Prompt Engineering for LLMs Educational Alignment

Giulio Barbero[1], Mike Preuss[1]

[1]Leiden University (LIACS), Einsteinweg 55, 2333CC, Leiden, Netherlands

Abstract

Since the release of GPT in 2022, generative artificial intelligence (AI) has quickly become ubiquitous. In educational institutions, the impact of generative AI has almost monopolised internal debate. Most of us have probably been involved in discussions about how to tackle the growing reliance of students on large language models (LLM). Research on the impact of this technology in education is catching up. In the case of computer science education, experimental projects in the field report contradictory results. This paper is composed of two parts: it first explores existing literature and compares studies to shed light on the impact of generative AI specifically for programming education. In doing so, we highlight the risks arising from uncontrolled generative AI. However, we also argue in favour of controlled generative AI as a tutoring tool. This leads us to the second part; we involve experts in a preliminary experiment to investigate how prompting can be used to align AI models with specific educational styles. Initial results suggest that detailed role-playing instructions, built upon existing pedagogical research, lead to more effective feedback.

Keywords

tutoring, generative AI, large language models, programming education

1. Introduction

Generative AI has rapidly become widespread across various activities. As is often the case, the advancement of disruptive technologies fosters new discussions within old contexts; think about the legal considerations around AI-generated content [1] or the studies about potential applications of generative AI in education [2]. Concerning the latter, many studies highlight the risks of using generative AI in computer science education. For example, with the assistance of GPT models, students are able to complete assignments more quickly, but they also retain less information compared to their peers who worked without AI help [3, 4]. On the other hand, other studies in similar settings revealed that students' computational thinking skills improved using generative AI [5].
Therefore, to have a full picture of the impact of generative AI on programming education, the discrepancies among studies should be analysed. In the first part of the present paper, we perform a literature review to investigate current experimental studies in the field. From this, we isolate how the characteristics and goals of the experimental context influence its effectiveness. As a result of this analysis, we argue that:

1. Unrestricted use of LLMs in class positively affects students' confidence but negatively impacts knowledge retention.

2. LLMs have potential as tutoring tools as long as prompting is somehow restricted.

In the second part, we explore the possible applications of generative AI as a tutoring tool by developing and testing prompts based on the quality of the feedback provided. We build upon existing research, which showed how careful prompt engineering can be used to misalign LLMs with so-called "persona attack" [6]. However, we aim to use prompt engineering to align an LLM's responses with education contexts. We specifically want to test the impact of role-playing information, a common technique used in persona attacks. Ultimately, the goal of the paper is to discuss the following research questions:

1) What is the effect of generative AI on programming education?

g.barbero@liacs.leidenuniv.nl (G. Barbero); m.preuss@liacs.leidenuniv.nl (M. Preuss)
0000-0003-1258-4915 (G. Barbero); 0000-0003-4681-1346 (M. Preuss)

- How is the impact of current experimentations measured?
- What are the results of these interventions?
- How can we reconcile apparently opposite results in the field?

2) How does the prompt influence the quality of the feedback provided?

2. Literature Review

Our understanding of generative AI's impact on programming education remains limited. A recent literature review about empirical research in the field includes only thirty-seven studies [7]. Of these, only two evaluate students' computational thinking and programming skills. In fact, the majority of the studies are limited to how LLMs perform in terms of programming. These include skills such as debugging [8], pair programming [9] or program generation [10]. However, these skills do not necessarily reflect the impact of generative AI on learning; they depict it as a useful tool for programmers. This seems to be a common misconception in current research in the field, where generative AI performance in teaching environments is evaluated based on its programming performance [7]. Going back to the aforementioned literature review [7], the two studies focusing on students' computational thinking and programming skills development present very different methodologies. The first one highlights the positive effects on both metrics for students using generative AI. However, the measurement of computational thinking is quite complex and heavily dependent on the model used. In the case of this experiment, computational thinking skills are measured using a scale based on a more abstract model [11]. This includes skills that are not necessarily exclusive or focused on programming: creativity, algorithmic thinking, problem solving, critical thinking, cooperativity and communication skills. However, the most critical flaw of the study is the testing methodology for programming skills development, measured using a self-efficacy scale focused on students' confidence in tackling abstract programming problems [12] (see figure 1). We argue that, with these tools, conclusively evaluating the actual impact of generative AI on students' programming learning is impossible.

The second study has a completely different approach. In this case, generative AI has been implemented in a gamified interface [13]. The article presents the experiment as ongoing, and therefore, actual data about students' performance and improvement is not available and necessitates further studies. The author only reports positive effects on motivation and acceptance that are typical of a gamified environment [14]. However, the interesting element in the study is the innate limitations of generative AI that a gamified environment can present. In general, both studies are reported as dealing with computational thinking and programming skills development, but we argue that their final measurements are not necessarily focused on these topics. Studies measuring actual student performance have only recently emerged. For example, a recent experiment in a programming class of Fortran indicates that retention is superior for students who do not use generative AI [4]. In the experimental setup, the experimental groups are allowed a quite free use of various modern generative models. The control group is allowed to use Google exclusively.

However, other studies take a different perspective, evaluating generative AI in its ability to tutor students. Current research highlights the potential for LLMs to perform almost as well as human tutors [15]. Obviously, evaluation of tutoring is quite complex, and it often relies on experts. Moreover, in order to make results comparable, many studies limit what students can ask, for example, using preselected prompts [15]. In general, most studies report positive results on students' acceptance and motivation using generative AI [16]. However, others highlight a negative correlation between perceived ease of use and perceived usefulness [17].

2.1. Discussion

In this chapter, we take a moment to answer the sub-questions of the first research question. We deem this step important in order to illustrate the relevance of the subsequent experiment as an exploration of the second research question.

Item No.	Item Description (alpha reliability estimates for factors in parentheses)
	Factor 1: Independence and persistence (alpha = .94)
23	Complete a programming project if I had a lot of time to complete the program.
22	Complete a programming project once someone else helped me get started.
21	Complete a programming project if I could call someone for help if I got stuck.
24	Complete a programming project if I had just the built-in help facility for assistance.
20	Complete a programming project if I had only the language reference manual for help.
19	Complete a programming project if someone showed me how to solve the problem first.
25	Find ways of overcoming the problem if I got stuck at a point while working on a programming project.
17	Debug (correct all the errors) a long and complex program that I had written, and make it work.
	Factor 2: Complex programming tasks (alpha = .94)
13	Understand the object-oriented paradigm.
16	Make use of a class that is already defined, given a clearly labeled declaration of the class.
14	Identify the objects in the problem domain and declare, define, and use them.
8	Build my own C++ libraries.
18	Comprehend a long, complex multi-file program.
12	Organize and design my program in a modular manner.
32	Write a program that someone else could comprehend and add features to at a later date.
15	Make use of a pre-written function, given a clearly labeled declaration of the function.
11	Write a long and complex C++ program to solve any given problem as long as the specifications are clearly defined.
28	Mentally trace through the execution of a long, complex, multi-file program given to me.
29	Rewrite lengthy confusing portions of code to be more readable and clear.

Figure 1: Example of statements used for students' self-efficacy evaluation in [12].

2.1.1. How is the impact of current experimentations measured?

Current research primarily measures the acceptance of generative AI among students and teachers. This is often performed through the typical technology acceptance model, a well-studied and validated tool for measurement. On the other hand, other aspects of the impact of generative AI on programming education are more difficult to measure. We argue that metrics and research goals are not always well aligned. In the case of computational thinking skills development, different models can be used as references. However, each model has a different perspective on these skills and selecting the best-suited one is a necessary evaluation. As for programming skills, teachers already have many tools to test students' learning. Self-assessment tools have their own reason and space, however, they tend to be more strongly related to the respondent's confidence than their actual development. Confidence is particularly reinforced with the ability to perform a certain job, something that, especially in introductory programming curricula, generative AI can certainly provide. However, we argue that this is not necessarily related to students' proficiency in programming but with their perceived capacity to pass the assignments provided. Other studies strongly focus on human comparison. In these cases, measurements are usually performed by humans who evaluate generative AI's performance compared

to other human experts. It is the case of experiments centred around the effect of AI-powered tutoring. Another specific characteristic of this format is that it often relies, by necessity, on predetermined prompts or other forms of limitations that make human and AI tutoring comparable.

2.1.2. What are the results of these interventions?

Results of empirical research in the field show a definite improvement in students' motivation. This is definitely a relevant effect, probably related to the ease of use of generative AI and the enthusiasm emerging from a new, and quite frankly impressive, technology. Results emerging from technology acceptance studies (using variations of the technology acceptance model [18]) are also extremely encouraging, especially for younger participants. As mentioned above, many studies also highlight a positive effect on students' confidence and self-assessment. However, these effects do not automatically translate into students' final retention and learning. In this regard, the ability to have at one's disposal immediate solutions may hinder deep learning, as students bypass the work required to internalise concepts. In fact, cognitive load theory suggests that excessive assistance reduces mental effort, preventing students from actively engaging in knowledge construction, no matter if the assistance is by humans or AI. In studies about AI tutoring, human tutors evaluate generated answers positively, almost at the level of human tutoring. However, the performance of AI tutoring without constraints compared to its human counterpart is still unknown.

2.1.3. How can we reconcile apparently opposite results in the field?

The diversity of results in the field can be justified by two main elements:

- The field is still in its infancy; as shown in [7], a related literature review on empirical studies only reports thirty-seven studies. It is natural that this leads to a great variation of results as the novelty gives great space for exploration.
- Mainly, the results are not necessarily conflicting; effects on motivation and self-efficacy reports do not necessarily translate to performance or retention. Students can feel more motivated and more confident in engaging with their tasks, but at the same time, not fully absorb the necessary information.

3. Experiment: Prompt Engineering for Tutoring

In this section, we investigate whether prompt engineering can be used to control generative AI and align it with a tutoring role. While we can look at this specific problem from very different perspectives, we begin by taking a naive approach to it, arguing that giving the model research-grounded directions will lead to better feedback. In this regard, we make use of two main tutoring directions. First, we want the model to follow a bottom-up approach, providing problem-specific feedback first and, if appropriate, only subsequently introducing more general concepts [19]. By doing so, we truly focus on the role of generative AI as a tutor in the context of assignment completion. Secondly, we want our model to facilitate students' problem-solving instead of replacing it completely. Therefore, our second tutoring direction is to use a "moderate" approach, preserving students' autonomy and agency [20]. Finally, in order to make the experiment as realistic as possible, we use two actual programming assignments used in introductory courses of our faculty. Therefore, we design three separate prompt conditions:

- No instructions prompt ("Can you help with this assignment?" + assignment)
- Simple instructions prompt ("You are a programming tutor. Can you help with this assignment?" + assignment)
- Detailed instructions prompt ("You are a programming tutor. You give students moderate directions but let them retain autonomy in completing their homework. You focus on the specific

problem at hand before providing insight into general concepts related to it. Can you help with this assignment?" + assignment)

3.1. Methodology

First, we need to recreate a reasonable student query. We take into consideration two assignments:

- merging two distinct files containing lists of email addresses into one final list. Some email addresses are present in both lists, the final result should avoid repeating the same address twice. Moreover, we give specific instructions about the order in which these two lists should be processed.
- checking a string containing a DNA sequence. The students need to check that the string contains only valid characters ("A", "C", "T" or "G"). If these are the only letters, the code should print "Valid". On the other hand, if an invalid character is found, the code should stop and print "Invalid" followed by the valid letters so far.

In order to simulate a typical interaction, we also provide the LLM with code that includes common student mistakes for each assignment. Therefore, the text for the email lists merging assignment is the following:

```
#Assignment 4 - Mail merge, Unique emails, DNA valid part

#In the assignment 4 link on Brightspace you are provided with two
    txt files, each containing a list of emails. Some emails are
    recurring in both documents.

#A. Your first goal in this assignment is to merge the two lists of
    emails into one final list containing all email addresses only
    once (no doubles!). Print first the emails appearing in emails_01
    .txt then the others.
#Print the final list of emails, with no duplicates and no space
    between the lines (simply each line one email). Compare with the
    expected output in the assignment documentation.


file_1 = open("emails_01.txt", "r")
text_1 = file_1.read()
file_2 = open("emails_02.txt", "r")
text_2 = file_2.read()

x = text_1 + "\n" + text_2
print(x)
```

The text for the DNA checking assignment is the following:

```
dna_sequence = input("Enter␣DNA␣sequence:␣").upper()

#DNA sequences can only be formed of the letters "ACTG", any other
    letter is wrong.
#Your code should go through the sequence and print "Valid" if the
    sequence is fully correct and "Invalid PART_OF_SEQUECE_VALID" if
    the sequence is only partially correct. Notice that you do NOT
    print all the valid letters but only the ones valid until the
    wrong one (see below)
```

```
#Example 1: "ACTG" -> "Valid"; "UTCG" -> "Invalid "; "ACUTG" -> "
    Invalid AC"
#Print "valid" or "invalid" only once!

for letter in dna_sequence:
    if letter is not "A" or "C" or "T" or "G":
        print("invalid")
    else:
        print("valid")
```

As mentioned before, the final input takes the form of *(prompting condition + assignment text)*. We use the generated output as is in order to avoid biases. For this experiment, we use Gemini2.0 as one of the most widely available state-of-the-art models. We then ask expert programming teachers to compare and rate the results. We also ask them two questions after each assignment for qualitative analysis:

- What elements did you take into consideration rating the responses?
- What would you have done differently if the student had asked for your help?

3.2. Results

We have surveyed 3 experts with extensive experience in tutoring students in programming courses. The highest-scoring prompt for both assignments is the one including detailed tutoring instructions. On the other hand, the prompts including no or simple instructions are scored low (see fig.2).

As for the question *"What elements did you take into consideration rating the responses?"*, all experts mentioned for both assignments that they considered whether the produced answer is accurate. Furthermore, they took into consideration how much the answer can be simply acritically interpreted as the solution to the assignment. Specifically, two experts mention the risk of the student copy-pasting solutions, leading to reduced learning. Finally, in the question *"What would you have done differently if the student had asked for your help?"*, two experts mentioned they would take a step-by-step approach, leading the student towards the solution in a constructivist fashion.

3.3. Discussion

In this section, we use our preliminary data to answer the second research question. In general terms, we can say that prompting can improve the tutoring skills of LLMs. Specifically, our results so far indicate that role-playing can be used to better align AI models with educational contexts. However, the way role-playing information is built also influences the final results; specifically, providing clear indications about the desired tutoring style based on existing research yields better results than more generic role-playing indications (i.e., "You are a tutor"). The experts involved indicated two main pitfalls arising when it comes to LLM tutors:

- the possibility that the solution provided is incorrect
- the tendency to provide the complete solution to the assignment at hand

The first point is quite straightforward; due to the stochastic nature of AI systems and the tendency of LLMs to hallucinate, there is the possibility for the output to be incorrect. This becomes particularly central when it comes to education, and it is natural that good tutoring is associated with giving correct information. The second point derives from a constructivist view on education, which is supported by the above literature review. Expanding this reasoning, students who use generative AI freely (as reported in [4]) retain less knowledge because the learning process is disrupted by the generated answers containing the full solution.

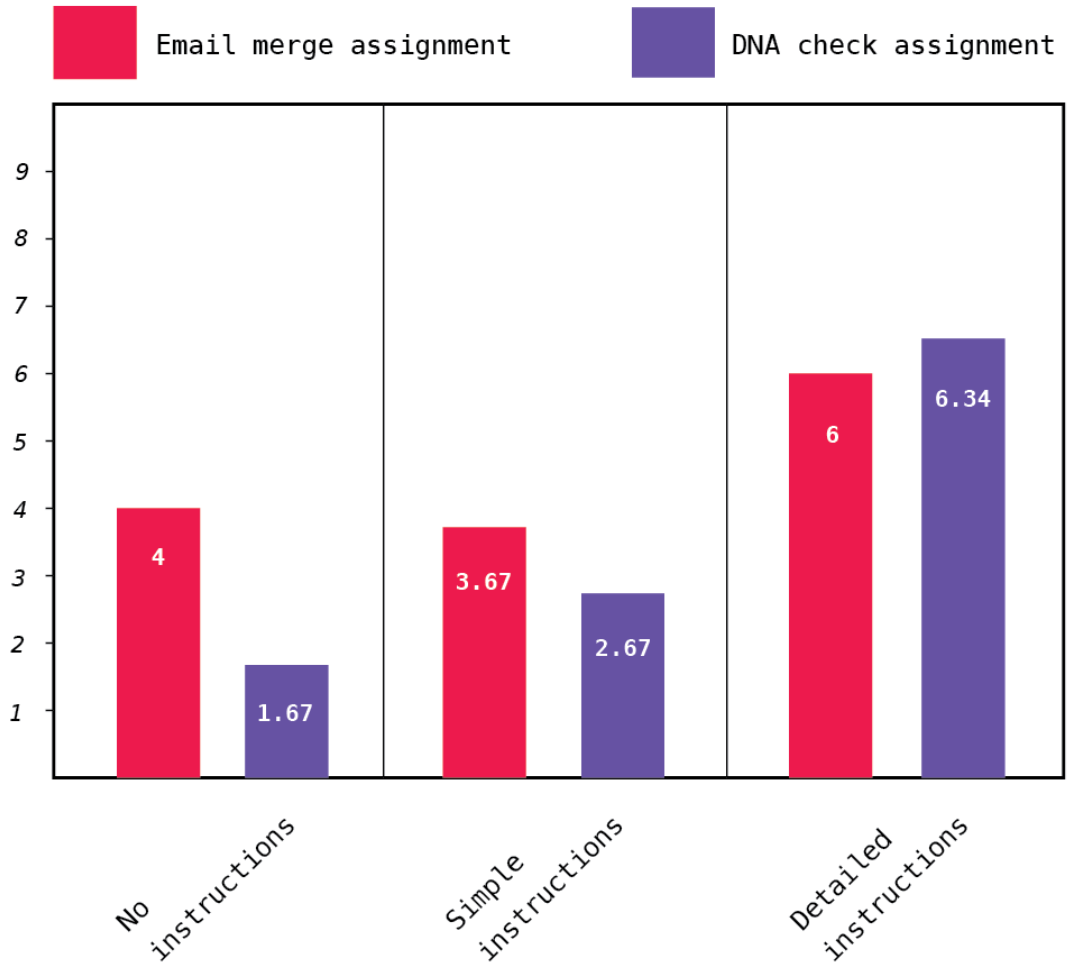

Figure 2: Average score given by the three experts to each generated message for each assignment

4. Conclusions

Literature research in the field indicates that, although unrestricted generative AI has a negative impact on programming education, its controlled applications can be very useful to provide personalised and accurate feedback to students. Control over these applications can be enforced through prompt engineering, as long as the indications are detailed and based on existing pedagogical research. In this regard, LLMs seem to be able to align themselves with the desired tutoring style and provide qualitatively better output. More generic limitations, however, yield worse results with the risk of disrupting students' learning process.

4.1. Limitations

This paper presents a preliminary study, and limitations are numerous. At the moment, we are working towards tackling three in particular:

- low number of responders: obviously, the number of responders for this study is far lower than the statistical relevance. We aim to expand the study involving more experts. In particular, we want to include teachers with an instructivist approach, although these might be a rarity in programming education.

- one model: there is great variance between the performance of different LLMs. Exploring different ones can provide insight into how the model influences the results. Moreover, testing the capabilities of smaller models can also be valuable, specifically for applications in low-resource contexts.
- interaction realism: our methodology revolves around evaluating a single generated response to faulty codes. However, real-life use of these technologies would greatly differ, often taking the form of a conversation between the user and the LLM. We argue that extending the interaction will probably highlight the differences among prompts even further.

4.2. Future Work

When it comes to future work in the field, it is important to continue to develop empirical literature to paint a clearer picture of the impact of generative AI on programming education. In particular, we need more research focusing on the performance and retention of programming knowledge and skills. While performing evaluations in an actual teaching context is extremely valuable, we have to consider that the ubiquity of generative AI could influence the results in unpredictable ways. Therefore, it could be valuable to start in smaller and more controlled settings. Moreover, AI tutors are flexible technologies, and their impact will likely vary depending on the medium used. As mentioned in the literature review, there are promising applications involving games and gamification [13]. The main advantage of these media is that they can inherently and subtly implement prompt engineering and other forms of control over the generated content while presenting even stronger effects on students' motivation [14, 21]. This is another promising direction for future empirical research. Finally, the effect of LLM tutors is also impacted by the way they are incorporated into the learning environment, whether this is a university course or an educational video game. Empirical research performed in the field should also take into account the way AI technologies are presented and implemented from a holistic perspective.

5. Declaration on Generative AI

During the preparation of this work, the author(s) used Grammarly in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] R. Abbott, E. Rothman, Disrupting creativity: Copyright law in the age of generative artificial intelligence, Fla. L. Rev. 75 (2023) 1141.

[2] D. Baidoo-Anu, L. O. Ansah, Education in the era of generative artificial intelligence (ai): Understanding the potential benefits of chatgpt in promoting teaching and learning, Journal of AI 7 (2023) 52–62.

[3] E. Klopfer, J. Reich, H. Abelson, C. Breazeal, Generative AI and K-12 Education: An MIT Perspective, An MIT Exploration of Generative AI (2024). Https://mit-genai.pubpub.org/pub/4k9msp17.

[4] E. Shein, The impact of ai on computer science education, 2024.

[5] R. Yilmaz, F. G. K. Yilmaz, The effect of generative artificial intelligence (ai)-based tool use on students' computational thinking skills, programming self-efficacy and motivation, Computers and Education: Artificial Intelligence 4 (2023). doi:10.1016/j.caeai.2023.100147.

[6] L. Schwinn, D. Dobre, S. Xhonneux, G. Gidel, S. Günnemann, Soft prompt threats: Attacking safety alignment and unlearning in open-source llms through the embedding space, Advances in Neural Information Processing Systems 37 (2024) 9086–9116.

[7] F. Deriba, I. T. Sanusi, O. O Campbell, S. S. Oyelere, Computer programming education in the age of generative ai: Insights from empirical research, SSRN (2024).

[8] F. A. Sakib, S. H. Khan, A. R. Karim, Extending the frontier of chatgpt: Code generation and debugging, in: 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET, IEEE, 2024, pp. 1–6.

[9] D. Spinellis, Pair programming with generative ai, IEEE Software 41 (2024) 16–18. doi:10.1109/MS.2024.3363848.

[10] B. Idrisov, T. Schlippe, Program code generation with generative ais, Algorithms 17 (2024). doi:10.3390/a17020062.

[11] Özgen Korkmaz, R. Çakir, M. Y. Özden, A validity and reliability study of the computational thinking scales (cts), Computers in Human Behavior 72 (2017) 558–569. doi:10.1016/j.chb.2017.01.005.

[12] V. Ramalingam, S. Wiedenbeck, Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy, Journal of Educational Computing Research 19 (1998) 367–381. doi:10.2190/C670-Y3C8-LTJ1-CT3P.

[13] C. Cao, Scaffolding cs1 courses with a large language model-powered intelligent tutoring system, in: International Conference on Intelligent User Interfaces, Proceedings IUI, Association for Computing Machinery, 2023, pp. 229–232. doi:10.1145/3581754.3584111.

[14] S. Deterding, D. Dixon, R. Khaled, L. Nacke, From game design elements to gamefulness: Defining "Gamification", in: Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments, MindTrek '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 9–15. URL: https://doi.org/10.1145/2181037.2181040. doi:10.1145/2181037.2181040.

[15] T. Phung, V.-A. Pădurean, J. Cambronero, S. Gulwani, T. Kohn, R. Majumdar, A. Singla, G. Soares, Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors, in: Proceedings of the 2023 ACM Conference on International Computing Education Research-Volume 2, 2023, pp. 41–42.

[16] A. Baytak, The acceptance and diffusion of generative artificial intelligence in education: A literature review, Current Perspectives in Educational Research 6 (2023) 7–18.

[17] K. Kanont, P. Pingmuang, T. Simasathien, S. Wisnuwong, B. Wiwatsiripong, K. Poonpirome, N. Songkram, J. Khlaisang, Generative-ai, a learning assistant? factors influencing higher-ed students' technology acceptance, Electronic Journal of e-Learning 22 (2024) 18–33.

[18] F. D. Davis, Perceived usefulness, perceived ease of use, and user acceptance of information technology, MIS Quarterly: Management Information Systems 13 (1989) 319–339. doi:10.2307/249008.

[19] G. Sandstrak, B. Klefstad, A. Styve, K. Raja, Analyzing pedagogic practice and assessments in a cross-campus programming course, IEEE Transactions on Education (2024).

[20] K. E. Boyer, R. Phillips, M. D. Wallis, M. A. Vouk, J. C. Lester, The impact of instructor initiative on student learning: A tutoring study, ACM SIGCSE Bulletin 41 (2009) 14–18.

[21] M. Hassenzahl, M. Burmester, F. Koller, Attrakdiff: Ein fragebogen zur messung wahrgenommener hedonischer und pragmatischer qualität, Mensch & Computer 2003: Interaktion in Bewegung (2003) 187–196.