the algorithm did not satisfy the constraints of the problem statement, as it computed simulation steps in all minutes, even in presence of a computer swap. We reported the error and ChatGPT acknowledged it. However, on the first prompt, it did not change it at all, and on the second one, it made a minor, irrelevant change.

  We asked a summary of the algorithm behavior and ChatGPT did it well, but it also included summaries of the problem statement (especially, the target function)

and its complexity analysis. On prompt, it again kept parts of the problem statement.

Regarding time and space complexity of the algorithm, ChatGPT delivered a brief reasoning and the final, correct order of complexity. However, when we asked to prove the number of iterations of the loop, it was too verbose, and the answer was unsatisfactory, as it never resorted to a mathematical reasoning, such as the use of a constant series. With respect to space complexity, it omitted the space occupied by the parameters, but it corrected the analysis on prompt.

Later, ChatGPT was asked to suggest a different heuristic. It provided one but, on noticing that the target function seemed to be different, it changed it into another one. In addition, we asked for more heuristics, and it outlined five additional heuristics. It was interesting to note that all heuristics were given a name. We demanded more precision on a detail of the fifth heuristic. It referred to "blocks" of time, but we asked the length of such blocks. ChatGPT gave twice a lengthy description of the experiment we should do determine the best length. Finally, we suggested that a block length equal to one seemed to be adequate for this problem; ChatGPT accepted that the suggestion *could* work, but it was not more concrete, including three additional diverging paragraphs.

Finally, a local search algorithm was demanded, and it provided a well-structured solution. We noticed again that the algorithm counted simulation steps on a move, and it changed it correctly. However, we warned again and again that a Boolean condition checked whether a 'onReassignment' variable was equal to 'R', while there was no statement which assigned such a value. ChatGPT always agreed that the researcher was right and changed the algorithm slightly, but the error was never fixed.

- *Backtracking*. Firstly, a design of a search tree was asked. ChatGPT explained a candidate search tree and included a text-based drawing of a tree for an own example, of length 2. Then we prompted it to draw the search tree for the example given in the problem statement. The search tree was badly constructed and, for three times, a source of error was successively pointed out to ChatGPT. First, the tree height had one level less than expected. Second, we warned that its answer claimed an optimal value not present in the tree. Third, we contributed the optimal sequence of decisions and its associated value for the given problem instance. However, ChatGPT was persistently unable to build the correct search tree.

  Later on, we asked to obtain the code of a backtracking based on the search tree. Surprisingly, the code generated three recursive calls per node, whereas the search tree only contained two children nodes. ChatGPT did not fix it despite of being warned of the mistake. Actually, its answer suggested that ChatGPT confused children nodes of the search tree with function arguments.

  Backtracking algorithms are often coded as recursive algorithms (with the if-else structure usual in recursive algorithms) with an inner loop (for the children of the current node). Horowitz et al. [22] propose a more efficient template for backtracking algorithms, which promotes the loop out from the if-else. We asked ChatGPT to convert its previous algorithm to this new format. The algorithm was successively refined along four prompts, resulting in an equivalent algorithm.

- *Branch and bound*. We successively asked whether a bounding function should compute an upper or a lower bound for the given problem, to define a bounding function, and to code a branch-and-bound algorithm that extended the previous backtracking algorithm with the newly defined bound. In all cases, the answer was correct, but too verbose. We asked to name the bounding function, and the answer was satisfactory.

  We also asked for additional bounding functions. ChatGPT contributed with three bounding functions, each one with a name. We noticed that one was probabilistic, thus it could compute values which were not an upper bound, and ChatGPT agreed. We asked a confirmation on the correctness of the two remaining upper bounds. ChatGPT confirmed and argued on its correctness. We noticed that they did not guarantee an upper bound, but we did not continue arguing.

- *Dynamic programming*. A recursive solution was asked, contributing ChatGPT with a backtracking algorithm. We asked a purely recursive algorithm, without accumulating parameters, and we then obtained an adequate algorithm. After three prompts, noting different mistakes, we obtained a correct algorithm.

  Given that ChatGPT had demonstrated capability to draw text-based trees, we asked the recursion tree corresponding to the example given in the problem statement. The tree had fewer nodes than expected, but ChatGPT was unable to correct it.

  We also asked to convert the last recursion tree into a dependency graph, obtaining an incorrect graph, with nodes not present in the tree and even one node duplicated.

  We asked a Java declaration of a table capable to store in a structured way the values of the graph nodes. ChatGPT. It twice provided a memoization algorithm which used an appropriate table. We also asked an iterative version, which was correct. Finally, we asked the time and space complexity analysis, which was right but again too verbose and without calculating the number of iterations in a formal way.

- *Probabilistic algorithms*. We asked a probabilistic algorithm, and it developed a well-designed algorithm, except for the omission again of not accumulating simulation steps for the minute of a computer swap. However, we focused on the randomized part of the algorithm. The only problem here was that the algorithm was implemented as a method with three parameters, where the third parameter was the number of random trials to compute (non-present in the problem statement). On demand, it was unable to remove such a parameter until the third version, which finally was correct.

C. Use in a Language Processors Course

Language Processors is a compulsory course placed in the second semester of the third year of the Informatics Degree. The course spans 15 weeks, with two sessions per week, each two hours long. Language processors are integral to compilers and interpreters. A compiler analyses and translates source code into executable binary or low-level code. Language processors, a part of a compiler, provide the theoretical foundations for this translation and are heavily reliant on fields like automata and formal languages theory, making them a complex subject in computer science degrees [23].

The course syllabus comprises four main topics: introduction to language processors, lexical analysis, syntax analysis, and syntax-directed translation. The introductory topic briefly describes where language processing concepts are located within scope of computer science and what are its foundations.

The lexicographic analysis topic deals with the first general phase of a language processor. This topic provides an overview with its specific foundations –finite automata and regular expressions–, describes its main responsibilities –mainly producing tokens for the syntax analyzer–, and explains how it can be implemented, including the use of a parser generation tool like ANTLR[1]. Therefore, students have to dedicate some effort to understand how the implementation of a lexical analyzer is actually based on a deterministic finite automaton (DFAs). And, assuming the automatic generation of DFAs from regular expressions, they have to work on representing the typical constructions of a programming language in terms of regular expressions, even deciding which ones can be represented and which ones cannot. The main result of this topic is the construction of a lexical analyzer, also known as the scanner.

The syntax analysis topic deals with the second phase of a language processor. The product of this phase, the syntax analyzer, is also known as the parser. Again its foundations are briefly explained –stack automata and independent context grammars– together with its relation with the scanner. Afterwards, two approaches to parser design are explained: top-down parsing techniques (based on LL grammars) and bottom-up parsing techniques (based on LR grammars). The implementation of these approaches are easily understood, although it is time consuming. Therefore, students are taught how to use automatic parser generation tools in order to be able to build their own parsers; ANTLR is again used for this objective.

The last topic deals with how the parsing process can be used to generate to subsequent actions [24] needed to perform the translations and produce the outcome of a language processor, the third phase. This topic asks to students to use a different approach for its comprehension. Since the two previous topics where mostly based on procedural concepts, this one has a heavy design component. The students have to learn how to manage information –with attributes associated to the symbols of the grammar– and process it –with semantic actions associated to the grammar rules. Again, ANTLR is used to facilitate students a way to implement real language processors.

The course lab project progresses in a similar manner, with students successively developing a scanner, a parser, and a syntax-directed translator. Close to half of the sessions are dedicated to the lab project, including learning how to use ANTLR.

The teaching methodology used in this course is flipped classroom [25]. The first time it was used in the lab part was for the scanner development [26], but it has been extended to the theoretical part of the three last topics, which is more than the 90% of the course. The evaluation of the course is divided in the theoretical part of the course (performed with a test) and the course project lab. Students must pass both parts in order to pass the course.

The contents of this course require form the students the comprehension of the theoretical aspects related to the syllabus. However, the use of the GenAI tool will be focused on the practical exercises, either the simple ones used during class explanations or the more complex ones involved in the course lab project. It must be taken into account that there already exist many tools that support students in some parts of the course, for example, JFlap2[2] can be used to work on the formal languages theory as well as the construction and tracing of parsers with both approaches: LL(1) and SLR(1).

This is the first time that GenAI tools will be used in this course. Given the existing (non GenAI) tools, the students will be most benefited by the use of GenAI with exercises asking the creation or the transformation of elements. Some examples have been tested to evaluate what can the GenAI tool offer to students:

- *Scanner.* GenAI can provide solutions for exercises asking the specification of regular expressions for tokens, as well as their specification using ANTLR. Given the standardization of the language specification for regular expressions and ANTLR, the solutions provided by the GenAI tool are quite correct. But there is an issue with the ANTLR exercises, the solutions provided use either lexical and syntactical grammars, while the exercises must use only lexical grammars. This requires a careful design of the prompt.
- *Parser.* This topic has a heavy load of algorithms, together with the previously mentioned supporting tools. Therefore, the use of GenAI will focus on LL(1) parsers, especially in disambiguation and left recursion removal exercises. Of course, this exercises are solved with algorithms as well, but their application is more difficult than the other algorithms used in this topic. The solutions provided by the GenAI tool are quite correct with useful explanations of how the algorithms are applied.
- *Syntax directed translation.* The use of GenAI in this topic will provide solutions to exercises related to translators written for the ANTLR tool. In fact, solutions provided are correct but, again, the GenAI tool use all the possibilities of ANTLR. This requires from the user rewording the prompt in order to accomplish with requirements, e.g. use only BNF grammars. In addition, many exercises of this topic use a non-formal language for translator specification, similar to pseudocode for programming education. In

[1] https://www.antlr.org/

[2] https://jflap.org/

this case, the GenAI tool could only offer general guidelines for the specification design.

The main role of GenAI in this course will be as a provider of alternative solutions or hints (when the teacher cannot provide them immediately) to exercises. The students will be asked to solve the exercises and then ask the GenAI tool to provide them with alternative solutions. In addition, the effectiveness of the answers provided by GenAI tools highly depends on how they are prompted [27], so GenAI tools often provide wrong or incomplete answers. Therefore, students will be involved in a reflecting exercise comparing their solutions with the ones provided by the GenAI tool (either correct or not). This will support the understanding of the concepts used in the exercise [28].

IV. Discussion

A. Video Games Project

The case study carried out on a video game subject provides relevant information about its use, but even more important is the uncertainty it generates in the students. This highlights the need for further discussion and education about the role and impact of GenAI tools in various fields, including game development.

On the one hand, it is interesting to see how AI is being incorporated into different fields or categories. This can provide students with practical experience and skills that are increasingly relevant in today's digital world. On the other hand, the use of current tools is biased, using only a few tools. The results reveal that the tools most known to the general public are also the most used by video game students. This suggests that familiarity and general awareness play a significant role in tool selection, which may not always lead to the most effective or efficient choice for the task.

Something surprising is the low percentage of use of GenAI coding tools. GitHub Copilot is a widely known tool, but not sufficiently exploited by the engineers or video game developers of tomorrow. This suggests that these tools are not fully utilized in practice, potentially due to a variety of factors such as lack of familiarity, comfort with existing methods, or perceived relevance to the task at hand. It underscores the importance of continued education and exploration of these tools among aspiring developers.

Finally, it is worth noting a feeling of guilt and concern in the use of these types of tools. The students feel as if they are cheating or deceiving themselves, which generates a halo of frustration. According to the more personal responses offered by the respondents, the use of these tools could jeopardize many jobs in the near future. This highlights the ethical and societal implications of AI and automation, which are important considerations in the ongoing development and adoption of these technologies.

B. Advanced Algorithms

The experiment conducted with ChatGPT has revealed strengths and weaknesses. Posteriori, we may guess that the results can be explained from the origin of its "intelligence": pattern recognition but lack of actual understanding. The pros and cons balance each other but, if it was a student of the course, the teacher would probably rate it with a fail grade.

On the one side, ChatGPT generated apparently good algorithms, which corresponded to the design technique demanded and were coded in a good style. It was able to generate alternative algorithms, based on common strategies with a name. ChatGPT also performed consistent transformations of code on demand, usually after several prompts. Finally, it analyzed correctly the time and space complexity of iterative algorithms.

On the other hand, ChatGPT was too verbose, even when succinctness was demanded. It had difficulties understanding an intricate problem statement, as it is the case of the combinatorial optimization problem used for the experiment. This produced inability to fix buggy algorithms, even when the error was identified. ChatGPT also had difficulties dealing with algorithm design decisions. In cases where sophisticated reasoning was required (e.g., bounding functions), wrong strategies were suggested. When it was asked to fix errors, it often persisted in the error. It was sometimes fixed after several prompts, but it was not in other cases. ChatGPT was unable to prove time complexity in mathematical terms (i.e., by using series). Finally, ChatGPT had problems with some cases of tracing, such as presenting a search tree or a recursion tree for given input data. We wonder whether this problem could also be due to the difficulty of presenting non-lineal structures as text.

The difficulties identified are consistent with findings by other authors. Thus, the success of ChatGPT solving problems seems to be inverse to the problem difficulty and the intricacy of the problem statement [11, 12, 13]. It also has difficulties producing an adequate or even a correct answer when the output is not strictly related to the source code but it involves an implicit thinking, e.g., tracing (which involves operational semantics) or proofs (which involve mathematical thinking) [14].

However, the teacher can make use of these virtues and defects in the classroom. ChatGPT can be a good source of inspiration, according to algorithmic strategies often used in certain design techniques or problems. The teacher can use ChatGPT to obtain candidate design decisions or code and discuss them with students, prompting for fixing them and making explicit the errors and inconsistencies presented. The same didactic use can be given to summaries demanded of algorithmic behavior. Finally, ChatGPT can be used to illustrate good programming style.

C. Language Processors

The inclusion of GenAI tools in education is at an early stage and more experience is needed in order to find effective ways of use, design proper educational contexts and check the impact on the learning experience. Three main aspects have emerged from this proposal. Firstly, the usefulness of the solutions provided by the GenAI tool depends on the type of exercises. The solutions are useful and correct when the exercises are simple and must be specified using a standard and formal language, e.g. regular expressions, context free grammars or ANTLR specifications. This could support students for concrete doubts and errors [29]. However, when some restrictions must be applied to solutions or the complexity of exercises is increased, the user has to dedicate a significant effort to design carefully the prompt in order to receive a solution.

Secondly, following the previous idea, it must be studied to what extent the effort dedicated to prompt design supports student's learning. In addition, students must be able to evaluate the correctness of the solutions provided by the GenAI tool. Although it has been showed that working with

wrong examples is positive [28], there is a risk of incorporating erroneous knowledge [30] if students do not recognise wrong or incomplete solutions. The approach that students should follow could be guided by the following questions: *Do I completely understand the solution? Am I sure that this solution is sound and correct?*

Finally, the following question raises, Should the evaluation system be adapted to the use of GenAI tools? Given the current concern about cheating and plagiarism with GenAI tools [30], the answer is yes, it must be adapted. The evaluation of the theoretical part of this course has no problem with GenAI tools because it is performed by invigilated tests. On the contrary, the evaluation of the course lab project must be adapted taking into account that students will be able to use GenAI tools. This could be done in some different ways. On the one hand, a compulsory oral presentation of the lab project could be required. This means that more time will be needed in order to schedule the presentations of all the students. On the other hand, the theoretical tests could include essay questions directly related with the students' solutions to lab project.

V. Conclusions

We have presented three different experiences of use of GenAI in informatics courses. One experience has presented the results of a survey on the use of GenAI by students of a video games degree. Another experience has presented the results of experimenting an instructor with different techniques for an optimization problem in the context of an advanced algorithm course. Finally, a proposal of use of GenAI in a language processors course has been presented.

The main results of the experiences follow. Firstly, GenAI is a technology that must not be ignored by instructors, because students are actually using them. Furthermore, instructors must think carefully about how to integrate the different types of GenAI (text, images, code, etc.) in their courses. Secondly, students could benefit from being trained in these tools, especially when they are demanded a large number of artifacts and different media. In particular, GenAI tools often generate inaccurate code or explanations for non-trivial tasks. Consequently, critical assessment of GenAI deliveries is a necessary skill that students and instructors must develop for their productive use. Finally, the courses must be adapted in their different parts to successfully integrate GenAI. The reflective of GenAI tools can be integrated into the classroom instruction, as a third part who may be asked to participate by contributing in the class dynamics. Assignments must be reconsidered to avoid cheating. Good potential measures are demanding students to generate different but related artifacts (e.g., visualizations) or to make an oral presentation of their work.

References

[1] Joint Council for Qualifications, "AI use in assessments: Protecting the integrity of qualifications," https://www.jcq.org.uk/wp-content/uploads/ 2023/04/JCQ-AI-Use-in-Assessments-Protecting-the-Integrity-of-Qualifications.pdf, 2023.

[2] M. Welsh, "The end of programming," *Communications of the ACM*, vol. 66, no. 1, pp. 34-35, January 2023.

[3] D. M. Yellin, "The premature obituary of programming," *Communications of the ACM*, vol. 66, no. 2, pp. 41-44, February 2023.

[4] M. Hirzel, "Low-code programming models," *Communications of the ACM*, vol. 66, no. 10, pp. 76-85, October 2023.

[5] D. Weintrop, "Block-based programming in computer science education," *Communications of the ACM*, vol. 62, no. 8, pp. 22-25, August 2019.

[6] A. Shoufan, "Can students without prior knowledge use ChatGPT to answer test questions? An empirical study," *ACM Transactions on Computing Education*, vol. 23, no. 4, article 45, Dec. 2023.

[7] I. J. Pérez-Colado, V. M. Pérez-Colado, A. Calvo Morata, R. Santa Cruz Píriz and B. Fernández-Manjón, "Using new AI-driven techniques to ease serious games authoring," 2023 IEEE Frontiers in Education Conference (FIE), College Station, TX, USA, 2023, pp. 1-9, doi: 10.1109/FIE58773.2023.10343021.

[8] N. Ahmad, S. Murugesan, and N. Kshetri, "Generative artificial intelligence and the education sector," *Computer*, vol. 56, no. 6, pp. 72-76, June 2023.

[9] British Government, "Generative artificial intelligence (AI) in education", policy paper. Retrieved from https://www.gov.uk/government/publications/generative-artificial-intelligence-in-education/generative-artificial-intelligence-ai-in-education, October 2023.

[10] Code.org, CoSN, Digital Promise, European EdTech Alliance, Larimore, J., and PACE, "AI guidance for schools toolkit". Retrieved from https://www.teachai.org/toolkit, December 2023.

[11] N. D. Tran, J. J. May, N. Ho, and L. B. Ngo, "Exploring ChatGPT's ability to solve programming problems with complex context," *Journal of Computing Sciences in Colleges*, vol. 39, no. 3, pp. 195-209, Oct. 2023.

[12] B. Puryear, and G. Sprint, "Github Copilot in the classroom: Learning to code with AI assistance," *Journal of Computing Sciences in Colleges*, vol. 38, no. 1, pp. 37-47, Nov. 2022.

[13] E. L. Ouh, B. K. S. Gan, K. J. Shim, and S. Wlodkowski, "ChatGPT, can you generate solutions for my coding exercises? An evaluation on its effectiveness in an undergraduate Java programming course," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023)*, pp. 55-60.

[14] J. Savelka, A. Agarwal, C. Bogart, Y. Song, and M. Sakr, "Can Generative Pre-trained Transformers (GPT) pass assessments in higher education programming courses?," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023)*, pp. 117-123.

[15] R. Balse, B. Valaboju, S. Singhal, J. M. Warriem, and P. Prasad, "Investigating the potential of GPT-3 in providing feedback for programming assessments," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023)*, pp. 292-298.

[16] J. Prather, B. N. Reeves, P. Denny, B. A. Becker, J. Leinonen, A. Luxton-Reilly, G. Powell, J. Finnie-Ansley, and E. A. Santos, "«It's weird that it knows what I want»: Usability and interactions with Copilot for novice programmers," *ACM Transactions on Human-Computer Interaction*, vol. 31, no. 1, article 4, Nov. 2023.

[17] J. Leinonen, P. Denny, S. MacNeil, S. Sarsa, S. Bernstein, J. Kim, A. Tran, and A. Hellas, "Comparing code explanations created by students and large language models," in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023)*, pp. 124-130.

[18] T. Lehtinen, "Questions about learners' code: Extending automated assessment towards program comprehension," PhD thesis, Aalto University, 2024.

[19] J. Á. Velázquez-Iturbide, and A. Pérez-Carrasco, "How to use the SRec visualization system in programming and algorithm courses,", *ACM Inroads*, vol. 7, no. 3, pp. 42-49, Sept. 2016.

[20] J. Á. Velázquez-Iturbide, "A unified framework to experiment with algorithm optimality and efficiency," *Computer Applications in Engineering Education*, vol. 29, no. 6, pp. 1793-1810, 2021.

[21] J. Kleinberg, and É. Tardos, Algorithm Design, Pearson Education, 2006.

[22] E. Horowitz, S. Sahni, and S. Rajasekaran, Computer Algorithms, Computer Science Press, 1998.

[23] M. Hewner, "Undergraduate conceptions of the field of computer science," in *Proc. of the Ninth Annual Intl. ACM Conf. on International Computing Education Research, ser. ICER '13*. New York, NY, USA: ACM, 2013, pp. 107–114. [Online]. Available: http://doi.acm.org/10.1145/2493394.2493414

[24] D. Knuth, "Semantics of context-free languages," 1968, pp. 127-145. [Online]. Available: https://doi.org/10.1007/BF01692511

[25] M. Lage, G. Platt, and M. Treglia, “Inverting the classroom: A gateway to creating an inclusive learning environment,” *Journal of Economic Education*, vol. 31, no. 1, pp. 30–43, 2000. [Online]. Available: https://doi.org/10.2307/1183338

[26] J. Urquiza-Fuentes, “Increasing students’ responsibility and learning outcomes using partial flipped classroom in a language processors course,” *IEEE Access*, vol. 8, pp. 211211–211223, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3039628.

[27] L. S. Lo, “The art and science of prompt engineering: A new literacy in the information age,” *Internet Reference Services Quarterly*, vol. 27, no. 4, pp. 203–210, 2023.

[28] T. Heemsoth and A. Heinze, “The impact of incorrect examples on learning fractions: A field experiment with 6th grade students,” *Instructional Science*, vol. 42, pp. 639–657, 2014.

[29] K. Kuramitsu, Y. Obara, M. Sato, and M. Obara, “Kogi: A seamless integration of ChatGPT into Jupyter environments for programming education,” in *Proceedings of the 2023 ACM SIGPLAN International Symposium on SPLASH-E*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 50–59.

[30] J. Prather, P. Denny, J. Leinonen, B. A. Becker, I. Albluwi, M. Craig, H. Keuning, N. Kiesler, T. Kohn, A. Luxton-Reilly, S. MacNeil, A. Petersen, R. Pettit, B. N. Reeves, and J. Savelka, “The robots are here: Navigating the generative AI revolution in computing education,” in *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 108–159.