Lessons Learned 6, 1 (2026)
Submitted: November 10, 2025
Accepted: February 11, 2026

DOI: https://doi.org/10.25369/ll.v6i1.109
ISSN: 2749-1307.

The benefits and side effects of using ChatGPT in programming – a pilot study in the teaching and learning lab technology

I. Gideon, N. Link

Chair of Technology and Technical Education, Institute of Physics and Technical Education, Faculty of Natural and Social Sciences, Karlsruhe University of Education

Abstract

Generative AI chatbots such as ChatGPT offer new possibilities for supporting learning processes in programming education, but their effectiveness depends on their didactic integration. In a pilot study, students developed an Arduino project with the help of ChatGPT. Both the programs created by the students and their logbooks on prompt usage were analyzed. The logbooks were coded using a literature-based category system of effective prompts. The results show that the prompts were mostly imprecise, hardly contextualized, and rarely developed iteratively. Students often adopted the generated program code without understanding its functional logic, which was reflected in difficulties in recognizing and correcting code errors. The findings suggest that programming based solely on ChatGPT does little to promote knowledge acquisition among novices and may even hinder it.

Generative KI-Chatbots wie ChatGPT zeigen neue Möglichkeiten zur Unterstützung von Lernprozessen in der Programmierausbildung, deren Wirksamkeit jedoch von der didaktischen Einbettung abhängt. In einer Pilotstudie entwickelten Studierende ein Arduino-Projekt mithilfe von ChatGPT. Analysiert wurden sowohl die erstellten Programme der Studierenden als auch ihre Logbücher zur Promptnutzung. Die Logbücher wurden anhand eines literaturgestützten Kategoriensystems effektiver Prompts kodiert. Die Ergebnisse zeigen, dass die Prompts meist unpräzise, kaum kontextualisiert und selten iterativ weiterentwickelt wurden. Studierende übernahmen den generierten Programmcode häufig, ohne dessen Funktionslogik zu verstehen, was sich in Schwierigkeiten beim Erkennen und Beheben von Codefehlern zeigte. Die Befunde deuten darauf hin, dass eine rein ChatGPT-basierte Programmierung den Wissenserwerb von Novizinnen und Novizen kaum fördert oder ihn erschweren kann.

*Corresponding author: igor.gideon@ph-karlsruhe.de

This article was originally submitted in German.

1. Background

Understanding how software works and writing programs are common requirements in technical professions; the complexity of learning to program often leads to educational failure and frustration among learners [1]. Generative AI chatbots such as ChatGPT are increasingly being used in programming education [2, 3] because they open up novel and promising possibilities for supporting such learning processes [4, 5]. In addition to pure code generation, they can provide explanations, identify syntax errors, or offer alternative solutions [6]. Their use is now also increasingly being discussed in subject-specific didactic research [7, 8]. Overall, the integration of OpenAI into programming education is becoming a general trend, with chatbots, intelligent tutoring systems, and automated assessment systems enabling personalized support and feedback [1, 9].

Co-programming tools are also gaining in importance, integrating generative AI directly into development environments (e.g., Cursor/X, Visual Studio Code with GitHub Copilot, or development environments with agents such as Antigravity or Gemini), thus enabling a high degree of task and process integration in the programming workflow [23]. In scientific studies, this closely interlinked form of human-AI collaboration is described as a "cyborg" approach, in which the workflow is continuously intertwined with AI interactions (in contrast to more division-of-labor-based "centaurs") [23].

Typical strengths of generative AI chatbots lie in natural language processing and the possibility of personalized feedback [10, 11, 12]. In addition, models from alternative providers (e.g., Anthropic Claude Opus medium) are often shown to be very powerful in public performance comparisons (e.g., SWE-bench).

Nevertheless, their use also has limitations, for example in the form of faulty code, superficial logic, or ethical issues [10, 12, 13]. High expectations of the benefits of AI, coupled with perceived risks, confirm this ambivalence [14].

A similar tension exists in the educational context: while some students report individually perceived increases in efficiency and learning progress, teachers observe losses in quality and competence among learners resulting from the use of OpenAI [2]. It is often found that the use of OpenAI encourages a more receptive style of use, with students tending to copy and paste the generated code without understanding how it works [3, 8]. Particularly critical is the finding that the use of ChatGPT leads to a significant decline in collaborative learning (interaction and discussion) [2].

Although established guidelines for learning-promoting prompting exist in the literature, these have rarely been consistently implemented in practice to date. These include simple chain-of-thought prompts (e.g., from the general to the specific), which increase the accuracy and quality of AI responses [6], as well as the frequently cited *Five S framework* (contextualization, precision, structuring, meta-organization, iteration) as a practical heuristic for improving output quality [8]. Studies report that learners without a didactic framework overlook the fundamental aspects of successful prompting, so that the actual learning progression tends toward passivity [2]. This logically leads to deficits in the ability to understand, critically analyze, or modify program code [2]. In this way, the potential of generative AI chatbots can be used less as a learning aid and more as a shortcut to solutions [2]. In this context, researchers warn against "blind trust," as the easy option of having ChatGPT provide solutions can impair the development of one's own critical thinking and independent problem-solving skills [12]. Instead, knowledge should be acquired through dialogue, in this context through feedback interactions with ChatGPT [15]. Overall, the research results to date suggest that ChatGPT is only of limited benefit to learning in programming without educational measures [16, 17].

One possible reason for this could be an insufficient understanding of how AI systems work—in particular, what inputs are required for learning processes, not just for generating outputs. A lack of knowledge about the specific wording of individual prompts and the metacognitive ability to use them strategically therefore makes it difficult to use AI effectively for learning. It is therefore recommended that the use of prompts and reflection on the limitations of AI be explicitly included in the learn-

ing content in order to prevent the development of possible competence deficits [4, 12, 14].

2. Research questions

Since there have been no studies to date on the learning effectiveness of generative AI chatbots in teacher training in the context of technical education, there is a need for further research. It remains to be clarified how AI chatbots can be integrated into the learning processes of teacher training students in such a way that they promote competence development.

In order to address this question, it seems necessary to first examine whether teacher training students who learn programming intuitively with the help of ChatGPT actually acquire only limited programming skills. In addition, it is necessary to investigate how students formulate their prompts for generating program code and how the quality of these prompts relates to their programming skills and the quality of the resulting code. On this basis, the following two research questions are formulated:

1. *How do students formulate prompts in ChatGPT to obtain program code?*
2. *How does the formulation of prompts relate to programming skills and the quality of code functionality?*

3. Pilot study

Task

In February 2025, a pilot study was conducted in a technology class (secondary level I) in which teacher training students (N = 13), most of whom had little or no programming knowledge, developed their own projects. The task was to design and implement a technical system using an Arduino Uno R4 in the online application TinkerCAD. This system was to contain at least two sensors and two actuators and implement corresponding measurement, control, and regulation functions.

An example model (Fig. 1) includes level measurement using an ultrasonic sensor, an LED display in traffic light logic, and the control of a DC motor via a transistor switch depending on the measured fill level, whereby the motor is switched via a relay controlled by the transistor so that the Arduino only supplies the control current and the motor current flows separately via the 9 V supply. The LEDs visualize three states: green signals a non-critical range (small distance, high fill level), yellow signals a warning range (increasing distance, decreasing fill level), and red signals a critical range (large distance, low fill level) in which the motor—for example, as a drive for a water pump—is activated. In addition, a blue status LED is provided to indicate active pump operation independently of the traffic light display, for example at a spatially separate location as an operating indicator. The distance, which is visualized to the right of the ultrasonic sensor in the model by a blue marker, is varied manually in the learning environment to simulate different fill levels. Participants were expressly permitted to use ChatGPT and other online resources to create the corresponding programming code. Most students used the free version of ChatGPT. They had access to GPT-4o (with usage limits) and o3-mini; when the limits were exhausted, GPT-4o mini could be used.

While working on the tasks, the students kept a logbook in which they documented prompts addressed to ChatGPT.

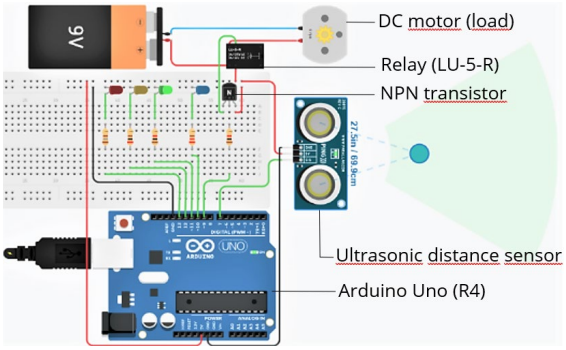

Fig. 1: TinkerCAD simulation of an Arduino Uno project for distance measurement and motor control

4. Method

The pilot study followed a mixed-methods approach. In addition to analyzing the code created and the ability to understand the logic of this code, the prompts documented by the students were evaluated. Questionnaires were also used to collect self-reported information on programming and ChatGPT knowledge.

The study primarily aims to record performance indicators during task completion (product- and comprehension-oriented aspects of programming ability) and not to quantify learning gains in the sense of a before-and-after comparison. Statements about "competence acquisition" therefore refer to evidence from observable understanding, explanation, and modification in the task process.

Analysis of self-reported prior knowledge

To assess the students' prior knowledge, their self-reported information on their use of OpenAI and programming skills was collected. Two people were excluded from further analysis because they stated in their self-reported information that they had advanced programming skills, and the study was aimed at novices.

The self-assessment test used covers the dimensions (1) "Fundamentals of programming" and (2) "Working with OpenAI or ChatGPT." The first dimension covers aspects such as creating simple block- or text-based programs (e.g., "I can create simple text-based programs"), understanding basic structures such as loops, conditions, variables, as well as debugging and test runs. The second dimension covers self-assessment of understanding and use of ChatGPT. It includes knowledge of how it works and its limitations, the ability to formulate prompts, assessment of response quality, and reflective application in different contexts (e.g., "I use systematic methods, such as step-by-step explanations, to get better answers").

Participants rated their abilities on a five-point Likert scale (1 = uncertain to 5 = confident). For further analysis, the five response levels were first transformed to a linear scale from 0 to 1 (unsure = 0; somewhat unsure = .25; neutral = .5; somewhat sure = .75; sure = 1). On this basis, the mean values of both dimensions were determined.

Analysis of prompt formulations

Based on the established Five S framework model [8], a category system was used to analyze the prompts. It comprises a total of seven dimensions that systematize key recommendations for effective prompting with AI systems such as ChatGPT:

- *Context:* Strategies for establishing a content-related or social frame of reference. These include embedding situations and formulating contextualized or orienting prompts [18]. In addition, role and persona assignments [5, 17] as well as the integration of examples and external information can increase the chatbot's connectivity [19].
- *Precision:* Techniques for linguistic and thematic refinement. This refers to explicit expressions, linguistic simplifications, and the concretization of questions [8].
- *Structuring:* Measures that promote a clear and logical structure of AI outputs. These include step-by-step instructions, strategies for dealing with ambiguities, and techniques such as "zero-shot chain-of-thought" (e.g. “Think step by step”), which stimulate explicit thought structures [6, 19].
- *Meta-organization:* Higher-level control mechanisms in the prompting process. Examples include differentiating between task types, defining learning styles, creating your own format logic, or requesting specific visualizations [6, 17].
- *Iteration:* Forms of sequential prompting in which subsequent inputs follow on from previous outputs in a targeted manner. These include "prompt chaining," "dialogic prompting," and "flipped interaction," in which responses are developed step by step and converted into a connectable sequence. Requesting feedback also falls into this category [5].
- *Normative-ethical aspects:* Cross-cutting issues such as ethical questions or the tension between user intention and AI-generated creativity [5].
- *Ineffective prompting:* Typical forms of prompting that are ineffective, such as unclear wording, unrealistic expectations of AI, or excessive detail [17].

In total, the category system comprises 28 individual categories, which are distributed across the seven dimensions as follows: Context (7), Precision (5), Structuring (4), Meta-organization (4), Iteration (3), Ineffective Prompting (3), and Normative-ethical aspects (2) (see Appendix 1).

The prompts were evaluated on a criteria-oriented basis using a dichotomous scale, with a

code being assigned to each category (1 = applies, 2 = does not apply). No exclusive assignment to only one category was made. The frequencies of the codes per category were then calculated.

Interrater reliability

An interrater reliability analysis was performed to verify the reliability of the coding. For this purpose, 50% of the data was double-coded by two independent raters using the category scheme. To determine the overall consistency, a global kappa value (*k*) according to Cohen [20] was calculated across all double-coded units. According to Landis and Koch [21], the resulting value $k = 0.76$ indicates substantial agreement between the raters.

Analysis of programming ability

In this study, programming ability is understood as the competence not only to create functional code or have it generated by ChatGPT and apply it, but also to understand and explain its logic [11]. This construct was divided into two sub-aspects:

(1) *Product-oriented* through the programs created by the students in TinkerCAD, which provide information about syntactic correctness and functional feasibility.

(2) *Understanding-oriented*, since pure code creation does not necessarily reflect underlying programming ability [22]. To this end, students were asked to explain how their code works and the relationships between individual lines and sections of code. If their program codes contained errors, these were to be identified and corrected.

This was intended to reveal any possible discrepancy between "functional results" and "conceptual understanding." Limitations in explaining and modifying were interpreted as indications of limited comprehension-oriented competence (development) in the task process.

5. Results

Self-assessment of prior knowledge

The participants' self-assessments show that, on average, they had moderate prior experience with OpenAI applications ($M = .77$; $SD = .19$)[1] , while they rated their programming skills as low ($M = .34$; $SD = .30$).

Research question 1

Analysis of the prompts created by students shows that the prompting principles recommended in the literature were rarely implemented. These include, in particular, the *Five S strategies* according to Tassoti [8] and other best practices of prompt engineering [5, 6, 17, 18]. Across all codings, only about

- 11% of the observed categories met the criteria of the respective principles, while in
- around 89% of cases no application of these principles could be identified. Instead, a one-time prompt entry by the students followed by the adoption of the generated program code predominated in most cases.

Since the participants were not introduced to these techniques, prompting was probably largely intuitive.

Research question 2

The results of the pilot study also show that the code generated and adopted by the students using ChatGPT was generally functional and controlled the models correctly. However, significant difficulties arose when adjustments were necessary and the learners had to modify the code themselves.

Overall, the students found it difficult to keep track of the code and explain the logical connections between individual lines or sections.

In relation to the operationalization of programming ability (product- and understanding-oriented) defined in Chapter 4, this reveals a discrepancy: while the product-oriented results are successful, the understanding-oriented component remains limited in many cases.

6. Discussion

The results observed in this pilot study suggest that the potential of ChatGPT was often exploited at the expense of actively engaging with programming content. This could have led to a passivization of the learning process and limited the ability to analyze and correct errors: Many participants were unable to identify and fix the flaws in the generated code.

[1] *M* = mean value; *SD* = standard deviation

This confirms that although ChatGPT can serve as a supportive tool in programming instruction, it is not a substitute for established didactic concepts for novices [2]. Accordingly, the finding should not be interpreted as a "deficit" on the part of the students, but rather as a possible consequence of a lack of instruction. However, it is relevant for the classification of the (limited) acquisition of skills that the observed style of using ChatGPT reduced the number of learning opportunities in which learners had the code explained step by step, varied it in a reasoned manner, diagnosed errors, or improved it iteratively.

The use observed in this study carries the risk of reinforcing dependencies on external knowledge rather than building skills. This highlights the need to systematically promote prompting as a skill in its own right.

However, the literature does not consistently support this assumption: Sagar [17] reports cases in which learners without a structured prompting framework achieved better results than learners who learned with a framework. Possible reasons for this could lie in the suitability of the framework used or in the different learning strategies of the participants. Studies in the field of physics education also show that the quality of ChatGPT outputs depends heavily on the prompting used and can be significantly improved through targeted techniques [6]. Further research should therefore clarify how frameworks need to be designed in order to productively support different learning strategies.

For teacher training in the subject of technology, this means that not only the possibilities but also the didactic limitations of generative AI must be taken into account in a reflective manner. The development of prompt competence should be established as a learning objective. In addition, it makes sense to establish a progression that leads from basic handling to the analysis of external solutions to the independent development and optimization of prompts.

Without targeted guidance, there is a risk that AI assistance will inhibit independent activity and competence development. Embedding it in didactically designed learning environments can counteract this by promoting problem-solving skills, strategic approaches, and knowledge acquisition. Future research should therefore investigate how AI-supported programming assistance can be integrated into learning processes in such a way that these beneficial effects are actually realized.

7. Limitations

The significance of this pilot study is limited by several restrictions.

First, the sample size is small (N = 13) and limited to teacher training students in a specific context (Arduino Uno in TinkerCAD). This calls into question the transferability of the findings to other target groups, learning environments, or real hardware setups.

Second, there was no comparison condition with, for example, an introduction to effective prompting strategies as a varying factor. This makes it impossible to determine conclusively whether the difficulties observed are due to the use of AI itself or to the lack of didactic framing when using such tools.

Third, the analysis of prompt quality was based on a dichotomous coding scheme (1 = true, 2 = false). This binary approach reduces gradual differences and can level out subtle qualitative gradations in prompt quality.

Fourth, human-AI collaboration in this study was primarily realized as "AI as a teammate" (in the sense of a counterpart). Other modes of collaboration, in particular highly integrated co-working approaches in the sense of a "cyborg" mode [23] in which humans and AI work in an interlinked and iterative manner, were not implemented. Therefore, in view of the Five-S framework model [8] outlined in this paper, it should be noted that this framework model was primarily developed for classic chatbot interactions. In other models, e.g., with explicit reasoning mechanisms, identical prompts can trigger different response strategies; the inconsistency in the literature discussed in Chapter 6 should therefore be interpreted against this background.

The findings of this work should be understood as trends and provide initial indications of prompting practices and challenges in AI-supported programming learning by teacher training students in the subject of technology.

Their confirmation and generalization require further studies with larger and more heterogeneous samples as well as controlled designs in order to systematically examine the observed correlations and capture them in a more differentiated manner. The type of human-AI collaboration (e.g., *centaur* vs. *cyborg*) could also be taken into account as an experimental factor.

Author contributions

Igor Gideon:
Conceptualization, methodology, formal analysis, investigation (data collection), data curation, writing of the original draft, visualization.
Nico Link:
Conceptualization, methodology, review and revision (review & editing), supervision.
The DeepL tool (version 25.11.4) was used for linguistic revision; the authors are responsible for the content.

Literature

[1] Skalka, J., Drlik, M., Benko, L., Kapusta, J., Rodriguez del Pino, J. C., Smyrnova-Trybulska, E., … Turcinek, P. (2021). Conceptual framework for programming skills development based on microlearning and automated source code evaluation in virtual learning environment. Sustainability, 13(6), 3293. https://doi.org/10.3390/su13063293

[2] Groothuijsen, S., van den Beemt, A., Remmers, J. C., & van Meeuwen, L. W. (2024). AI chatbots in programming education: Students' use in a scientific computing course and consequences for learning. Computers and Education: Artificial Intelligence, 7, 100290. https://doi.org/10.1016/j.caeai.2024.100290

[3] Sun, D., Boudouaia, A., Zhu, C., & Li, Y. (2024). Would ChatGPT-facilitated programming mode impact college students' programming behaviors, performances, and perceptions? An empirical study. International Journal of Educational Technology in Higher Education, 21(1), 14. https://doi.org/10.1186/s41239-024-00446-5

[4] Leifheit, L., Loefflad, D., Belschner, S., Beuttler, B., Winkelmann, J., Meurers, W. D., & Holz, H. (2024). KI im Unterricht: Entwicklung von Lehrveranstaltungen für Lehramtsstudierende der Sprach- und MINT-Fächer. Ludwigsburger Beiträge zur Medienpädagogik, 24, 1–19. https://doi.org/10.21240/lbzm/24/08

[5] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. arXiv. https://doi.org/10.48550/arXiv.2302.11382

[6] Polverini, G., & Gregorcic, B. (2024). How understanding large language models can inform the use of ChatGPT in physics education. European Journal of Physics, 45(2), 025701. https://doi.org/10.1088/1361-6404/ad1b3a

[7] Greco, C. M., & Tagarelli, A. (2024). Bringing order into the realm of Transformer-based language models for artificial intelligence and law. Artificial Intelligence and Law, 32(4), 863–1010. https://doi.org/10.1007/s10506-023-09372-x

[8] Tassoti, S. (2024). Assessment of students' use of generative artificial intelligence: Prompting strategies and prompt engineering in chemistry education. Journal of Chemical Education, 101(6), 2475–2482. https://doi.org/10.1021/acs.jchemed.3c01069

[9] Malik, S. I., Ashfque, M. W., Tawafak, R. M., Al-Farsi, G., Usmani, N. A., & Khudayer, B. H. (2022). A chatbot to facilitate student learning in a Programming 1 course: A gendered analysis. International Journal of Virtual and Personal Learning Environments, 12(1), 1–20. https://doi.org/10.4018/IJVPLE.310007

[10] Yılmaz, R., & Yılmaz, F. G. K. (2023). Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. Computers in Human Behavior: Artificial Humans, 1(2), 100005. https://doi.org/10.1016/j.chbah.2023.100005

[11] Wang, T., Díaz, D. V., Brown, C., & Chen, Y. (2023). Exploring the role of AI assistants in computer science education: Methods, implications, and instructor perspectives. In 2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) (pp. 92–102). IEEE. https://doi.org/10.1109/VL-HCC57772.2023.00018

[12] Rahman, M. M., & Watanobe, Y. (2023). ChatGPT for education and research: Opportunities, threats, and strategies. Applied Sciences, 13(9), 5783. https://doi.org/10.3390/app13095783

[13] Chen, E., Huang, R., Chen, H. S., Tseng, Y. H., & Li, L. Y. (2023). GPTutor: A ChatGPT-powered programming tool for code explanation. In International Conference on Artificial Intelligence in Education (pp. 321–327). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-36336-8_50

[14] Samek, W., & Schmid, U., et al. (2025). Nachvollziehbare KI: Erklären, für wen, was und wofür (Whitepaper der Plattform Lernende Systeme). Plattform Lernende Systeme / acatech. https://doi.org/10.48669/pls_2025-2

[15] Beyerer, J., Kirchner, E., André, E., Behnke, S., Bloss, G., Dzaack, J., Egloffstein, T., Griepentrog, H. W., Gustmann, M., Hafner, V. V., Koert, D., Nüchter, A., Seyler, J., Straube, S., Tchouchenkov, I., von Stryk, O., Wedler, A., Wolf-Ostermann, K., & Zimmermann, M. (2025). KI in der Robotik: Flexible und anpassbare Systeme durch interaktives Lernen (Whitepaper der Plattform Lernende Systeme). Plattform Lernende Systeme / acatech. https://doi.org/10.48669/pls_2025-1

[16] Stoyanova, D., Stoyanova-Petrova, S., & Mileva, N. (2025). Exploring students' and teachers' perceptions about using ChatGPT in programming education. International Journal of Engineering Pedagogy, 15(2).

[17] Sagar, P. (2024). Exploring the use of ChatGPT and the prompting framework as a self-learning aid for Arduino coding & circuit building for artists and designers (Doctoral dissertation, Massachusetts Institute of Technology). MIT Libraries. https://hdl.handle.net/1721.1/151432

[18] Ekin, S. (2023). Prompt engineering for ChatGPT: A quick guide to techniques, tips, and best practices. TechRxiv. https://doi.org/10.36227/techrxiv.22683919.v1

[19] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Advances in Neural Information Processing Systems (NeurIPS 35) (pp. 24824–24837). https://proceedings.neurips.cc/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[20] Cohen, J. (1960). A coefficient of agreement for nominal scales. Educational and Psychological Measurement, 20(1), 37–46. https://doi.org/10.1177/001316446002000104

[21] Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. Biometrics, 33(1), 159–174. https://doi.org/10.2307/2529310

[22] Röhner, G. (2016). Bildungsstandards Informatik für die Sekundarstufe II (Januar 2016). Gesellschaft für Informatik e. V. https://doi.org/10.18420/rec2016_057

[23] Dell'Acqua, F., McFowland III, E., Mollick, E. R., Lifshitz-Assaf, H., Kellogg, K., Rajendran, S., ... & Lakhani, K. R. (2023). Navigating the jagged technological frontier: Field experimental evidence of the effects of AI on knowledge worker productivity and quality. Harvard Business School Technology & Operations Mgt. Unit Working Paper, (24-013).

Appendix

Appendix 1. Overview of the "prompting principles" category system with its own assignment to the seven dimensions.

Nr.	Category	Authors	Subcategory	Description
1	Context	Prompt strategies (Five S, Tassoti 2024)	Set the Scene (provide context)	Is a specific background or situation described?
2	Context	Prompt engineering (Giray, 2023)	Contextual prompts	Does the prompt include additional background information to focus the response on specific aspects?
3	Context	Prompt components (Polverini & Gregorcic, 2024; Sagar, 2024)	Assign ChatGPT a persona	Is ChatGPT assigned an explicit role to influence the depth and context of the responses?
4	Context	Prompt components (Sagar, 2024)	Define your persona	Is it explicitly stated what role or what level of expertise the questioner has?
5	Context	Prompt engineering (Ekin, 2023) - techniques	Illustration through examples	Are examples used to increase the precision of the task statement?
6	Context	Prompt engineering (Polverini & Gregorcic, 2024)	Few-shot CoT	Are examples used to illustrate how tasks should be solved?
7	Context	Prompt engineering (Ekin, 2023) - best practices	Integration of external data	Are external sources or data explicitly integrated into the prompt?
8	Precision	Prompt pattern catalog (White et al., 2023)	Motivation	Does it specify what the prompt is intended to achieve?
9	Precision	Prompt strategies (Five S, Tassoti 2024)	Simplify your language	Does it request simple, easy-to-understand language?
10	Precision	Prompt components (Sagar, 2024)	Elaborate topic of interest	Is the learning objective clearly stated?
11	Precision	Prompt engineering (Ekin, 2023; Giray, 2023; Five S, Tassoti 2024)	Specific instructions	Are misunderstandings avoided through explicit, specific instructions?
12	Precision	Prompt pattern catalog (White et al., 2023)	Question refinement	Does the prompt explicitly ask for refinement or optimization of the questions posed?
13	Structuring	Prompting strategies (Polverini & Gregorcic, 2024; Tassoti 2024)	Structure the output	Is a structured, step-by-step, or otherwise organized answer requested?
14	Structuring	Prompt pattern catalog (White et al., 2023)	Output automater	Does the prompt refer to specific response patterns?
15	Structuring	Prompt engineering (Ekin, 2023) - techniques	Control of verbosity	Does it explicitly ask for a particular level of detail or brevity?
16	Structuring	Prompt engineering (Ekin, 2023) - advanced strategies	Dealing with ambiguity	Are explicit strategies used to clarify ambiguous inputs?
17	Metorganization	Prompt components (Sagar, 2024)	Define your learning style	Is a specific method of presenting the information explicitly requested?
18	Metorganization	Prompt engineering (Ekin, 2023) - techniques	Differentiating task type (meta)	Does it explicitly distinguish between simple factual questions and complex analytical tasks?
19	Metorganization	Prompt pattern catalog (White et al., 2023)	Meta language creation	Is a user-defined notation/abbreviation introduced to simplify complex matters?
20	Metorganization	Prompt pattern catalog (White et al., 2023)	Visualization generator	Does the prompt explicitly request text-based visualization instructions or a visual output?
21	Iteration	Prompting strategies (Five S, Tassoti 2024)	Share feedback	Does it explicitly ask for feedback or the correctness of results?
22	Iteration	Prompt engineering (Ekin, 2023; Polverini & Gregorcic, 2024)	Prompt chaining	Are multi-step, iterative interactions used?
23	Iteration	Prompt pattern catalog (White et al., 2023)	Flipped interaction	Is ChatGPT instructed to actively ask questions itself in order to gather required information?
24	General	Prompt engineering (Ekin, 2023) - Best practices	User intent vs. creative freedom	Is a balance between user intent and creative AI output apparent?
25	General	Prompt engineering (Ekin, 2023) - Best practices	Ensuring ethical use	Is there an explicit effort to ensure ethical principles are followed?
26	Negativ		Assumption of omniscience	The prompt suggests that OpenAI knows everything, including what is left unsaid.
27	Negativ		Assuming error diagnosis without sufficient	Information provided is insufficiently detailed.
28	Negativ		Excessive details	The prompt asks for overly specific things.