Hey, ChatGPT, Look at My Work: Using Conversational AI in Requirements Engineering Education

Sahar Badihi
shrbadihi@ece.ubc.ca
Univ. of British Columbia, Canada

Michael Tegegn
mtegegn@ece.ubc.ca
Univ. of British Columbia, Canada

Evelien Riddell
evelien.riddell@uwaterloo.ca
Univ. of Waterloo, Canada

Krzysztof Czarnecki
krzysztof.czarnecki@uwaterloo.ca
Univ. of Waterloo, Canada

Julia Rubin
mjulia@ece.ubc.ca
Univ. of British Columbia, Canada

Abstract

The emergence of conversational AI tools in late 2022 practically changed the face of software engineering and software engineering education. Contemplating the question of how to best prepare and evaluate students in this new reality, we experimented with systematically introducing a conversational AI tool, ChatGPT, into the 2023 offering of an upper-level undergraduate project-based course on Software Engineering. In this course, 20 groups of four students each had to design and implement a project of their choice, with an Android-based mobile client and a Node.js-based cloud server. This paper discusses our goals, approach, and lessons learned from introducing ChatGPT into the first phase of the project development: scoping the work and defining the project requirements.

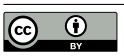
Students were allowed to use ChatGPT at any stage of their project, albeit in a controlled way, to fulfill the educational objectives of the course. Our experience shows that students can achieve comparable results using a variety of ChatGPT interaction modes and the success of each mode largely depends on students' preferences, learning styles, and the invested effort. Yet, in any of the modes, with moderate effort, students can produce artifacts of a mid-range quality level of around 80%. Moving above this range requires substantial investment, which can be spent on brainstorming, crafting high-quality prompts, or critically assessing ChatGPT's output. We also observe low prompting proficiency of the students: students can improve their prompting strategies by providing a more adequate description of their course and project setup, examples, and expected output format for their requests. Interestingly, students can often be "swayed" by ChatGPT's projected confidence, even when their original ideas are, in fact, more appropriate than the proposed refinements.

CCS Concepts

• **Software and its engineering → Requirements analysis**; • **Applied computing → Education**.

Keywords

Requirements engineering, software engineering, generative AI, conversational AI, ChatGPT, education



ICSE '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2025-3/26/04
https://doi.org/10.1145/3744916.3773112

ACM Reference Format:
Sahar Badihi, Michael Tegegn, Evelien Riddell, Krzysztof Czarnecki, and Julia Rubin. 2026. Hey, ChatGPT, Look at My Work: Using Conversational AI in Requirements Engineering Education. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3744916.3773112

1 Introduction

The last few years have been turbulent in the field of computing education, when conversational Artificial Intelligence (AI) agents, such as ChatGPT [1] and GitHub Copilot [3], introduced new opportunities and challenges to instructors and students [36, 37, 39, 43]. As "resistance is futile" [39], in this paper, we report on our experience systematically integrating conversational AI into the 2023 offering of an upper-level undergraduate project-based course on Software Engineering. The students taking this elective course are in their third to fifth year of studies. They completed at least two prerequisite programming courses and other fundamental computer science courses, such as algorithms and data structures. About 60% of students taking the course have prior internship experience in industry, either independently or through a co-op program.

In the scope of the course, 20 groups of 4 students each had to design and implement a large-scale project of their choice, with an Android-based mobile client and a Node.js-based cloud server. They used iterative development and were tasked to produce multiple deliverables, including project scope and its requirements, design, project implementation, integration and system tests, manual and automated code review results, and project documentation. Students were allowed to use ChatGPT as an assistant in completing their assignments, such as preparing requirements and design, albeit in a controlled way, to fulfill the pedagogical objectives of this course offering: assess the pros and cons of using AI tools in software engineering processes and instill critical thinking when working with these tools.

For this paper, we focus on the first phase of the project development: scoping the work and defining the project requirements [20]. We designed two alternative processes for students to use ChatGPT for this task. As a start, we asked each group to come up with two possible ideas for their course project. Then, for one of these ideas picked by a teaching assistant (TA) at random, students were asked to spend one and a half hours during a lab session to brainstorm the task and manually specify the initial set of requirements (as they typically did in all prior offerings of the course). After the lab, students were asked to use ChatGPT to refine and update this

initial specification. We refer to this process as *Process A*. For the second project idea, the students were asked to work with ChatGPT from the start, without manually specifying the requirements first. We refer to this process as *Process B*. For both processes, in addition to the final deliverable, students were asked to submit their intermediate artifacts and all conversations with ChatGPT.

For the results reported in this paper, we grade and analyze all requirements specifications produced by students in both processes: the initial requirements specification produced in the lab for Process *A*, the output produced after ChatGPT interactions in both processes, and the final submissions in both processes (only the final submissions were graded in the scope of the course). The goal of this analysis is to investigate which of the processes, *A* or *B*, leads to higher-quality requirements specifications.

It is well-established that the quality of ChatGPT output depends on the quality of the prompts [52, 67]. As such, we also aim to put this analysis in context of the prompting strategy students applied and the refinements they performed on the output obtained from ChatGPT. To this end, we set a grading metric for ChatGPT prompts, deriving it from OpenAI guidelines [47] and existing literature, e.g., [13, 38, 62, 64, 66]. We use the metric to assess the quality of the prompts produced by the students, categorize different modes in which students interact with ChatGPT, and discuss the strengths and areas of improvement in their prompting strategies.

Furthermore, to assess the ability of ChatGPT to produce high-quality requirements specifications, we construct an experiment that replicates the students' work using an improved prompting strategy. We assess the effort required to produce such improved prompts and compare it with the effort self-reported by students in their submissions. Finally, we analyze the students' self-reported reflections on the use of ChatGPT for this task.

Our analysis is driven by the following research questions: **RQ1**: Which process, *A* or *B*, results in better requirements specifications? **RQ2**: What effort is required to produce high-quality artifacts in each process? **RQ3**: How do students utilize ChatGPT in their work? Our results show that:

1. Both processes *A* and *B* can be equivalently effective. While Process *A* increases the students' ability to critically assess the results produced by ChatGPT, it is also associated with a decreased quality of prompts in ChatGPT discussions, mostly because students provide less clear instructions, mistakenly assuming the initial specs they provide are sufficient. Overall, students can improve their prompting strategies by providing a more adequate description of their course and project setup, examples, and expected output format for their requests. Moreover, students can be "swayed" by ChatGPT's confidence, even though the artifacts they initially produced are more correct than fixes proposed by ChatGPT; it is thus important to provide them with evidence-based education about the need to analyze ChatGPT results more critically, which often increases their ability to produce high-quality outcomes.
2. While on average students could achieve similar results in both processes, obtaining a high-quality output requires an investment of time and effort, in terms of time spent in initial brainstorming, crafting high-quality prompts, critically assessing ChatGPT's output, and refining the results. The majority of this effort is spent to achieve the improvement beyond the relatively-easy-to-achieve generated requirements quality of around 80%.
3. Students perceive ChatGPT as a useful tool for ideation, refinement, and formal writing, with ChatGPT's suggestions for the last two use modes receiving the highest adoption rate, especially for students with at least partially formed initial ideas. At the same time, students are concerned about over-reliance on ChatGPT. Some students indicate a clear preference for having preliminary brainstorming and discussions while others prefer to work with ChatGPT directly. We believe different modes of work can be beneficial for different student personality types.

Data Availability and Ethics Approval. The full description of the course setup, our grading rubrics, the aggregated statistical data collected from student projects, and all data used in our own experiment are available in our online appendix [5]. The study of student projects was approved by the ethics board in our institution.

2 Course Setup

The Software Engineering course discussed in this paper was taught in the Fall 2023 term (September-December) at the University of British Columbia (UBC). In what follows, we outline the structure of the course, its requirements specification milestone, and the use of ChatGPT in the context of this milestone.

Course Format. The main goal of this elective course is to teach core Software Engineering principles required for building non-trivial software-intensive systems. Students can take the course starting from their third year of undergraduate studies, after completing at least two prerequisite programming courses and several fundamental computer science courses, such as algorithms and data structures. Many also complete at least one internship or co-op term, during which they acquire professional, paid work experience. Students in our university can take up to five co-op terms overall, resulting in a five-year-long undergraduate program.

The course uses a term-long development project as the main learning vehicle. Working in groups of four, students develop a client-server software system of their choice, with Android-based mobile client and a Node.js-based cloud server. Each project must contain certain course-required features, such as third-party authentication, and project-specific features defined by the students themselves. The students are expected to scope their work and define project requirements (the focus of this paper), provide a high-level design of their project, its implementation, integration- and system-level tests, results of the automated and manual code review, and documentation.

This offering of the course had 88 students, with 70% of the course cohort in their fourth or fifth year and 60% of the course cohort having completed at least one co-op term. Thus, the majority of the students already have fundamental programming experience and take the course to learn good software engineering principles. Per the guidelines of the ethics board, the students were given the option to opt out from using their project data in our study. Two of the 22 groups chose to opt out. Thus, the results of this study are based on the work of 20 groups (80 students).

Requirements Specification. In the requirements specification, students are asked to define the scope and specify the main functionality and constraints of the system they wish to build [58]. We asked students to provide both functional and non-functional requirements for their projects (FRs and NFRs, respectively).

FRs describe the system functionality, i.e., what the system should do (rather than how). For this course, students were asked to capture requirements in form of (1) a use case diagram [29] with 5-6 clearly defined non-trivial use cases and 1-3 actors. Then, for each use case, the students were asked to provide (2) a more descriptive specification that includes a list of success and failure scenarios [7], where success scenarios describe the normal flow of events and failure scenarios describe what can go wrong and how to mitigate failures in each step of the success scenario.

NFRs describe system properties and constraints related to performance, safety, security, scalability, usability, etc. Students were asked to describe 2-3 main NFRs for their project in plain text, together with the justification for why each requirement was needed and how it could be validated.

The goals of this milestone were both to familiarize the students with the concepts of requirements elicitation, analysis, and specification, and to have them think through the details of their proposed project before they embark on its implementation. Due to the iterative nature of the project, students were also encouraged to revise and refine their requirements throughout the term, as the implementation progressed and new information and constraints emerged. Yet, in the scope of this paper, we focus only on the initial specification they produced in the requirements milestone.

The requirements were graded based on the following attributes, inspired by the course guidelines and existing literature [6, 58]: 1. *Complete*, i.e., cover all parts of the project goals, providing sufficient information to implement the system. 2. *Consistent*, without having contradictory definitions across multiple requirements. 3. *Unambiguous*, i.e., free of multiple, potentially conflicting interpretations of a requirement. 4. *Focused*, i.e., defined at the right level of detail (not too coarse-grained or too fine-grained). 5. *Relevant* to the project at hand (rather than applicable to any project). 6. *Feasible* to implement given the time/resource constraints. 7. *Verifiable/Measurable*, i.e., where it is possible to clearly define corresponding test cases. 8. *Correctly classified as FR vs. NFR*, i.e., FRs are not classified as NFRs and the other way around. 9. *Well-formatted*, with appropriate notations of use case diagrams, success and failure scenarios, etc. The full description of the requirements specification approach used in the course, our evaluation criteria, and our full grading rubric are available online [5].

Use of ChatGPT. One of the educational objectives of this course is to expose students to the advantages and disadvantages of using AI tools in Software Engineering processes. As such, throughout the course, students were allowed to use ChatGPT as an assistant in completing their assignments. We did not teach or enforce any specific prompting strategy, and students could use the tool in any way they preferred. Yet, they were required to critically analyze advantages and disadvantages of the tool and submit the results of their analysis together with the milestone artifacts. Students were also asked to submit their full chat history, which was not graded.

For the requirements milestone, students were instructed to use ChatGPT 3.5, which was the publicly available free version at the time the course was given. This restriction was necessary to ensure a consistent and fair experience. To facilitate our and students' critical analysis of the utility and outputs of the tool, we designed two working modes, referred to as Process A and B (see Figure 1).

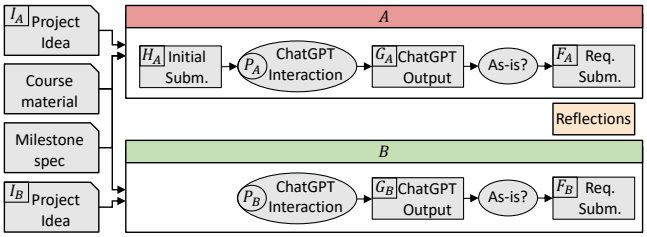


Figure 1: Processes A and B.

Process A relies on a thoughtful consideration of the project scope and requirements prior to ChatGPT interaction, while in Process B students started directly from interacting with the tool.

To enable efficient comparison of the processes, each group of students was first asked to come up with two possible ideas for a project they might want to implement within the scope of the course (one per process). To avoid any bias that may arise from personal preferences of students when picking which idea to work on first, we asked half of the groups to use process A to work on their first idea and process B to work on their second idea. The other half of the groups used process A to work on their second idea and process B to work on their first idea. That is, each group delivered requirements specifications for both of their ideas (40 requirements specification documents in total). However, we controlled for which of their ideas they developed with each process. We denote by I_A the project idea for which the student group followed process A and by I_B – the project idea for which they followed process B.

The students started to work on I_A in a two-hour-long course lab session, where they produced requirements specifications without using any AI or other external tools, as was typically done in pre-ChatGPT offerings of the course. Throughout the session, course TAs provided initial instructions and were available to answer questions the groups had. In total, the students worked on the requirements specification of I_A for 1.5 hours and were asked to submit the version they produced by the end of the lab session. We refer to this version as H_A (see the upper part of Figure 1; H stands for 'Human'). This intermediate submission was not graded in the scope of the course, but we later graded it for this study.

After the end of the lab session, students were given 10 days to complete their requirements specifications for both I_A and I_B. Specifically, they were asked to use ChatGPT to further improve H_A through a series of discussions. They had to submit all their conversations with ChatGPT, which we refer to as P_A. Students could choose to stop interacting with ChatGPT at any point of this step, if they believed the interactions did not lead to any meaningful improvement. We refer to the requirements specifications we extracted from these conversations as G_A (G stands for 'GPT'). This artifact was also not graded in the scope of the course, but we graded it for the discussion provided in this paper. As the last step in this process, students were asked to produce and submit their final requirements specification for I_A, which we refer to as F_A (F stands for 'Final').

Additionally, students developed requirements for their I_B (lower part of Figure 1). Unlike in process A, they interacted with ChatGPT without having a preliminary discussion session, but could further refine ChatGPT results. We refer to artifacts produced in this process as P_B (for ChatGPT conversations), G_B (for the requirements

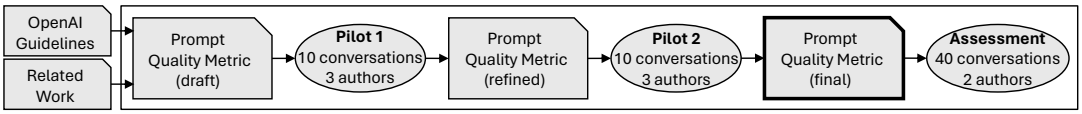


Figure 2: Deriving the prompt quality assessment metric.

specifications produced through ChatGPT interaction), and F_B (for the final submission of their I_B requirements specifications, which was graded in the scope of the course).

To summarize, the main difference between the processes is that process A involved an in-class TA-facilitated session, where students had to brainstorm and draft their project scope and requirements prior to ChatGPT interaction. No such session was present in B and students could start directly from interacting with the tool (as we assumed many would do if not explicitly encouraged to meet and brainstorm). In both cases, TAs were available outside of the lab hours, to answer questions about the course material and the assignment via Piazza [4]. A detailed description of the requirements assignment is available online [5].

3 Study Methodology

We now describe the methodology we followed for answering the research questions outlined in Section 1.

3.1 RQ1: Process A vs. B

Grading Process. Even though in the scope of the course, we only graded the final requirements specifications produced by the students in each process, to answer this research question, two authors of this paper, who were also TAs of the course, graded all intermediate artifacts produced by students, i.e., requirements specifications (H_A, G_A, F_A, G_B, and F_B) and prompts (P_A and P_B).

The authors cross-validated grading for each artifact and all disagreements (5% disagreement rate for requirements artifacts and 6% for prompts) were resolved in a discussion with another author, who served as the primary course instructor. Our full grading rubric, for both requirements and prompts, is available online [5].

Grading Requirements Specifications. We assigned to each of the requirements quality attributes specified in Section 2 a grade on a scale of 0-5. To this end, for each quality attribute, we extracted a set of issues, such as missing or incomplete descriptions, logical or factual errors, and more. The full list of issues is available in our online appendix [5]. We assigned a severity for each issue: from 0 – absent, to 1 – rare, 2 – moderate, 3 – abundant. We further added and normalized the issue severity ranks for a quality attribute, producing a 0-5 scale attribute score (the higher the better). Finally, we averaged the attribute scores for all nine quality attributes to produce the final grade for a requirements specification, giving completeness a double weight, as incomplete information reduces the likelihood of identifying other issues.

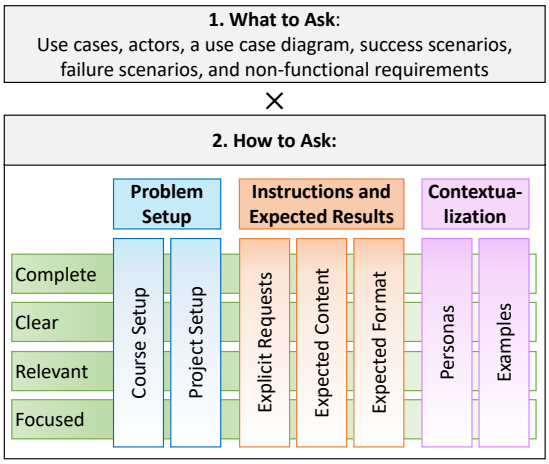


Figure 3: Prompt quality assessment metric.

Grading ChatGPT Prompts. We used the process in Figure 2 to derive a set of quality attributes used for assessing the quality of ChatGPT prompts. We started from the OpenAI prompt engineering guide for GPT models [47] and extensive advice from existing literature on prompt engineering best practices [13, 15, 38, 64].

As existing guidelines are largely high-level and cannot be directly used for assessing conversation quality, comparing conversations with each other, and correlating the quality of conversations with the quality of requirements specifications produced in these conversations, we aimed to turn the guidelines into a concrete metric. To this end, we derived an initial set of attributes and used them to assess 10 randomly selected ChatGPT conversations – five for each process A and B. The assessment was conducted by three authors of the paper independently, with each one collecting their observations and suggestions for improving the attributes. We refined the attributes following these suggestions and conducted a second pilot study by independently grading another 10 conversations selected at random. Only minor revisions were required at this stage and we used the refined set of attributes to grade all 40 conversations.

Figure 3 shows the final set of attributes we derived. They are divided into two parts: the "What to Ask" part contains task-specific attributes. In our case, these are attributes of the requirements specification, i.e., use cases, actors, a use case diagram, success scenarios, failure scenarios, and non-functional requirements.

The second, "How to Ask" part, is mostly agnostic to the task at hand. It has seven attributes grouped into three categories:

1. *Problem Setup*, which includes the *Course* and *Project Setup* attributes, ensures an adequate description of the problem space (e.g., project goal, scope, and time constraints) and course-specific ways to define requirements (i.e., with success and failure scenarios) [47].
2. *Instructions and Expected Results*, which includes *Explicit Requests*, *Expected Content*, and *Expected Format* attributes, focuses on the importance of clearly defining the instruction/question for ChatGPT [15] (vs. just asking "Fix this" or "What do you think", as some of our students did) and specifying the content and format of the expected results [47].
3. *Contextualization*, which includes the *Personas* and *Examples* attributes, helps provide an adequate context and guide the desired style of the ChatGPT-generated output [13, 38, 47, 64].

For each of these categories, we identified four cross-cutting qualities (horizontal rows in the figure), which ensure that each category

is defined in an adequate manner. Specifically: 1. *Complete* means that all the necessary details are provided for each attribute. 2. *Clear* means the information is well-formulated and is free of ambiguities and mistakes. 3. *Relevant* means the information is relevant to the task at hand (vs. asking generic questions about marginally-related topics or providing unnecessary contextual information). 4. *Focused* means that the instructions are fine-grained and specific (vs. asking many questions at once).

As with requirements, when grading, we identified a set of issues for each of the 28 factors (7 attributes × 4 cross-cutting qualities), assessed them based on their severity, and converted the assessment into a numeric score for each factor, on the scale of 0 to 5 (the higher the better). We further averaged the scores across categories, producing the total score for the "How to Ask" (denoted by H).

For the "What to Ask" part, we assessed whether students had at least one request or question related to each of the six requirements components in their chats. We then computed the fraction of requirements components they consulted ChatGPT about. E.g., if the students consulted ChatGPT about actors, use cases, a use case diagram, success scenarios, and failure scenarios, but omitted the non-functional requirements, the assigned score would be 5/6. We denote this score by W.

We compute our final prompt assessment score as $W \times H$, to independently consider both the completeness of the assigned task (W) and the employed prompting strategies (H). That is, our metric makes it possible to distinguish between students who use excellent prompting strategies but only complete part of the assigned task and those who complete the entire task.

Comparing artifacts produced in processes A and B. For Process A, we compare the quality of the students' initial specification made in the lab (H_A), with the specification we extracted from their ChatGPT outputs (G_A), and their final requirements specifications (F_A). For Process B, as no initial specification was made, we compare the quality of G_B and F_B.

In addition, to assess the "trajectory" students followed for arriving at their final requirements specifications and determine to what extent ChatGPT output was used, we compared the ChatGPT outputs (G_A and G_B) with their corresponding final specifications (F_A and F_B). We noted the number of the requirements components (i.e., use cases, actors, a use case diagram, success scenarios, failure scenarios, and non-functional requirements) for which students diverted from the ChatGPT output and deemed groups that diverted in three or more components (50% or above) as "high-editing" groups. We deemed all the remaining groups as "low-editing".

3.2 RQ2: Effort

As the quality of results produced by ChatGPT highly correlates with the quality of the prompt, we aimed to gain further insights into the effort required to complete the deliverable at a high-quality level while interacting with ChatGPT. To this end, we conducted an additional experiment, where we recruited a human expert who was not involved in the course, neither as a student nor as an instructor. In what follows, we refer to this expert as an *investigator*. At the time of the study, the investigator held an undergraduate degree with distinctions in Computer Science from UBC and was a Master's student at the University of Waterloo. The investigator also subsequently became a co-author of the paper.

Subjects. We asked the investigator to use ChatGPT to produce deliverables for four case study subjects systematically selected from the 40 projects in this course offering: two for process A and two for process B.

S_1, S_2: First, we selected two subjects that correspond to projects that received the highest and the lowest H_A grade, out of all specifications produced by the students in the lab. We refer to these subjects as S_1 and S_2, respectively. As a starting point for the interactions with ChatGPT, the investigator used these requirements specifications, as was also done by the course students in process A. Our rationale for selecting the specifications with the highest and the lowest grade was to factor out the quality of the inputs and focus on the prompting effort required to produce high-quality refined requirements specifications with ChatGPT.

S_3, S_4: We further selected two projects for which students received the highest and lowest prompt grades in Process B and, as a result, produced the highest and lowest ChatGPT-generated requirements specifications G_B. We refer to these subjects as S_3 and S_4, respectively. Here, the investigator used as a starting point the high-level project descriptions produced by the students for these projects, as was done by the course students in process B. Our rationale for selecting these subjects was to, again, factor out the quality of the inputs, demonstrating that our prompt generation strategies can consistently lead to high-quality outputs, and to compare student prompts with those of our investigator.

Prompt Crafting Principles. We equipped the investigator with the course material, to simulate the experience of students taking the course, but did not disclose our requirements assessment metric, to avoid biasing the study. However, unlike with the course students, we equipped the investigator with the prompt quality metric and discussed the attributes of good prompts. We asked the investigator to improve the initial specifications for S_1 and S_2 and to produce requirements specifications for S_3 and S_4, as the students of the course did, only using better prompting strategies.

Specifically, we instructed the investigator to focus on crafting well-formulated questions but not engage in any follow-up clarification discussions with ChatGPT, to avoid subjectivity and ensure our experiment is reproducible. As such, we essentially obtained the lower bound on the result that can be achieved with an initial good prompting, without any further clarifications made. Furthermore, to avoid subjectivity, the inputs provided to ChatGPT for the case studies from each process varied only in the initial project description (H_A for subjects S_1 and S_2; I_B for subjects S_3 and S_4), which were used without any modifications.

After reviewing the course project specification (to build a clear understanding of the target system's goals, scope, and constraints), the milestone description (to understand the scope of the assignment), and the lecture notes on requirements engineering (to familiarize themselves with the covered material), the investigator mainly followed the "divide and conquer" principle to design a series of ChatGPT questions.

Specifically, the investigator worked in a hierarchical manner, starting from a high-level description of the personas, and course and project setup. Then, the investigator split the tasks: first, by working on each requirements component individually and then by working on its sub-parts individually ("What to ask"). This was

done to both reduce the scope of each ChatGPT question and to ensure no part of the assignment is overlooked.

For each individual step, the investigator provided all relevant parts of the course and project setup (repeating them, if necessary, for a new question, to ensure sufficient context is provided), expected output content and format, and an explicit request. The investigator also aimed for complete, clear, relevant, and focused requests, as per our prompt guidelines ("How to ask"). For example, after a brief introduction of the course, assignment, and project scope, the investigator instructed ChatGPT as follows: *"Let's start with functional requirements. I will now provide you my initial specification of functional requirements and will ask you to refine them."*. The investigator then further split the question in a hierarchical manner: *"Let's consider one functional requirement at a time. <...>*.

At the end of each step, the investigator asked ChatGPT to combine the information about individual items, producing a complete output: *"Let's put everything together. List all functional requirements for the project using the format described below <...>*.

The prompt templates the investigator used and example conversations for processes A and B are available online [5].

Assessment. All requirements produced by the investigator in this experiment were graded by the TA authors of this paper, who also graded all student artifacts, using the method in Section 3.1. To avoid issues related to the instability of ChatGPT, which are reported by existing work [34, 48] and which we also observed in our study, the investigator repeated the experiment for each subject three times. We observe only minor differences between the results obtained for each subject across the three ChatGPT iterations ($< 2\%$ Std. Dev.) and report the average result. The time invested in these repetitions was not considered for the purpose of this study.

We measured prompting effort by using prompting time and the count of questions and words in a prompt. All students spent 1.5 hours in the lab crafting the original submission in Process A (H_A). We relied on students' self-reported (mandatory) assessment of the time they spent completing requirements specifications for Process A and producing specifications for Process B. Like for students, we rely on the investigator's self-reported time for crafting ChatGPT prompts and producing the necessary deliverables.

We used metrics such as the number of questions asked during a ChatGPT session and the total number of words in these questions, to assess the effort required for providing complete information to ChatGPT. Furthermore, we categorized the words in a prompt by the prompt's quality attributes they contribute to, to identify words that describe Course Setup, Project Setup, etc. To this end, two of the authors of this paper manually inspected students' and the investigator's conversations for subjects S_1-S_4, marking each sub-phrase/word with its purpose. Words that could not be reliably classified, e.g., "Please" and "Thank you", were marked as *Other*. The authors discussed any major disagreements in counts and involved the investigator, when necessary.

3.3 RQ3: Usage Patterns

For the course, students were asked to describe advantages and disadvantages of using Generative AI technology for each of the Software Engineering tasks. In this research question, we analyze the students' self-reflections and interpret their reports.

We further complement the students' reflections with our own analysis that aims to explore how students utilize ChatGPT for their requirements engineering assignment. To this end, we used open coding – a qualitative data analysis technique borrowed from grounded theory [60], to extract patterns students use when interacting with ChatGPT. More specifically, we started from a pilot study where two authors of the paper independently read the prompts of five student groups and identified the goals behind their questions/commands. These two authors and an additional arbiter then met to discuss the identified concepts and any disagreements, refine concept labeling, and merge related concepts, if needed. After agreeing on a coding style, the authors split and assessed the prompts of all 20 groups.

4 Results

4.1 RQ1: Process A vs. B

Requirements Specifications. Figure 4 shows the average score for each of our graded artifacts across all projects and processes. When inspecting students' interactions with ChatGPT, we observed that 7/20 groups in process A did not, in fact, provide their manually-defined requirements specifications (H_A) to ChatGPT (using ChatGPT only as a vehicle to generate new ideas to augment their original specifications). These groups essentially followed a process similar to process B, only after already having their requirements defined manually in the lab. To better understand the impact of this decision, we further split the process A artifacts into those that included the original specifications in conversations (referred to as A^+) and those that did not (referred to as A^-).

Furthermore, we split the groups in each process into high-editing and low-editing, as discussed in Section 3, and report the numbers of such groups in each of the processes (an ellipse with either single or multiple check marks, which appears before the final requirements specification deliverable). We provide the average grades for the deliverable for these two types of groups separately, and then show the aggregated average grade for each process.

We were surprised to see that, overall, students were able to achieve comparable grades in all three processes: 82% for A^+, 75% for A^-, and 77% for B, with no statistically significant difference between the processes (see the online appendix for details [5]). In fact, out of the ten top-ranked projects, five followed Process A (three: A^+; two: A^-) and the other five – Process B. However, when following A^+ and A^- processes, students tended to edit the result produced by ChatGPT more frequently (in 11/13 and 5/7 cases for A^+ and A^-, respectively, compared with 7/20 cases for B). This

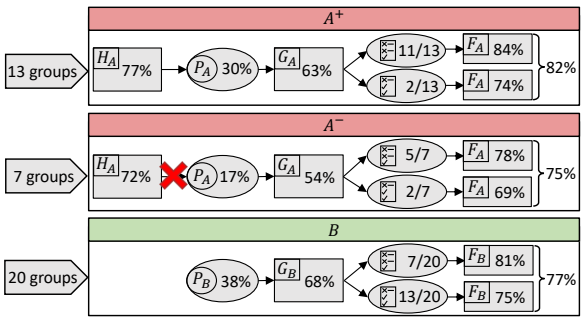


Figure 4: Average quality of artifacts in A^+, A^-, and B.

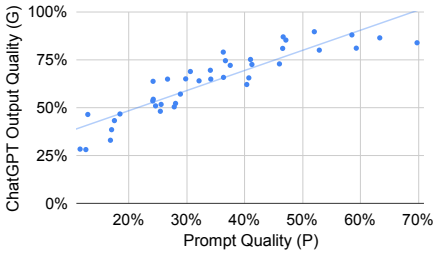


Figure 5: Prompt vs. generated output quality.

is likely because these students were more inclined to refine the results based on their preliminary discussions. This observation is also aligned with impressions reported by students, e.g., «*B* process just uses answers from ChatGPT, whereas the process for *A* uses the best answers from ChatGPT and our own answers».

Students' Prompts. It is widely established that the quality of results produced by ChatGPT highly correlates with the quality of the prompt [40, 49]. Figure 5 confirms this observation for our study as well, showing a strong correlation between the quality of requirements produced by ChatGPT in both processes (y-axis, denoted by G and combining both G_A and G_B) and the quality of the corresponding prompt (x-axis, denoted by P and combining P_A and P_B). The figure also shows that our prompt quality metric is indeed able to reliably distinguish between prompts that produce low- vs. high-quality requirements specifications.

Figure 6 shows average scores for each prompt attribute in each of the three processes. Overall, students did not provide an adequate description of the course setup, examples,and personas in their prompts. They also lacked a sufficient description of the project setup and expected output format.

To our surprise, we observed that students who followed process *B* scored better on almost all prompt quality attributes, likely because students in A^+ mistakenly assumed that their initial submission already provides all this relevant contextual information. The low quality of some initial specifications further complicated matters, as these specifications did not contain clear descriptions and examples of what needed to be done.

The prompt quality is even lower in A^- groups, where no initial submissions were provided. We conjecture that this is because students had already spent time discussing their requirements in the lab and, when deciding to use ChatGPT to augment their work, they implicitly assume the tool also "knows" all the necessary details.

Reliance on ChatGPT. The ability to critically analyze and improve ChatGPT outputs is another factor affecting students' grades. In our study, students who edited ChatGPT results more heavily

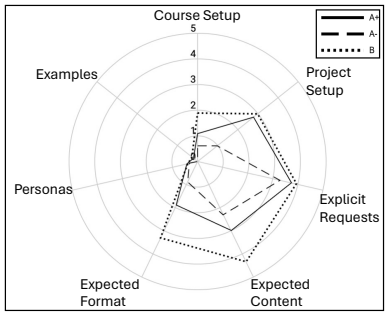


Figure 6: Prompt attribute scores for A^+, A^-, and *B*.

achieved higher grades than those who did not, across all processes. In fact, the five lowest requirements specification grades across all processes all belong to low-editing groups. As an example, when asked to generate use cases for a project, ChatGPT often listed security, performance, and responsiveness as parts of project functionality. However, in the scope of the projects students developed in the class and according to the course specification, most of these would be considered non-functional requirements. Students who correctly fixed ChatGPT suggestions received a higher grade.

Aligning the feasibility of requirements with the scope of the course is another major modification applied by the students. Students from one of the groups noted that «[they] have a way better understanding than ChatGPT in terms of the scope of the project». ChatGPT also often provided generic requirements not directly relevant to the scope of the project, such as "In-App Purchases and Monetization". Overall, we observed that ChatGPT often produced incomplete specifications, generic and unfeasible requirements, and incorrectly classified FRs/NFRs. Interestingly, we also observed that some of the Process *A* students were "swayed" by ChatGPT's projected confidence, even when their original ideas were, in fact, more appropriate for the scope of the course than the refinements proposed by ChatGPT.

Answer to RQ1: Both Process *A* and *B* can result in requirements specifications of comparable levels of quality of around 80%. To achieve better results, students can improve their prompting strategies by providing a more adequate description of the course and project setup, examples, and expected output format for their requests. It is also important to provide students with evidence for the need to analyze ChatGPT results more critically, as students can often be "swayed" by ChatGPT's projected confidence, even when their original results are more accurate.

4.2 RQ2: Effort

To evaluate the quality of requirements one can generate by prompting only and the effort required to craft high-quality prompts, we inspect the data collected in our expert-investigator experiment for subjects S_1-S_4 and compare it to that of students.

Artifacts Produced by Prompting. The Artifacts part of Table 1 (columns 2-5) shows the quality of the artifacts for the four subjects in our experiment. The Students sub-part shows the quality of the initial requirements specifications produced by the students in the lab (H_A, available for S_1 and S_2 only), the quality of their prompts (P), and the quality of the requirements specifications produced by ChatGPT for the prompts (G). The investigator sub-part, *Inv.*, shows the quality of the requirements specifications produced by ChatGPT in the investigator's experiment. We omit other columns for the investigator as (a) the investigator used the exact same initial requirements specifications (H_A) as students did, and (b) the investigator's prompts were explicitly designed to optimize for our metric, thus, the quality of their prompts is 100% by definition.

Provided with such prompts, ChatGPT was able to produce requirements specifications graded at 96% on average (min: 94%, max: 99%). The reasons for not reaching 100% include our deliberate decision not to follow up with clarification, as well as ChatGPT-related issues, such as generating incomplete specifications, generic and unfeasible requirements, and incorrectly classified FRs/NFRs. While

Table 1: Prompt Assessment

Subj.	Artifacts				Effort																			
	Students			Inv.	#Questions		#Words																	
							Total		Course Stp.		Project Stp.		Explicit Req.		Exp. Content		Exp. Format		Personas		Examples		Other	
	H_A	P	G	G	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.	Stu.	Inv.
S_1	92%	27%	65%	99%	9	29	188	6,206	24	566	118	3,655	36	328	10	572	0	52	0	40	0	982	0	11
S_2	27%	17%	38%	97%	16	29	527	2,915	0	566	208	364	269	328	15	572	0	52	0	40	0	982	35	11
S_3	-	70%	84%	94%	23	23	617	3,045	273	566	79	291	50	343	80	610	22	48	68	40	36	1,137	9	10
S_4	-	18%	47%	95%	4	23	56	2,906	0	566	32	152	24	343	0	610	0	48	0	40	0	1,137	0	10
Avg	-	33%	58%	96%	13	26	347	3,768	74	566	109	1,115	95	336	26	591	6	50	17	40	9	1,060	11	11

similar issues also occurred in outputs ChatGPT produced for students, their magnitude is mitigated by the good prompting strategy. For S_1, the high-quality specifications H_A (92%) provided as an input allowed ChatGPT to inherit these specific and well-scoped requirements and further help produce nearly-perfect output (99%).

The improved quality of the prompts comes at the expense of effort invested in crafting such prompts, which we discuss next.

Time. For our investigator, interactions with ChatGPT took five hours per subject, out of which four hours were spent on crafting and refining the prompt (common for all subjects) and one additional hour was spent on the actual interactions with ChatGPT. Interestingly, students who achieved the highest overall requirements specifications scores also reported investing around five hours in their projects, on average. However, this investment was spent on a combination of various activities, including crafting ChatGPT prompts, brainstorming, and inspecting and manually adjusting ChatGPT results.

For the projects in the medium grade range of around 80%, students invested 1.5-3 hours, on average. Interestingly, requirements specifications generated in the 1.5 hours lab session, without any use of ChatGPT (H_A), were, on average, of comparable quality to the mid-range final artifacts produced by the students in the course and were even better than artifacts produced by turning to ChatGPT directly in Process B (G_B).

When focusing on students' prompting effort in particular, the students with the highest-quality prompts reported that crafting prompts took them about three hours, on average. In comparison, for the remaining groups, this time was one hour, on average. Yet, students' highest-quality prompts still resulted in requirements specifications graded around 80%, e.g., the grade for the ChatGPT-produced requirements specifications for subject S_3 (the highest prompt quality and, as a result, the highest-quality requirements specification from all students' prompts) is still 84%, due to the remaining prompt issues.

Question and Word Count. The Effort part of Table 1 shows the number of questions (#Ques) and the number of words (#Words) in a chat conversation, for students (Stu.) and the investigator (Inv.), separately. We further report the word count for each prompt quality attribute: Course Setup, Project Setup, etc.

These metrics, again, assess the needed investment in producing high-quality prompts. For example, the students' prompt for subject S_3 indeed had the highest number of questions (23) and words (617) among other students' prompts. While our investigator asked the same number of questions in this case, the questions were around 5 times longer, allowing them to achieve 94% vs. 84% in the quality of the generated requirements specification (in return for the effort of producing text with 3,045 vs. 617 words). Across all case studies, the investigator conversations were 10 times longer, on average.

Interestingly, for S_2, the relatively high number of questions (16) and words (527) in the students' artifacts did not translate to a high-quality outcome (38% for G). Our analysis shows that students in S_2 omitted many of the prompt quality components, such as course setup, expected output format, and examples, all of which were included in students' S_3 prompt.

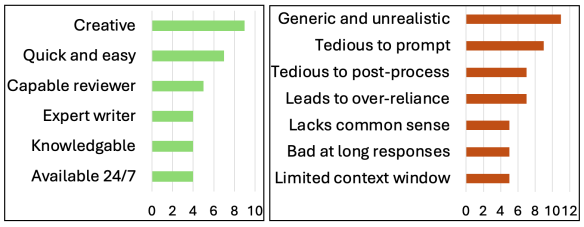
The effort required for closing the quality gap while relying on ChatGPT is also aligned with impressions reported by the students themselves: «getting to 80% is easy, but a lot of work needs to be done to get the remaining 20%».

Answer to RQ2: While a mid-range quality level outcome can be achieved with moderate effort, reaching high-quality results often requires more than twice the investment, with or without ChatGPT. This investment is spent in brainstorming, crafting high-quality prompts, and critically assessing and refining ChatGPT's outputs.

4.3 RQ3: Usage Patterns

Students' Impressions. Figure 7 summarizes the most prominent observations reported by the students in their own reflections on using the tool. In a nutshell, a number of groups mention that ChatGPT is creative and can be a valuable tool for brainstorming, essentially treating it as another member of the team. Some also report that it is quick and easy to use, can act as an expert reviewer in catching mistakes, is knowledgeable, and is available 24/7, allowing the students to treat it as a member of the teaching staff. At the same time, ChatGPT responses being generic and unrealistic was the most frequently-mentioned issue, which is consistent with observations in our own study, where ChatGPT produced some generic and unfeasible results even when using high-quality prompts. The significant amount of effort required to craft prompts and refine the results, the lack of "common sense," and the tendency to over-rely on ChatGPT results which, in turn, fosters a poor understanding of materials, were other main mentioned disadvantages, all of which are consistent with our observations as well.

Further inspecting students' reflections, we observed that some groups valued the collaborative teamwork setup and found it to be more productive and enjoyable: «Time spent in the lab collaborating on the requirements specifications was undeniably more enjoyable and quite constructive». Others believed ChatGPT reduced their workload and made the requirements engineering process more efficient: «B was more productive because ChatGPT takes away that initial brainstorming time and gives you what you need in



(a) Advantages. **(b) Disadvantages.**

Figure 7: Student reflections on using ChatGPT.

seconds». We believe each of these different working modes is effective and suitable for students with different personality traits: some prefer an interactive environment while others work better with a digital assistant. Further investigating such correlations is beyond the scope of this work.

As already mentioned in Section 4.1, we observed that some groups overly relied on ChatGPT: despite the high quality of the in-lab requirements specification (92% and 86%), these groups did poorly in prompting and further adopted those mistakes made by ChatGPT that they did not originally have, such as including very generic use cases, non-measurable non-functional requirements, and inconsistent descriptions. This led to a low score in their final specification (65% and 67%, respectively). We believe that the confident tone adopted by ChatGPT could sway the opinion of inexperienced users in the wrong direction. Indeed, some of the groups confirm this observation: "ChatGPT is indeed a double-edged sword, the person needs to be well-versed about the topic".

Usage Modes. To gain further insights into how students utilized ChatGPT, two authors independently read the students' prompts, as discussed in Section 3.3. We identified four main ChatGPT usage modes among students, which we define below. Each of the usage modes is applied to one or several requirements specification components: use cases, actors, use case diagram, success/failure scenarios, and non-functional requirements.

1. **Generation** refers to cases when students asked ChatGPT to generate new components, e.g., "can you generate failure scenarios for this functional requirement?".
2. **Concretization** refers to cases when students asked to refine an existing component with more detailed content or information, e.g., "Can you provide numerical values for these non-functional requirements based on average values from high-performing apps?".
3. **Rewriting** refers to cases when students asked to adjust formatting, improve the clarity of existing text, or fix typos and grammar mistakes, e.g., "can you help me edit the grammar of my work?".
4. **Concept Understanding** refers to cases when students' questions focused on gaining knowledge about some requirements specification component, utilizing ChatGPT as a replacement of an information source, such as the course teaching staff or internet, e.g., "what is a functional requirement?".

In what follows, we analyze how students utilize ChatGPT outputs obtained in various usage modes, focusing on Generation, Concretization, and Rewriting, as these are more directly related to the artifacts produced and evaluated in this work. Specifically, to estimate students' satisfaction with ChatGPT suggestions for each mode, we consider all 40 projects in processes A^+, A^-, and B, measuring the percentage of projects where students accepted at least one output for a particular usage mode out of all projects where students utilized ChatGPT in this mode.

Table 2: Usage Patterns and Acceptance Rate

Process	Total	Generation		Concretization		Rewriting	
	#Projects	#Projects	Accept. Rate	#Projects	Accept. Rate	#Projects	Accept. Rate
A^+	13	11	27% (3 of 11)	10	80% (8 of 10)	9	66% (6 of 9)
A^-	7	7	29% (2 of 7)	4	50% (2 of 4)	0	-
B	20	20	60% (12 of 20)	7	100% (7 of 7)	0	-
All	40	38	45% (17 of 38)	21	81% (17 of 21)	9	66% (6 of 9)

Table 2 shows the outcome of this analysis, for each process individually and for all processes combined. As expected, all groups following A^- and B processes, as well as the majority of groups in A^+ process, used ChatGPT in Generation mode to either create from scratch or extend their requirements specification. However, the acceptance rate was much lower in both Process A^+ and A^- groups, compared with that in Process B, as the former groups already had a good preliminary idea of what the requirements specifications should look like.

Instead, many Process A^+ groups successfully utilize ChatGPT to concretize their existing ideas (the highest acceptance rate overall) as well as provide writing improvements (albeit with a lower acceptance rate). As for the groups following Process A^- and B, they successfully used ChatGPT to concretize preliminary partial information about their project, as well as intermediate ChatGPT replies, e.g., «For an Android app that lets users query a grades website for a university, describe a functional requirement for authenticating the user». Surprisingly, none of the groups in A^- and B processes asked ChatGPT for rewriting, presumably because the text they used was generated with ChatGPT to start with. Groups also often combined patterns, such as requesting new functional requirements while concretizing existing ones.

Overall, the concretization usage mode led to the highest acceptance rates across all processes, demonstrating ChatGPT's ability to improve on the well-defined user input: «ChatGPT outputs are more desirable when you ask it for improvement».

Answer to RQ3: Students perceive ChatGPT as an easy-to-approach, available, and useful tool. Idea concretization and text rewriting are the most popular ChatGPT usage modes, especially for students with well-formed initial ideas. At the same time, students are concerned with generic and unrealistic suggestions, prompting difficulties, and the potential of over-reliance on the tool.

5 Threats to Validity

External validity pertains to conditions that limit the generalization of our findings. As in many other exploratory studies, our research is inductive in nature and thus might not generalize beyond the subjects that we studied. As our study involved a large-scale course with 20 student groups, we believe our results are reliable. However, to further mitigate this threat, we make our methodology and evaluation strategy publicly available to encourage replication studies across different settings, to further assess the generalizability of the findings.

Team dynamics pose another threat to validity. While the teams used ChatGPT during in-class lab sessions and off-class meetings,

we did not systematically control or collect data on team interactions. Team compositions, collaboration style, or interpersonal dynamics may have influenced students' experiences and satisfaction when using AI techniques. However, we believe our main findings hold across different team configurations and individuals.

The main threat to the **internal validity** of our results stems from the manual artifact inspection and grading. We mitigate this threat by having at least two authors perform all data analysis steps independently and further resolving all disagreements in a discussion with another author.

Our selection of the investigator for the case study could also affect our results. We mitigate this threat by carefully selecting a competent subject with the necessary background and skills matching our target audience. Finally, to reduce the effect of ChatGPT responses varying when repeated with the same prompt, we ran each case study three times and presented the averaged results.

6 Discussion and Future Directions

Training Students to Better Utilize AI Tools. As the students in this course offering did not receive any training in crafting high-quality prompts, our results probably represent an unbiased sample of typical ChatGPT usage. Sadly, our prompt quality assessment results show rather low prompting proficiency of the students, indicating educational opportunities in this domain. As one student put it in their feedback: «prompting is hard. it needs thoughtful thinking». Our results also show the importance of instilling critical thinking when analyzing AI outputs; students expressed the same sentiment: «[need to] teach students how to critically analyze the quality of AI answers». In this reality, it is important to systematically train students on how to interact with AI tools, both when prompting and when critically analyzing their outputs.

For prompting, we believe that the prompt quality metric we proposed and the prompting strategy our expert user followed could be a useful starting point of such training. Our prompt metric and strategy basically emphasize the importance of elaborating on each task at a sufficient level of detail and systematically breaking down larger tasks into smaller ones – concepts that are not new in the field of Software Engineering but that now receive renewed attention in the context of prompt-based use of AI.

For critical analysis, as students learn from failures probably as much as they learn from following successful practices, we believe that sharing with students concrete examples of conversational AI deficiency in the context of their assignments could be useful and we intend to leverage information and evidence collected in this work for future offerings of the course.

Furthermore, the course offering discussed in this paper was deliberately designed to promote critical thinking by allowing students to critically reflect on the process of using ChatGPT in two different modes (process *A* and *B*), explicate the strengths and weaknesses of both approaches, and better understand which strategies work best for them. While some student reflections led us to believe our design was effective, our teaching approach induced extra overhead on the course staff, e.g., because they had to grade two requirements specification artifacts instead of one.

For student cohorts larger than ours, such an approach might not scale well. One way to mitigate this issue while keeping the benefits of focusing students' attention on the critical analysis of their interaction with AI technologies, could be using selective and peer-review evaluations. That is, we could randomly select and grade only one of the requirements specifications produced by one of the processes, *A* or *B*. Asking teams to peer-review the requirements specifications is another option, which also has additional benefits of exposing students to several requirements specifications examples and training students in reviewing others' work. While these approaches may lead to less accurate grading, we believe that the benefits could outweigh the scaling drawbacks.

As industry shifts to the collaborative human-AI nature of work, i.e., Software Engineering professionals are nowadays augmented with AI tools, learning also shifts to the collaborative student-AI experience, i.e., students' course deliverables are now produced collaboratively with AI tools. Educators thus need to rethink the learning objectives they target and develop new assessment methods to estimate students' gained knowledge w.r.t. these objectives. In this course offering, we assessed whether the students gained knowledge in requirements engineering through an in-class paper-only written quiz. Analyzing the relationship between students' grades for the quiz and their reported reliance on ChatGPT, we found no correlation between the two. We thus believe students received adequate training in requirements engineering irrespective of their ChatGPT usage. However, we believe that a broader effort aiming to understand how to set up and assess a more general AI-aware and AI-targeted training strategy and how to embed it into the program curriculum, to best prepare professionals for the new AI-empowered world, is needed. We hope our experience and lessons learned from this offering of the course can be used as inspiration for such future educational initiatives.

AI Technology. Our analysis of students' ChatGPT conversations and our own prompting study shows some deficiencies of the underlying AI technology, such as being incomplete, unspecific, or infeasible; it has factual knowledge gaps and can produce outputs that are internally inconsistent. Moreover, it produces non-deterministic replies, with a varying degree of correctness.

Fine-tuning AI models for a task-specific setup, such as configuring ChatGPT for requirements engineering processes in a particular course, project, team, organization, etc., could save some of the prompting and refinement effort. For example, we observed that when students' prompts did not provide any clarifications about the intended use and scope of the project, ChatGPT suggested requirements that did not align with their expectations and needs, e.g., "the system should support over 10,000 users".

Our analysis of students' prompts also shows that students often ask ambiguous questions, such as "Improve this", which can be interpreted as "improve wording", "fix the structure of the text", "add more requirements", etc. In most instances where students provided such ambiguous prompts, ChatGPT made specific assumptions about the students' goals and proposed a set of improvements, which were not necessarily aligned with students' intentions. As a reasonable human actor would likely ask for clarifications on what kind of improvements are expected, we believe improving the ability of AI tools to figure out when to ask clarifying questions and which questions to ask would largely improve the models.

Finally, a number of students also flagged the lack of support for interacting with ChatGPT as a team. As teamwork is a crucial

element of many engineering tasks, an interface supporting the simultaneous interaction of multiple team members with ChatGPT would increase its usability in collaborative environments. Designing such interfaces seems valuable future work.

As AI technology continues to improve, our experience with ChatGPT 3.5 might not generalize to future AI tools. However, we believe that our main findings and proposed educational adaptations pertain to the use of AI technology in general, not ChatGPT 3.5 in particular, and will remain relevant.

7 Related Work

We discuss related work in the following three dimensions:

Pedagogical Approaches. A number of authors emphasize the need to adjust teaching and learning practices [22, 23, 37, 43] to the new era of conversational AI technologies. For example, Denny et al. [22] suggest teaching *prompt problems*, i.e., prompts that guide an AI tool to generate the code required to solve a given problem. Others explore the use of AI in generating teaching materials, including programming exercises [10, 11, 35, 55, 59], contextualized problem statements [55], and personalized questions tailored to students' interests [44]. Yet another line of research explores techniques for automating assessment generation, grading, and feedback generation processes [19, 24, 33, 34, 39, 65]. Our work is orthogonal to these directions.

A few authors conduct controlled experiments to evaluate the added value of GenAI in an educational context, where only the experimental group vs. the control group is granted access to GenAI tools. Specifically, Choudhuri et al. [18] study the effectiveness of ChatGPT in assisting students in SE tasks, such as creating a git branch or identifying/removing code smells. Similar to us, the authors found no statistical differences in participants' productivity or self-efficacy when using ChatGPT as compared to traditional resources. Garg et al. [28] report that, while experimental groups performed significantly better for data analysis, the performance is not as drastically different for end-to-end development tasks. Our work follows this general direction, encouraging students to apply critical thinking when working with AI techniques.

Yet another line of research emphasizes the limitations of using generative AI in education. These include the potential for students to become over-reliant on generative AI tools to solve problems [8, 10, 25, 26, 32] or to spend their time in unproductive ways, e.g., it may take longer to figure out an effective prompt than to write the code [2, 50, 63]. Our results confirm these findings and discuss in more detail both the anticipated effort for successful prompt engineering and cases where over-reliance on ChatGPT caused students to lose marks in the requirements engineering tasks.

Assessments of AI Capabilities. Several studies focus on assessing the capabilities of conversational AI techniques for solving typical course assignments and other similar educational activities. While the majority of this work looks at code-related tasks [21, 27, 31, 51], few also explore the applicability of AI in tasks that extend beyond coding, such as requirements engineering [9, 14, 17, 30, 46, 61]. These works mostly rely on interviews, surveys, and students' self-reflections. They show that students who were asked to use ChatGPT for the requirements engineering task generally have a positive perception of its capabilities [14, 30] and believe that ChatGPT enhanced their efficiency, accuracy, and understanding [61]. However, they rarely use it for this task if not explicitly asked [31]. Outside of educational contexts, studies on the applicability of GenAI in requirements engineering perform expert-guided evaluation of GenAI tools [12, 42, 53], e.g., using different prompting techniques. These studies report that the tools are able to achieve reasonably good performance overall and have the potential to assist in the requirement engineering process. Our work contributes to this direction. Yet, instead of relying on interviews, surveys, and experts, we perform a detailed analysis of student- and ChatGPT-generated artifacts, which allows us to gain deeper insights into the workflows students followed, successful and failing practices, and the effort required to employ ChatGPT for requirements engineering tasks.

Prompt Engineering. Prompt engineering is the systematic design and optimization of inputs (i.e., prompts) for conversational AI systems, to enhance the quality of their outputs [15]. Prompting strategies range from basic ones, as documented in the OpenAI prompting guide [47], to advanced ones, such as, chain-of-thought prompting [62] and program-of-thoughts prompting [16]. A variety of prompting techniques are classified by recent studies and surveys, e.g., [45, 54, 56]. Given the complexity and the specialized knowledge or tooling the advanced prompting techniques necessitate, we have chosen to exclude these advanced techniques from the scope of our current study, focusing instead on more broadly applicable ones.

Similar to us, Shah et al. [57] analyzed coding-related prompts and identified that students use one-shot prompting (i.e., asking for the entire implementation at once) and spend a significant amount of time debugging the generated output. A recent work concurrent with ours [41] analyzed 150 prompting-related papers and blogs and proposed a framework for evaluating prompt quality through 21 properties such as *objectives*, *examples*, *contextual consistency*, *token quantity*, *politeness*, and *complexity minimization*. Our prompt attributes align well with these findings.

8 Conclusion

In this paper, we reported on our experience integrating ChatGPT into the curriculum of a large upper-level undergraduate project-based course on Software Engineering, where students used ChatGPT in a controlled manner to help with the requirements specification task. We discuss the course setup we devised to encourage students to critically assess AI outputs, the effort required to produce high-quality ChatGPT outputs, and students' approaches to utilize ChatGPT for their work. Our study shows that there are different strategies for achieving high-quality results in interactions with GenAI technologies but reaching beyond 80% quality in any of the strategies requires a substantial effort. This effort can come in crafting high-quality prompts, thorough thinking before discussing with ChatGPT, and improving ChatGPT outcomes. We hope that our work will be informative for students and educators, and will provide ideas for AI technologists on how to improve and better integrate AI into existing workflows.

Acknowledgments

Part of this work was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] [n. d.]. ChatGPT OpenAI. https://chat.openai.com/.

[2] [n. d.]. Generative AI Assistants in Software Development Education: A Vision for Integrating Generative AI into Educational Practice, not Instinctively Defending Against it. ([n. d.]).

[3] [n. d.]. GitHub Copilot: Your AI Pair Programmer. https://copilot.github.com/.

[4] [n. d.]. Piazza. https://piazza.com/.

[5] [n. d.]. Supplementary Materials. https://resess.github.io/artifacts/GenAI_in_RE/.

[6] 2018. ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering. *ISO/IEC/IEEE 29148:2018(E)* (2018), 1–104.

[7] Steve Adolph and Paul Bramble. 2003. *Patterns for effective use cases.* Addison-Wesley.

[8] Matin Amoozadeh, David Daniels, Daye Nam, Aayush Kumar, Stella Chen, Michael Hilton, Sruti Srinivasa Ragavan, and Mohammad Amin Alipour. 2024. Trust in Generative AI Among Students: An Exploratory Study. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 67–73.

[9] Amal Al Badi, Khulood Al Handhali, Sumaiya Alsalmi, Ozear Al-Zadjali, Shihab AlYarobi, and Abdullah Alshibli. 2024. Exploring the Impact of ChatGPT in Engineering Education: A Mixed Methods Study at the Military Technological College in Muscat. In *International Conference on Digital Technology in Education (ICDTE)*. 247–254.

[10] Brett A Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming is Hard-or at least it Used to be: Educational Opportunities and Challenges of AI Code Generation. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 500–506.

[11] Jonnathan Berrezueta-Guzman and Stephan Krusche. 2023. Recommendations to create programming exercises to overcome ChatGPT. In *International Conference on Software Engineering Education and Training (CSEE&T)*. 147–151.

[12] Manal Binkhonain and Reem Alfayez. 2025. Are Prompts All You Need? Evaluating Prompt-Based Large Language Models (LLMs) for Software Requirements Classification. *Requirements Engineering* (2025), 1–21.

[13] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 33. 1877–1901.

[14] Juan Pablo Carvallo and Lenin Erazo-Garzón. 2023. On the Use of ChatGPT to Support Requirements Engineering Teaching and Learning Process. In *Latin American Conference on Learning Technologies*. 328–342.

[15] Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2025. Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review. *Patterns* 8, 8 (2025).

[16] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023. Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks. *Transactions on Machine Learning Research (TMLR)* (2023).

[17] Haowei Cheng, Jati H. Husen, Yijun Lu, Teeradaj Racharak, Nobukazu Yoshioka, Naoyasu Ubayashi, and Hironori Washizaki. 2025. Generative AI for Requirements Engineering: A Systematic Literature Review. *Software: Practice and Experience* (2025), 1–12.

[18] Rudrajit Choudhuri, Dylan Liu, Igor Steinmacher, Marco Gerosa, and Anita Sarma. 2024. How Far Are We? The Triumphs and Trials of Generative AI in Learning Software Engineering. In *International Conference on Software Engineering (ICSE)*. 1–13.

[19] Bruno Pereira Cipriano, Pedro Alves, and Paul Denny. 2024. A Picture Is Worth a Thousand Words: Exploring Diagram and Video-Based OOP Exercises to Counter LLM Over-Reliance. arXiv:2403.08396 [cs.SE]

[20] Marian Daun, Alicia M Grubb, Viktoria Stenkova, and Bastian Tenbergen. 2023. The Field of Requirements Engineering Education. In *International Conference on Software Engineering Education and Training (CSEE&T)*. 119–119.

[21] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 1136–1142.

[22] Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A Becker, and Brent N Reeves. 2024. Prompt Problems: A New Programming Exercise for the Generative AI Era. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 296–302.

[23] Paul Denny, James Prather, Brett A Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* 67, 2 (2024), 56–67.

[24] Felix Dobslaw and Peter Bergh. 2023. Experiences with Remote Examination Formats in Light of GPT-4. In *European Conference on Software Engineering Education (ECSEE)*. 220–225.

[25] Amanda S Fernandez and Kimberly A Cornell. 2024. CS1 with a Side of AI: Teaching Software Verification for Secure Code in the Era of Generative AI. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 345–351.

[26] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Australasian Computing Education Conference (ACE)*. 10–19.

[27] James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A Becker. 2023. My AI Wants to Know If This Will Be on the Exam: Testing Open AI Codex on CS2 Programming Exercises. In *Australasian Computing Education Conference (ACE)*. 97–104.

[28] Ashish Garg, K. Nisumba Soodhani, and Ramkumar Rajendran. 2025. Enhancing Data Analysis and Programming Skills through Structured Prompt Training: The Impact of Generative AI in Engineering Education. *Computers and Education: Artificial Intelligence* (2025), 100380.

[29] Andrew C. Gemino and Drew Parker. 2009. Use Case Diagrams in Support of Use Case Modeling: Deriving Understanding from the Picture. *Journal of Database Management (JDM)* (2009).

[30] Sharon Guardado, Risha Parveen, Zheying Zhang, Maruf Rayhan, and Nirnaya Tripathi. 2025. Students' Perceptions of the Use of LLMs in Requirements Engineering Education: A Cross-University Empirical Study. In *IEEE International Requirements Engineering Conference (RE)*. 130–141.

[31] Khadija Hanifi, Orcun Cetin, and Cemal Yilmaz. 2023. On ChatGPT: Perspectives from Software Engineering Students. In *IEEE International Conference on Software Quality, Reliability, and Security (QRS)*. 196–205.

[32] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *International Computing Education Research (SIGCSE TS)*. 93–105.

[33] Muntasir Hoq, Yang Shi, Juho Leinonen, Damilola Babalola, Collin Lynch, Thomas Price, and Bita Akram. 2024. Detecting ChatGPT-generated Code Submissions in a CS1 Course Using Machine Learning Models. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 526–532.

[34] Sajed Jalil, Suzzana Rafi, Thomas D LaToza, Kevin Moran, and Wing Lam. 2023. ChatGPT and Software Testing Education: Promises & Perils. In *IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 4130–4137.

[35] Mollie Jordan, Kevin Ly, and Adalbert Gerald Soosai Raj. 2024. Need a Programming Exercise Generated in Your Native Language? ChatGPT's Got Your Back: Automatic Generation of Non-English Programming Exercises Using OpenAI GPT-3.5. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 618–624.

[36] Ishika Joshi, Ritvik Budhiraja, Harshal Dev, Jahnvi Kadia, Mohammad Osama Ataullah, Sayan Mitra, Harshal D Akolekar, and Dhruv Kumar. 2024. ChatGPT in the Classroom: An Analysis of Its Strengths and Weaknesses for Solving Undergraduate Computer Science Questions. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 625–631.

[37] Vassilka D Kirova, Cyril S Ku, Joseph R Laracy, and Thomas J Marlowe. 2024. Software Engineering Education Must Adapt and Evolve for an LLM Environment. In *ACM Technical Symposium on Computer Science Education*, Vol. 1. 666–672.

[38] Aobo Kong, Shiwan Zhao, Hao Chen, Qicheng Li, Yong Qin, Ruiqi Sun, and Xin Zhou. 2023. Better Zero-Shot Reasoning with Role-Play Prompting. arXiv:2308.07702 [cs.CL]

[39] Sam Lau and Philip Guo. 2023. From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot. In *ACM Conference on International Computing Education Research (ICER)*, Vol. 1. 106–121.

[40] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 563–569.

[41] Do Xuan Long, Duy Dinh, Ngoc-Hai Nguyen, Kenji Kawaguchi, Nancy F. Chen, Shafiq Joty, and Min-Yen Kan. 2025. What Makes a Good Natural Language Prompt?. In *Proc. of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*. 5835–5873.

[42] Sebastian Lubos, Alexander Felfernig, Thi Ngoc Trang Tran, Damian Garber, Merfat El Mansi, Seda Polat Erdeniz, and Viet-Man Le. 2024. Leveraging LLMs for the Quality Assurance of Software Requirements. In *IEEE International Requirements Engineering Conference (RE)*. 389–397.

[43] Stephen MacNeil, Joanne Kim, Juho Leinonen, Paul Denny, Seth Bernstein, Brett A Becker, Michel Wermelinger, Arto Hellas, Andrew Tran, Sami Sarsa, et al. 2023. The Implications of Large Language Models for CS Teachers and Students. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 1255.

[44] Stephen MacNeil, Andrew Tran, Juho Leinonen, Paul Denny, Joanne Kim, Arto Hellas, Seth Bernstein, and Sami Sarsa. 2022. Automatically Generating CS Learning Materials with Large Language Models. *arXiv* (2022).

[45] Yuetian Mao, Junjie He, and Chunyang Chen. 2025. From Prompts to Templates: A Systematic Prompt Template Analysis for Real-world LLM Apps. In *ACM International Conference on the Foundations of Software Engineering (FSE Companion)*. 75–86.

[46] Sabina-Cristiana Necula, Florin Dumitriu, and Valerică Greavu-Șerban. 2024. A Systematic Literature Review on Using Natural Language Processing in Software Requirements Engineering. *Electronics* (2024), 2055.

[47] OpenAI. [n. d.]. Prompt engineering. https://platform.openai.com/docs/guides/prompt-engineering.

[48] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. arXiv:2308.02828 [cs.SE]

[49] Hammond Pearce, Benjamin Tan, Baleegh Ahmad, Ramesh Karri, and Brendan Dolan-Gavitt. 2023. Examining Zero-shot Vulnerability Repair with Large Language Models. In *Symposium on Security and Privacy (S&P)*. 2339–2356.

[50] James Prather, Brent N Reeves, Paul Denny, Brett A Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That it Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction (CHI)* 31, 1 (2023), 1–31.

[51] Ben Puryear and Gina Sprint. 2022. Github Copilot in the Classroom: Learning to Code with AI Assistance. *Journal of Computing Sciences in Colleges* 38, 1 (2022), 37–47.

[52] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[53] Krishna Ronanki, Beatriz Cabrero-Daniel, Jennifer Horkoff, and Christian Berger. 2024. Requirements engineering using generative ai: Prompts and prompting patterns. In *Generative AI for effective software development*. 109–127.

[54] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv:2402.07927 [cs.AI]

[55] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *ACM Conference on International Computing Education Research (ICER)*, Vol. 1. 27–43.

[56] Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, HyoJung Han, Sevien Schulhoff, Pranav Sandeep Dulepet, Saurav Vidyadhara, Dayeon Ki, Sweta Agrawal, Chau Pham, Gerson Kroiz, Feileen Li, Hudson Tao, Ashay Srivastava, Hevander Da Costa, Saloni Gupta, Megan L. Rogers, Inna Goncearenco, Giuseppe Sarli, Igor Galynker, Denis Peskoff, Marine Carpuat, Jules White, Shyamal Anadkat, Alexander Hoyle, and Philip Resnik. 2024. The Prompt Report: A Systematic Survey of Prompt Engineering Techniques. arXiv:2406.06608 [cs.CL]

[57] Anshul Shah, Anya Chernova, Elena Tomson, Leo Porter, William G. Griswold, and Adalbert Gerald Soosai Raj. 2025. Students' Use of GitHub Copilot for Working with Large Code Bases. In *Proc. of the 56th ACM Technical Symposium on Computer Science Education (SIGCSE)*. 1050–1056.

[58] Ian Sommerville and Pete Sawyer. 1997. *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc.

[59] Sandro Speth, Niklas Meißner, and Steffen Becker. 2023. Investigating the Use of AI-Generated Exercises for Beginner and Intermediate Programming Courses: A ChatGPT Case Study. In *IEEE International Conference on Software Engineering Education and Training (CSEE&T)*. 142–146.

[60] Anselm Strauss and Juliet Corbin. 1998. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Thousand Oaks, CA: Sage.

[61] Muhammad Waseem, Teerath Das, Aakash Ahmad, Peng Liang, Mahdi Fahmideh, and Tommi Mikkonen. 2024. ChatGPT as a Software Development Bot: A Project-based Study. In *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*.

[62] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 35. 24824–24837.

[63] Michel Wermelinger. 2023. Using Github Copilot to Solve Simple Programming Problems. In *Technical Symposium on Computer Science Education (SIGCSE TS)*. 172–178.

[64] Ning Wu, Ming Gong, Linjun Shou, Shining Liang, and Daxin Jiang. 2023. Large Language Models are Diverse Role-Players for Summarization Evaluation. In *Natural Language Processing and Chinese Computing Conference (NLPCC)*. 695–707.

[65] C. Zastudil, M. Rogalska, C. Kapp, J. Vaughn, and S. MacNeil. 2023. Generative AI in Computing Education: Perspectives of Students and Instructors. In *IEEE Frontiers in Education Conference (FIE)*. 1–9.

[66] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H Chi. 2022. Least-to-Most Prompting Enables Complex Reasoning in Large Language Models. In *International Conference on Learning Representations (ICLR)*.

[67] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large Language Models are Human-Level Prompt Engineers. In *International Conference on Learning Representations (ICLR)*.