Documenting User Stories: Does LLM Help?

Pedro Garcia[1], João Vitor Depollo[1], Laura Barroso[1], Eduardo Figueiredo[1],
Kattiana Constantino[2], Filipe Fernandes[1], Filipe Roseiro Côgo[3]
[1]Federal University of Minas Gerais (UFMG), Belo Horizonte, Brazil
[2]Federal University of Vale Jequitinhonha e Mucuri (UFVJM), Diamantina, Brazil
[3]Huawei Technologies, Kingstong, Canada

ABSTRACT

The increasing capabilities of large language models (LLMs) offer a promising avenue for enhancing human-intensive software engineering activities, such as requirements elicitation. However, we lack strong empirical evidence on the impact of LLMs, such as ChatGPT, not only on the productivity and quality of requirement artifacts, such as user stories, but also on the learning of requirements elicitation. To address this gap, this exploratory study investigates the impact of ChatGPT used by 28 undergraduate students from two universities on documentation of user stories. Based on the Latin Square design, students created user stories with and without support of the LLM. We then analyzed the effect of ChatGPT in terms of completion time, solution quality, and its reported benefits and challenges. As expected, our results indicate that participants using the LLM reduced the mean time to complete their tasks from 12 to 6 minutes. However, we observed major quality problems with the LLM-supported solutions, such as generic responses. Participants also reported other learning problems with the LLM use, such as little learning effort and a lack of creativity. We conclude that educators and software professionals should be careful when using LLMs since they not only may harm the student learning but also mitigate the thorough understanding of specific software requirements.

CCS CONCEPTS

• **Social and professional topics** → **Computing education**; • **Software and its engineering** → **Software creation and management**.

KEYWORDS

Education, LLM, Requirements, User Story

1 INTRODUCTION

Requirements engineering is a key software engineering activity aiming at discovering, documenting and managing the requirements for a software system [36]. This activity involves intense human interaction through interviews, observations, workshops, and document analysis, making it both time-consuming and susceptible to misinterpretation [17]. The widespread adoption of agile software development methods, such as Scrum [33] and Kanban [1], has led to the rise in the use of user stories in industry [13]. In fact, previous research [3] indicated that 59% of agile software developers used a specific template to write user stories. This template, known as Connextra, defines the following structured format [11]: As a <role>; I want to <action>; so that <benefit>.

Recent advances in Large Language Models (LLMs) have boost interest in their use for software engineering problems, such as code generation [26], software quality [28], and software documentation [22]. In fact, their ability to understand complex contexts allows LLMs to generate flexible and adaptable solutions [28]. In particular, LLMs offer a promising avenue for enhancing human-intensive software engineering activities [24], such as requirements elicitation. For instance, the popularity of user stories among practitioners and their simple yet strict structure make them ideal candidates for automatic generation by LLMs [13]. However, we lack strong empirical evidence on the impact of LLMs, such as ChatGPT, not only on the productivity and quality of requirement artifacts but also on the learning outcomes of requirements elicitation.

Recent interest in LLMs has led to a growing body of research exploring their potential in software engineering (SE), software quality and requirements engineering (RE). For instance, previous work [25, 38] conducted literature reviews focusing on the application of LLMs into requirements engineering contexts and identified key opportunities and limitations. In addition to literature reviews, other studies [32] conducted empirical studies to evaluate the use of LLMs in requirements engineering education. Beyond requirements engineering, some studies have also explored LLMs as a tool to support software engineering education more broadly [14]. For instance, Pereira et al. [29] investigated the role of LLMs within the context of software engineering education, contributing further insights into their educational value. However, as far as we are concerned, no previous work has quantitatively and qualitatively evaluated the benefits and challenges of using LLMs for use stories documentation.

This paper presents an exploratory study aiming to investigate the impact of using LLMs on the learning of software requirements documentation. In this exploratory study, 28 undergraduate students enrolled in software engineering courses documented two use cases using the Connextra template for user stories [11]. We employed the Latin Square design, allowing each participant to perform the tasks with and without the LLM support; the Web interface ChatGPT-4o in our study. We collected and analyzed data related to (i) the time required to complete each task, (ii) the quality of the provided user stories, and (iii) the perceptions of the students about the use of ChatGPT for requirements documentation.

Our results show that students completed their tasks faster with the LLM support. The mean completion time dropped

from 12 minutes (without LLM) to 6 minutes (with LLM). We also observed some interesting results. The manual inspections of requirements quality, performed by two software engineering professors, uncover major quality problems with the LLM-supported use stories. For instance, we observed that students using the LLM not only documented requirements in a more general way but also provided very similar responses. Our qualitative analysis of students feedback also highlights the most common learning challenges of using LLM for software requirements elicitation. Based on this analysis, we concluded that participants using LLMs have reduced learning gains and lack creativity in their solutions.

Our results have several practical implications for future agile software developers and educators. On the one hand, the use of LLMs in requirements documentation should be approached with caution, as it may negatively affect students' learning and creative abilities. On the other hand, the new generation of agile software developers should avoid relying too much on LLM-generated user stories because, despite generating quick solutions, these models lack a thorough understanding and the necessary creativity to document software requirements.

The rest of this paper is structured as follows. Section 2 presents background information about requirements engineering and large language models in software engineering activities. Section 3 describes the research method of this exploratory study, including its goal, research questions, steps and used material. Section 4 discusses our results focusing on answering three research questions. Section 5 discusses some threats to the study validity while Section 6 compares our study to the closest related work. Finally, Section 7 concludes this paper with directions for future work. All materials and data used in this study are available in the accompanying folder [6]

2 BACKGROUND

This section provides the theoretical and conceptual foundation for the study. We present a brief overview of requirements elicitation, the role and structure of use cases in software engineering, and the emergence and application of Large Language Models (LLMs) in the context of software development and education.

2.1 Requirements Elicitation

Requirements elicitation is a key software engineering activity in which the needs and expectations of stakeholders are identified, analyzed, and documented [36, 37]. This activity traditionally involves intense human interaction through interviews, observations, workshops, and document analysis, making it both time-consuming and susceptible to misinterpretation [34]. The quality of the requirements elicited directly impacts the success of the project, as unclear or incomplete requirements often lead to costly revisions and project delays [18].

The elicitation of traditional requirements is based on various techniques, including structured interviews, focus groups, questionnaires, observation, and document analysis [36, 37]. Each technique has distinct strengths and weaknesses and is chosen based on the specific context and situation. Requirements elicitation is a complex, multifaceted, and iterative activity that relies heavily on the communication-rich interaction between analysts and stakeholders [5]. These approaches are heavily dependent on the effectiveness of communication and the domain expertise of the participants involved [12].

Common challenges in eliciting requirements include communication barriers between technical and non-technical stakeholders, tacit knowledge that is difficult to articulate, and conflicting stakeholder viewpoints [9, 16]. Additionally, the ambiguity inherent in natural language can lead to misinterpretation of requirements, causing downstream development issues [34]. One effective approach to mitigate this ambiguity is the use of user stories [11] and use cases [8], which describe functional requirements by modeling interactions between users (actors) and the system to achieve specific goals. They focus on system behavior from the user's perspective and help identifying and structuring the system functionality [20].

2.2 Large Language Models (LLMs)

Generative Pre-trained Transformer, such as GPT-4, has demonstrated capabilities in natural language understanding, context comprehension, and text generation that makes them potentially valuable tools in various software engineering contexts [4]. Research has investigated LLM applications in different domains of software engineering, including code generation [19], bug fixing [21], and documentation creation [7]. These models can provide contextually relevant information, generate code snippets based on natural language descriptions, and help automate routine documentation tasks [7]. Their ability to process natural language makes them particularly suitable for tasks involving communication and documentation [2].

3 STUDY DESIGN

This section explores the overall design of the study, detailing the research goals and questions, participant characteristics, and proposed hypotheses. It also outlines the methods and materials used for data collection, as well as the planned for data analysis.

3.1 Goal and Research Questions

This experimental study aims to investigate the impact of using large language models as a tool to document user stories. Specifically, it analyzes the influence of ChatGPT support on the creation of user stories and their corresponding use case scenarios. The analysis focuses on three aspects: production time, quality of generated artifacts, and student perceptions of LLM support for common tasks. We compare student performance with and without LLM support to provide empirical insights into the advantages, challenges, and consequences of integrating these models into software engineering curricula. This study is guided by the following research questions.

Table 1: Activity Schedule during the Experiment

Activity	Instructor	Group 01	Group 02
Pre-Experiment	Review of concepts	Complete Background and Consent	Complete Background and Consent
Task 01	TechFix scenario	Perform task without LLM	Perform task with LLM
Task 02	GreenMarket scenario	Perform task with LLM	Perform task without LLM
Post-Experiment	Provide evaluation form	Complete evaluation form	Complete evaluation form

RQ1: How do LLMs impact the time required to create user stories?

Reducing the time to perform software engineering tasks is crucial in both academic and industrial contexts. This question explores whether LLMs can accelerate requirements creation.

RQ2: How do LLMs impact the quality of the documented user stories?

The usefulness of automatically assisted artifacts depends not only on speed but also on their correctness and completeness. This question addresses whether LLMs improve or degrade the quality of the requirements documentation.

RQ3: What challenges and benefits do students perceive in using LLMs for requirements documentation?

Understanding user perception is essential for evaluating the educational value and usability of LLMs in academic environments.

3.2 Hypotheses

Based on the research questions, we defined the following hypotheses.

- **RQ1:** Impact on completion time
  - **H_0 (Null Hypothesis):** *The use of LLMs does not significantly impact on the time required to produce requirements.*
  - **H_1 (Alternative Hypothesis):** *The use of LLMs impacts on the time required to produce requirements.*
- **RQ2:** Impact on requirement quality
  - **H_0 (Null Hypothesis):** *The use of LLMs does not impact on the quality of the produced requirements.*
  - **H_1 (Alternative Hypothesis):** *The use of LLMs significantly impacts the quality of the produced requirements.*

3.3 Methods

We employed the Latin Square study design, allowing each participant to perform the tasks both with and without the support of an LLM in alternating orders. We chose this design in our study aiming to minimize learning and sequence effects. Table 1 presents the activity schedule followed during the experiment. The procedure was structured into four main phases (lines in Table 1): pre-experiment, two task sessions, and post-experiment. The first column lists the activities conducted at each phase, while the second column describes the role or actions performed by the instructor. The third and fourth columns detail the specific actions taken by Group 01 (G1) and Group 02 (G2) of participants, respectively, highlighting the counterbalanced use of LLM support across the two experimental tasks.

During the pre-experimental phase, the instructor conducted a review of the fundamental concepts and supervised the completion of the background and consent forms. The instructor then divided the participants into 2 groups of similar sizes called G1 and G2. In the first task, group G1 created the user story without the LLM, while group G2 used the tool for support. In the second task, the roles switched: G1 used the LLM, and G2 worked without it.

Before starting the tasks, the students assigned to use ChatGPT accessed the tool through a web browser and authenticated themselves to enable the collection of the interaction link between the participant and the LLM. The participants recorded the execution time for each task directly in the response form. At the end of the practical activities, all participants completed an experience evaluation questionnaire.

3.4 Materials

Figure 1 illustrates the sequential flow of how the materials were used during the experiment. The process begins with the background questionnaire and informed consent form, followed by the user story template containing use-case sections. The participants then received the problem scenario descriptions for task execution, and finally the sequence concluded with a post-experiment questionnaire to gather feedback.

Consent and Background Before the experimental tasks, the researchers informed the participants about the study objectives, the voluntary nature of participation and the anonymized handling of the data. They then provided formal consent and completed a questionnaire capturing personal information, academic level, and self-assessed familiarity with software engineering topics. The participants also reported their practical experience in software development. These data support bias control and subgroup analyses during result interpretation.

User-Story and Use-Case Template: To standardize requirement artifacts and facilitate later evaluation, we provided

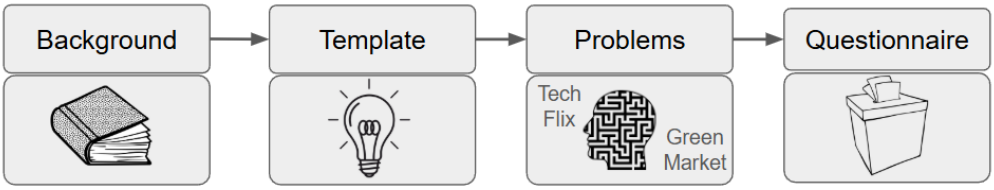

Figure 1: Chronological sequence of material usage by participants

participants with a structured template [8, 11]. The template guided them to describe user stories using the Conextra template [11] as defined below.

As a ⟨Persona⟩
I want to ⟨Objective⟩
So that ⟨Rationale⟩

In addition, both the normal and alternative flows of each use case had to be specified according to preconditions, main actions, and postconditions, based on behaviour drive development [8], as follows. The instructors illustrated and discussed this template during the review of the fundamental concepts.

Given ⟨Pre-conditions⟩
When ⟨Action⟩
Then ⟨Post-conditions⟩

Problem Scenarios: Our group of researchers developed and adapted two problem scenarios drawing on the main reference textbooks adopted in software engineering course [36, 37], with the goal of reinforcing theoretical concepts through practical requirement elicitation exercises. The first scenario, *TechFlix*, described a collaborative technical-support platform in which users report issues and specialists propose solutions. The second scenario, *GreenMarket*, portrayed an application for purchasing organic products directly from local producers. The researchers linked each practical task to one of these scenarios, ensuring that all participants worked with both contexts in different sessions.

Questionnarie: At the end of the experiment, the instructor administered a structured questionnaire to collect student feedback regarding their experience. The form prompted students to reflect on both the positive and negative aspects of the activities, with a particular focus on their interactions with ChatGPT. The researchers designed the questionnaire to capture qualitative insights into the perceived usefulness of the approach. Students actively engaged with the questionnaire and provided detailed responses. They identified strengths and weaknesses of the learning process, highlighted specific challenges they encountered, and commented on the pedagogical value of using an LLM in the context of requirements engineering. Additionally, each student shared with the the direct link to their conversation with ChatGPT, allowing the researchers to analyze the interactions in context and better understand how the tool supported or hindered their learning.

3.5 Participants Characterization

The experiment involved 28 students from two universities organized in two groups (Group 01 and Group 02) 15 students in Group 01 began the task without using a support of LLM, and 13 students in Group 02 started the task with the support of LLM. Figure 2 shows the distribution of participants' self-assessed knowledge levels across eight different areas of engineering and and quality: Object-Oriented Programming (OOP), Software Architecture, Web Technologies, Database Systems, Project Management, Requirements Engineering, Agile Methods, and the Use of Large Language Models. Each bar represents a knowledge area and is subdivided into knowledge levels ranging from 1 (no knowledge) to 5 (expert knowledge):

- Level 1: I never heard of it
- Level 2: I heard of it
- Level 3: I am familiar with the topic
- Level 4: I am able to teach my colleagues about it
- Level 5: I am an expert in the topic

The majority of participants reported an intermediate level of knowledge (Level 3) across most areas, particularly in Agile Methods, Software Architecture, and OOP. Higher proficiency levels (Level 4 and 5) were observed in Web Technologies and Use of LLMs, although few participants rated themselves at Level 5 overall. Figure 3 refers to professional experience. In this study 16 participants had no experience in software development, 7 had up to one year of experience, and 5 had between one and three years of experience.

3.6 Data Analysis

To address the research questions, we conducted inferential statistical analyses on two primary dependent variables: task completion time (RQ1) and artifact quality (RQ2). Prior to hypothesis testing, we applied the Shapiro–Wilk test to the data from each paired condition (with and without LLM support) to assess whether the distributions met the assumption of normality. The results of the Shapiro–Wilk tests indicated non-normal distributions for both dependent variables.

Given the violation of the normality assumption, we employed the Wilcoxon signed-rank test to compare the conditions. This test allowed us to evaluate whether statistically significant differences existed between the two experimental conditions regarding task completion time and artifact

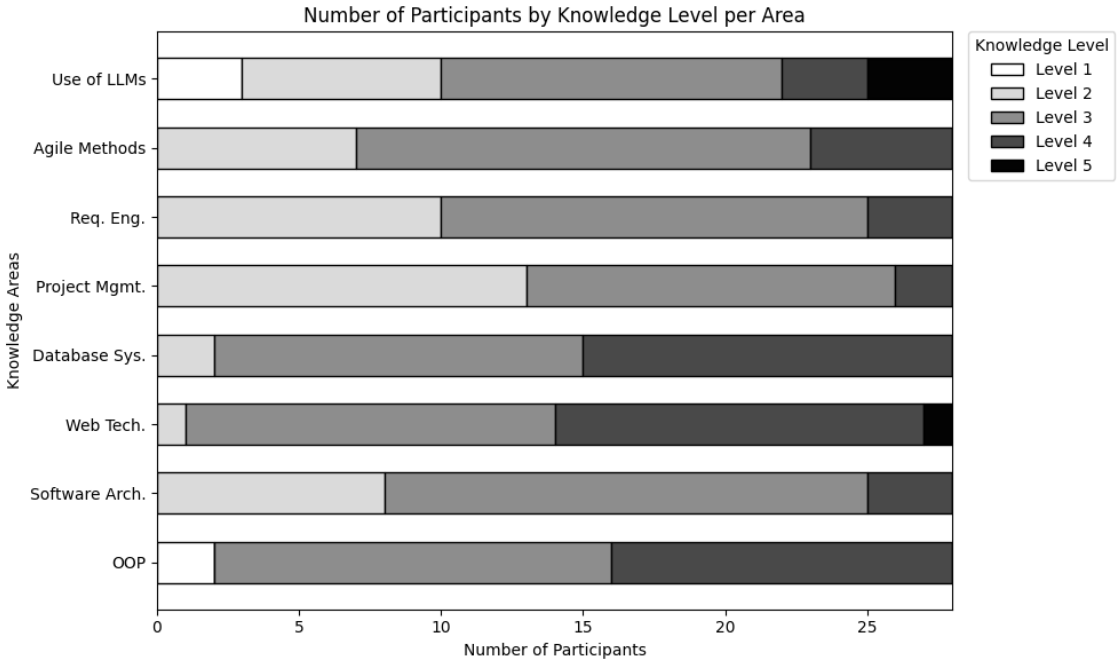

Figure 2: Details about the participants' knowledge.

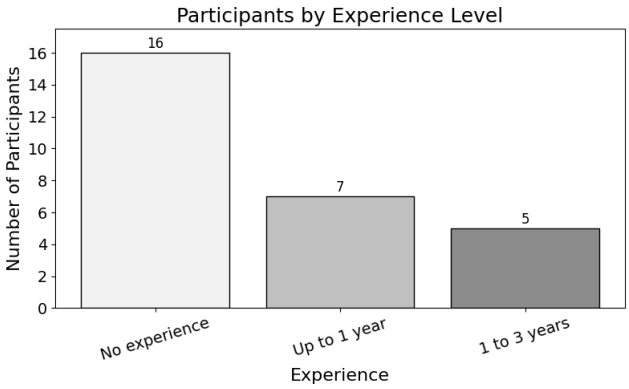

Figure 3: Details about the participants' experience.

quality. The researchers conducted all analyses using paired observations, as each participant completed the tasks under both conditions (with and without the LLM). We interpreted the results at a significance level of $\alpha = 0.05$, and reported both the test statistics and corresponding p-values for transparency and reproducibility.

To ensure the reliability of the exercise assessments, all participant responses were independently evaluated by two specialists holding a Ph.D. in the domain of software engineering. The specialists applied a predefined evaluation criteria to assign scores to each submission. The criteria followed a simple and consistent scale commonly used by the evaluator in academic settings, based on the following:

- **Score 0**: The participant did not attempt the task or failed to address the required elements.
- **Score 5**: The participant partially fulfilled the task. The reason for the score was noted in the evaluator's comments.
- **Score 10**: The participant completed the task as expected, meeting all the specified requirements.

Following the initial evaluation, the assigned scores were reviewed and discussed with the research team to validate the consistency and fairness of the assessments. The group decided to use the mean for evaluations where there was disagreement. This collaborative validation process aimed to reduce subjective bias and ensure alignment with the study's goals. They evaluated each response by assigning a score for each task and we used the Wilcoxon signed-rank test to compare the two conditions.

To investigate the perceived benefits and challenges associated with the use of LLMs, we applied a qualitative content analysis following Mayring's methodological framework [27]. This approach enabled a systematic and transparent analysis of participants' textual responses, aiming to identify, categorize, and quantify the most commonly reported difficulties and opportunities encountered when using LLMs during their tasks.

This qualitative procedure served as the primary method for answering RQ3, which focused on identifying the benefits

and challenges experienced by students when interacting with LLMs during the completion of software engineering tasks.

We initiated the analysis by conducting a close reading of all responses. We divided each response into meaningful segments, with each segment representing a distinct aspect of a participant's reported challenge. The initial coding scheme was iteratively refined through multiple passes, during which we reviewed, merged, and reorganized codes into broader thematic categories that represented key domains of perceived challenges.

To support quantitative interpretation, we calculated the frequency of each identified category by counting the number of participants who mentioned each theme. Furthermore, we documented the participant identifiers (IDs) linked to each category, enabling an assessment of the distribution of perceptions across the participant sample.

Our team stored all data in an SQL database. A RESTful API implemented in Python with FastAPI managed data retrieval and processing, while visualizations were rendered using the Matplotlib Python library.

4 RESULTS AND DISCUSSION

This section presents and discusses the results obtained from the experimental study in relation to the three research questions: RQ1 – Task Production Time, RQ2 – Artifact Quality, and RQ3 – Benefits and Challenges of Using LLMs.

4.1 Production Time (RQ1)

Figure 4 shows the results of the first RQ. As shown in the boxplot, the task completion times for the condition without LLM support exhibit a higher median and greater spread. This observation shows that participants took longer to complete the tasks without the assistance of the LLM. The condition with LLM support shows a lower median and tighter interquartile range, indicating a more consistent and faster task completion time. These results are supported by a Wilcoxon test ($p < 0.05$), suggesting an efficiency gain associated with the use of LLMs in software requirements elicitation (user stories and use cases) and aligning with findings reported in the literature that highlight the potential of LLMs to enhance productivity [35].

RQ1 Summary – Production Time

The analysis revealed a reduction in task completion time, from 12 to 6 minutes on average, when participants received support from LLMs. The Wilcoxon signed-rank test confirmed a statistically significant difference.

4.2 Artifact Quality (RQ2)

Figure 5 presents the distribution of student performance grouped by the use of LLMs, based on the selected metric (e.g., task completion time or artifact quality). We generated a boxplot to visually summarize the central tendency and dispersion of the scores for both conditions. Additionally, the arithmetic mean was computed for each group and marked with a black square to indicate the average performance. This visual indicator allows a quick comparison between the central tendency (mean) and the median, providing insight into the symmetry or skewness of the data distribution. The vertical axis emphasize differences between groups and ensure consistent visual interpretation. From this visualization, variations in performance between the two experimental conditions can be inferred, supporting the statistical analysis presented in the following sections.

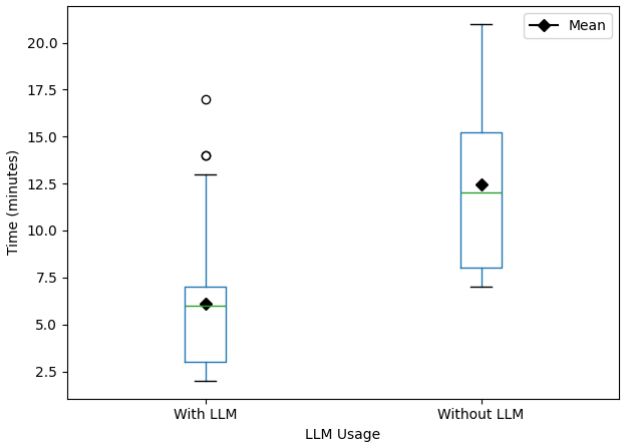

Figure 4: Comparison of task completion time with and without LLM support.

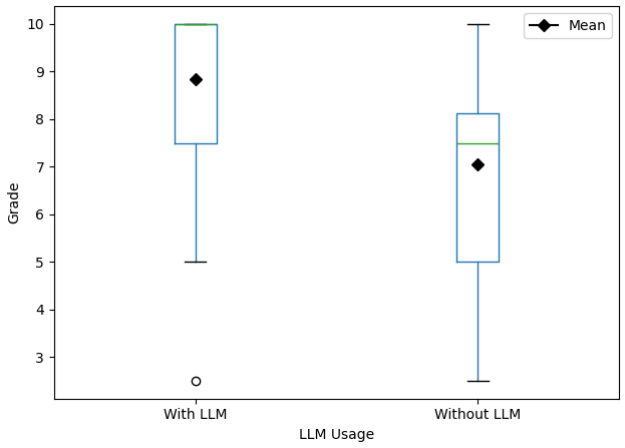

Figure 5: Comparison of artifact quality scores between conditions with and without LLM support by specialist.

RQ2 Summary – Artifact Quality

Descriptive statistics show that artifacts produced under LLM support receiving higher scores (mean = 8.8) compared to those created without LLM support (mean = 7.0) when evaluated by a software engineering specialists. The Wilcoxon signed-rank test revealed a statistically significant difference in artifact scores between conditions ($p = 0.00143$).

4.3 Benefits and Challenges of LLMs (RQ3)

Figure 6 presents six categories of challenges, which we organized based on Mayring's methodological framework [27], explained in Section 3.6. The number into the circle represent the count of participants opinion. The most commonly reported issue was *overdependence* (13 mentions). Several students expressed concerns that relying on LLMs can discourage deep engagement with the task and hinder the development of analytical and problem-solving skills. This concern aligns with broader discussions in the literature on how automation may negatively affect active learning processes [10, 23, 30].

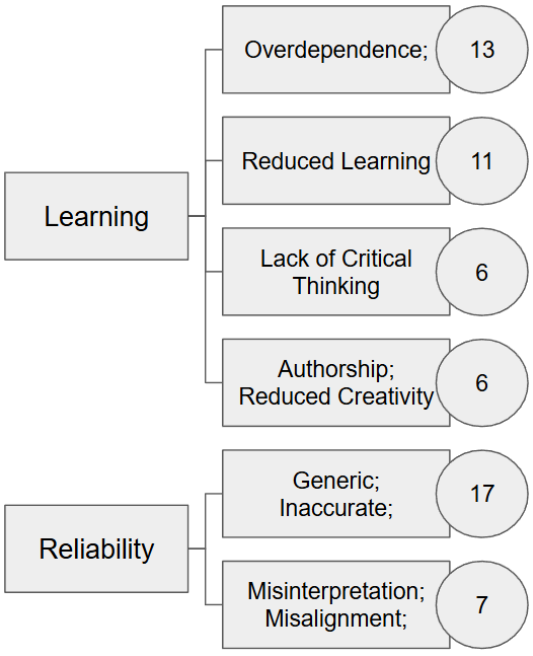

Figure 6: Perceived challenges of using LLMs in user story documentation.

Closely related to this, participants frequently noted *generic or inaccurate responses* (17 mentions). They pointed out that the model often gave superficial answers that lacked domain-specific depth. In some cases, it generated responses that seemed plausible but contained factual errors or misunderstood the context, requiring additional human validation.

Another critical concern was the *'Reduced Learning' and 'lack of critical thinking and review'* (18 mentions). Students indicated a tendency to accept the model's output, which could lead to unverified or incorrect information being incorporated into their work. This passive acceptance poses risks to the accuracy and quality of requirements documentation.

Several responses also pointed out to *Authorship* concerns (4 mentions), where students felt that their own ideas and voice were overshadowed by the tool's suggestions. Similarly, participants raised concerns about *misinterpretation or misalignment* with context (7 mentions), especially when the model failed to clearly understand the prompts or project constraints. Finally, participants cited reduced creativity (2 mentions) as a subtle but important drawback. Some students felt that prolonged use of LLMs hindered their ability to think creatively or explore alternative solutions independently.

Figure 7 presents a hierarchical classification of the perceived benefits reported by students when using Large Language Models (LLMs) in requirements elicitation activities. We grouped the responses into two main categories—Time and Quality—to reflect the nature of the benefits participants experienced during the tasks. Each benefit includes a frequency count, displayed within the circular markers, indicating how many participants mentioned the respective theme.

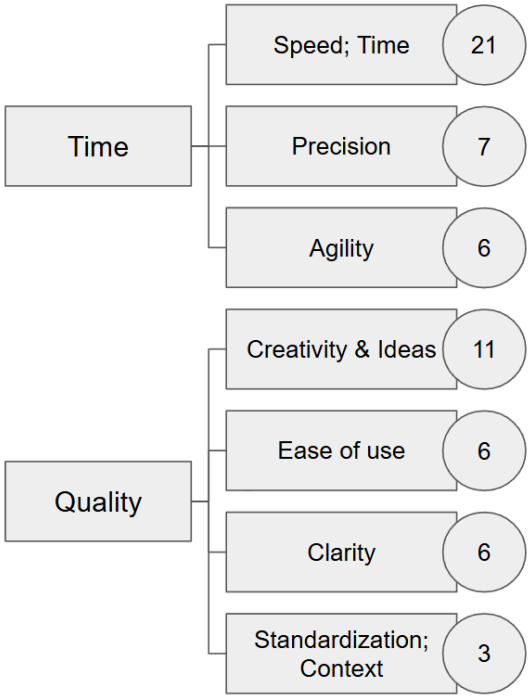

Figure 7: Perceived benefits of using LLMs in user story documentation.

The students feedback confirms that the use of LLMs accelerates the requirements elicitation process (RQ1) and enhances the quality of the generated artifacts (RQ2). The

significant reduction in completion time supports prior work [31] indicating that LLMs facilitate repetitive tasks and initial text formulation, thereby freeing students to focus on higher-order cognitive activities.

Figure 7 indicates that the majority of respondents experienced time savings. This observation is further supported by quantitative data presented in Figures 4 and 5, which shows that the time required decreased from 12 minutes to 6 minutes, representing a 50% reduction. In terms of quality, the ratings improved from 70% to 88% when evaluated by specialist.

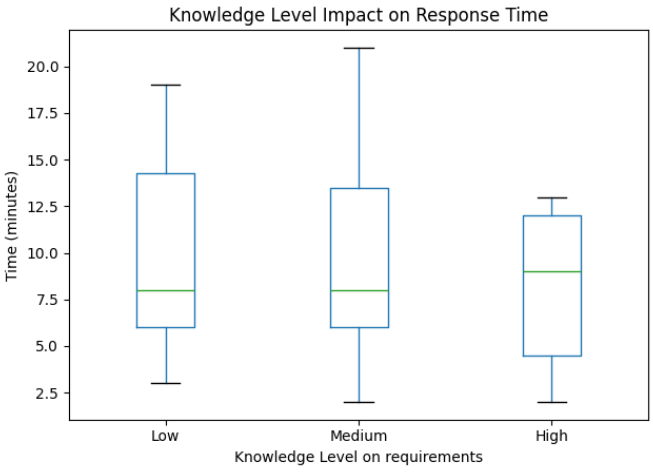

Figure 8: Task completion time, using LLM, compared to specific knowledge

RQ3 Summary – Benefits and Challenges Perceived

The primary benefit is the increased agility in completing tasks. However, some major challenges identified with using large language models (LLMs) to document user stories include excessive reliance on the tool, vague responses, and a lack of critical thinking regarding the LLM solution.

4.4 Discussion

Figure 8 illustrates that the participant with advanced requirements engineering knowledge dedicated more time to the task compared to the student with limited understanding of Requirements Engineering. This trend also appears with other software engineering knowledge areas, such as LLM use and Agile Software Development. It suggests that participants with more specialized knowledge in the task tend to take longer, as they make adjustments to the final output after processing information from the LLM.

Finally, the general comment section of the post-experiment questionnaire provided valuable insights into how participants perceived the role of LLMs in their learning experience. Some participants actively recognized both the benefits and the risks associated with using these models.

Several participants emphasized the positive impact on learning efficiency and conceptual understanding. For example, P001 reported that *"it facilitates understanding of concepts and accelerates the resolution of doubts."* Similarly, P011 highlighted that *"the LLM can accelerate learning by providing clear explanations, personalized examples, and immediate answers."* P018 offered a comprehensive view, stating that *"the LLM acts as a facilitator that offers quick explanations, practical examples, creative suggestions, and immediate feedback,"* which supports the consolidation of knowledge and helps users explore content from different perspectives. P020 reinforced this by noting that *"LLMs can positively influence learning by helping users better understand structures, explore new ideas, and reinforce previously acquired knowledge."* In addition, P026 remarked that *"LLMs significantly accelerate the learning process by democratizing access to knowledge and presenting it in more accessible language."*

At the same time, participants expressed concerns about the potential drawbacks of over-reliance on LLMs. P010 warned that *"it can impair logical reasoning, and the user may become dependent,"* while P013 noted that *"the lack of need to research and read reduces the knowledge acquired during the learning process."* P009 offered a balanced view, saying, *"if the person only copies the answers, they may end up not truly learning."* P024 shared a similar concern, observing that *"LLMs can hinder learning overall, as students may fail to retain content and become dependent on the tool."* P028 also noted that *"when users do not read and merely copy and paste, AI can weaken the learning process."*

To address these challenges, some participants advocated for conscious and critical use of LLMs. P005 emphasized that *"the LLM can accelerate learning when used as a supporting tool, but its use must be conscious to avoid replacing individual reflection."* P012 echoed this sentiment by stating, *"if used as an auxiliary tool and not as the main source, it can be a great ally."* Similarly, P017 stressed that *"if used correctly to learn and not just receive ready-made answers, it has a very positive influence."* In the same spirit, P021 commented that *"LLMs serve as facilitators of learning by providing explanations, but overuse can lead to laziness in the learning process."* P027 added that *"if used correctly and with proper supervision, LLMs can help educators deliver content more efficiently while also supporting student understanding."*

These perspectives revealed that participants not only leveraged LLMs to enhance their learning but also demonstrated a reflective understanding of their limitations. By integrating these qualitative insights with the quantitative findings, we reinforce the conclusion that the educational value of LLMs depends less on their capabilities and more on how learners engage with them—balancing their use as powerful aids with the need to preserve autonomy, critical thinking, and active knowledge construction.

5 THREATS TO VALIDITY

This exploratory study on the application of LLMs in Requirements Elicitation with developers has some limitations. The main threats and our respective actions to mitigate them are discussed below based on the proposed categories of Wohlin et al. [39].

Construct Validity. Construct validity reflects the extent to which operational measures that are studied represent what the researchers have in mind and what is investigated according to the RQs [39]. This kind of threat can occur in formulating the questionnaire with the set of tasks and questions for each group to answers with or/and without LLM during the experiment. To mitigate this threat, we thoroughly reviewed and discussed all experimental procedures. The first version of this questionnaire was piloted with five participants. After these pilot experiment, we reformulated the questionnaire to ensure that the questionnaire and time were sufficient to generate data to answer our research questions. Besides, to reduce the learning effect on the experiment results, we used the Latin square to distribute the tasks and LLM between two groups of participants.

Internal Validity. The internal validity is related to uncontrolled aspects (e.g., course period and level of knowledge) that may affect the results [39]. In this sense, a limitation of this study concerns the absence of balancing the participants in groups according to their knowledge. It can be argued that the level of knowledge of some participants may not reflect the state of practice (e.g., most participants have only minor knowledge of requirements elicitation). To minimize this threat, we applied the experiment to students enrolled in the software engineering course only after studying the content related to the elicitation of software requirements. Another threat is the use of statistical tools. When reporting our results, we paid particular attention to the suitable use of statistical tests (i.e., the Wilcoxon test), decreasing the possibility that our findings are due to random events.

External Validity. The external validity concerns the generalizability of results to other contexts [39]. The primary threats to external validity in this study stem from using LLMs. First, since our experiment relies solely on ChatGPT, our findings may not generalize to other LLMs. Second, analyzing a different LLM could yield divergent results. Finally, the inherent stochasticity of generative AI introduces variability: identical inputs may produce various outputs, and users cannot precisely control model behavior. This unpredictability may also affect the reliability of our experimental outcomes and conclusions.

Conclusion Validity. The conclusion validity concerns issues that affect the ability to draw the correct conclusions from the study [39]. The prior threat relate to this category in the analytical approach used to interpret the experimental results. To address this, we employed descriptive statistics and statistical hypothesis tests in our discussion. Additionally, all researchers participated in the data analysis and interpretation to minimize individual bias. However, the collected data may still have undiscovered or unreported issues despite these measures.

6 RELATED WORK

LLM have been used to provide solutions to traditional software engineering problems in several areas, such as development, testing, design, and maintenance [15].

Regarding software requirements, there is a significant number of works that address this topic. For instance, Marques et al. [25] and Vasudevan and Reddivari [38] performed literature reviews by exploring LLMs applications in requirements engineering aiming to understand how LLMs can help improve various activities, such as requirements elicitation, analysis, modeling, validation, specification, prioritization, and tracing. In general, ChatGPT was widely adopted in primary studies, indicating considerable advantages to support requirements engineering activities. However, as far as we know, none of the work focused on LLM as a help in the requirements engineering education process.

In contrast, Sampaio et al. [32] carried out an experiment involving the use of LLM in requirements engineering education. In their study [32], students evaluated, according to their perceptions, functional and nonfunctional requirements generated from ChatGPT with respect to equivalence, innovation, adequacy and importance indicators. Although our work also uses ChatGPT, one researcher evaluated the quality of software requirements because we believe students may not have enough knowledge to evaluate them.

We realize that there is a gap in the literature that specifically addresses LLM in the teaching of requirement engineering. However, some previous studies use it to support software engineering education in general. For example, Daun and Brings [14] discussed the transformative potential of ChatGPT in software engineering education, highlighting both the challenges and opportunities posed by generative artificial intelligence. They explore how artificial intelligence can enhance individualized learning experiences and support the development of novel pedagogical approaches. Xue et al. [40] conducted a controlled experiment involving computer science students, aiming to shed light on ChatGPT's potential influence on their learning outcomes in introductory programming. Finally, Pereira et al. [29] investigated LLM within the context of software engineering education, focuses on developing a set of prompt examples specifically designed for software engineering education.

In summary, the literature shows a growing interest in leveraging LLMs to support various activities in software engineering, including requirements engineering and educational contexts. While several studies have explored the application of LLMs to assist in requirements engineering tasks and a few have investigated their use in software engineering education more broadly [14, 29, 32, 40], there remains a notable research gap at the intersection of these two domains, the use of LLMs to support the teaching and learning of requirements engineering. This gap highlights the need for further investigation into how generative artificial intelligence tools,

such as ChatGPT, can be effectively integrated into software requirement education to enhance student engagement, understanding, and skill development.

7 CONCLUSION AND FUTURE WORK

Requirements elicitation is a software development phase characterized by high complexity, significant time investment, and heavy dependence on communication between participants. In this context, large language models, such as ChatGPT, have emerged as promising tools, capable of assisting in understanding and documenting requirements in a more agile and structured way.

However, in our study, participants reported several challenges associated with using ChatGPT. The main one was excessive dependence on the LLM, which can compromise deeper learning of content. Additionally, generic or imprecise responses were observed, requiring additional validation by students. Another critical point was the lack of critical review of generated responses, leading to passive acceptance of incorrect information. There were also reports of loss of personal authorship, with students feeling their ideas were being overshadowed by the model's suggestions, as well as problems with incorrect interpretation of context by the tool. Therefore, although LLMs present relevant benefits in terms of productivity, it is important that their use be guided as a complement to critical thinking, contextual analysis, and active knowledge construction, preserving students' autonomy and intellectual development.

This initial study highlights the potential of LLMs to support requirements elicitation, yet several avenues for future research remain open. Expanding the participant pool across diverse academic contexts could improve the generalizability of findings and uncover how individual differences impact LLM effectiveness. We also propose future longitudinal studies to help assess the sustained impact of LLM use on learning and strategy development. Investigating varied conditions, such as time constraints, task structures, and collaboration modes, could reveal when LLMs are most beneficial. Finally, we believe that integrating LLMs with other AI tools may enhance the elicitation process and overcome current limitations.

ACKNOWLEDGMENTS

This research was partially supported by the Brazilian funding agencies: CAPES, CNPq (Grant 305171/2025-9), and FAPEMIG (Grant APQ-01488-24).

REFERENCES

[1] Muhammad Ovais Ahmad, Jouni Markkula, and Markku Oivo. 2013. Kanban in software development: A systematic literature review. In *2013 39th Euromicro conference on software engineering and advanced applications*. IEEE, 9–16.

[2] Vincenzo Ambriola and Vincenzo Gervasi. 1997. Processing natural language requirements. In *Proceedings 12th IEEE International Conference Automated Software Engineering*. IEEE, 36–45.

[3] Anis R. Amna and Geert Poels. 2022. Systematic Literature Mapping of User Story Research. *IEEE Access* 10 (2022), 51723–51746. https://doi.org/10.1109/ACCESS.2022.3173745

[4] Yadagiri Annepaka and Partha Pakray. 2024. Large language models: A survey of their development, capabilities, and applications. *Knowledge and Information Systems* (2024), 1–56.

[5] Aybüke Aurum and Claes Wohlin. 2005. *Engineering and managing software requirements*. Vol. 1. Springer.

[6] Anonymous Authors. 2025. Materials and Data for "Documenting User Stories: Does LLM Help?". https://drive.google.com/drive/folders/1MKqPh2C2Z0hdG3kWK36M8h6mdQIGT-Ad. Accessed: 2025-05-20.

[7] Lenz Belzner, Thomas Gabor, and Martin Wirsing. 2023. Large language model assisted software engineering: prospects, challenges, and a case study. In *International Conference on Bridging the Gap between AI and Reality*. Springer, 355–374.

[8] Leonard Peter Binamungu and Salome Maro. 2023. Behaviour driven development: A systematic mapping study. *Journal of Systems and Software* 203 (2023), 111749.

[9] Larissa Chazette and Kurt Schneider. 2020. Explainability as a non-functional requirement: challenges and recommendations. *Requirements Engineering* 25, 4 (2020), 493–514.

[10] 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 *Proceedings of the IEEE/ACM 46th international conference on software engineering*. 1–13.

[11] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development*. Addison-Wesley.

[12] Jane Coughlan, Mark Lycett, and Robert D Macredie. 2003. Communication issues in requirements elicitation: a content analysis of stakeholder experiences. *Information and Software Technology* 45, 8 (2003), 525–537.

[13] Fabiano Dalpiaz and Sjaak Brinkkemper. 2018. Agile Requirements Engineering with User Stories. In *IEEE 26th International Requirements Engineering Conference (RE)*. 506–507. https://doi.org/10.1109/RE.2018.00075

[14] Marian Daun and Jennifer Brings. 2023. How ChatGPT Will Change Software Engineering Education *(ITiCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 110–116. https://doi.org/10.1145/3587102.3588815

[15] Görkem Giray. 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software* 180 (2021), 111031. https://doi.org/10.1016/j.jss.2021.111031

[16] Carol K Gonzales and Gondy Leroy. 2011. Eliciting user requirements using appreciative inquiry. *Empirical Software Engineering* 16 (2011), 733–772.

[17] Dulaji Hidellaarachchi, John Grundy, Rashina Hoda, and Ingo Mueller. 2023. The influence of human aspects on requirements engineering-related activities: Software practitioners' perspective. *ACM Transactions on Software Engineering and Methodology* 32, 5 (2023), 1–37.

[18] Hubert F Hofmann and Franz Lehner. 2001. Requirements engineering as a success factor in software projects. *IEEE software* 18, 4 (2001), 58.

[19] Dong Huang, Jie M Zhang, Qingwen Bu, Xiaofei Xie, Junjie Chen, and Heming Cui. 2024. Bias testing and mitigation in llm-based code generation. *ACM Transactions on Software Engineering and Methodology* (2024).

[20] Ivar Jacobson. 1995. The Use-Case Construct in Object-Oriented Software Engineering. In *Scenario-Based Design: Envisioning Work and Technology in System Development*, John M. Carroll (Ed.). Wiley, New York, 309–336.

[21] Matthew Jin, Syed Shahriar, Michele Tufano, Xin Shi, Shuai Lu, Neel Sundaresan, and Alexey Svyatkovskiy. 2023. Inferfix: End-to-end program repair with llms. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1646–1656.

[22] Junaed Younus Khan and Gias Uddin. 2023. Automatic Code Documentation Generation Using GPT-3. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. Article 174, 6 pages. https://doi.org/10.1145/3551349.3559548

[23] Jinhee Kim, Seongryeong Yu, Rita Detrick, and Na Li. 2025. Exploring students' perspectives on Generative AI-assisted academic writing. *Education and Information Technologies* 30, 1 (2025), 1265–1300.

[24] Qianou Ma, Weirui Peng, Chenyang Yang, Hua Shen, Kenneth Koedinger, and Tongshuang Wu. 2025. What Should We Engineer

in Prompts? Training Humans in Requirement-Driven LLM Use. *ACM Transactions on Computer-Human Interaction (TOCHI)* (2025). https://doi.org/10.1145/3731756

[25] Nuno Marques, Rodrigo Rocha Silva, and Jorge Bernardino. 2024. Using ChatGPT in Software Requirements Engineering: A Comprehensive Review. *Future Internet* 16, 6 (2024). https://doi.org/10.3390/fi16060180

[26] Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. 2023. On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot. In *IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. 2149–2160. https://doi.org/10.1109/ICSE48619.2023.00181

[27] Philipp Mayring. 2014. *Qualitative Content Analysis: Theoretical Foundation, Basic Procedures and Software Solution*. Beltz, Weinheim, Germany. https://nbn-resolving.org/urn:nbn:de:0168-ssoar-395173 Accessed May 2025.

[28] Henrique Nunes, Eduardo Figueiredo, Larissa Soares, Sarah Nadi, Fischer Ferreira, and Geanderson Santos. 2025. Evaluating the Effectiveness of LLMs in Fixing Maintainability Issues in Real-World Projects. In *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*.

[29] Juanan Pereira, Juan-Miguel López, Xabier Garmendia, and Maider Azanza. 2024. Leveraging Open Source LLMs for Software Engineering Education and Training. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*. 1–10. https://doi.org/10.1109/CSEET62301.2024.10663055

[30] James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, et al. 2025. Beyond the hype: A comprehensive review of current trends in generative AI research, teaching practices, and tools. *2024 Working Group Reports on Innovation and Technology in Computer Science Education* (2025), 300–338.

[31] Sanka Rasnayaka, Guanlin Wang, Ridwan Shariffdeen, and Ganesh Neelakanta Iyer. 2024. An empirical study on usage and perceptions of llms in a software engineering project. In *Proceedings of the 1st International Workshop on Large Language Models for Code*. 111–118.

[32] Savio Sousa Sampaio, Márcia Sampaio Lima, Eriky Rodrigues de Souza, Maria Alcimar Meireles, Marcela Savia Pessoa, and Tayana Uchoa Conte. 2024. Exploring the Use of Large Language Models in Requirements Engineering Education: An Experience Report with ChatGPT 3.5 *(SBQS'24)*. Association for Computing Machinery, New York, NY, USA, 624–634. https://doi.org/10.1145/3701625.3701687

[33] Ken Schwaber and Jeff Sutherland. 2011. The scrum guide. *Scrum Alliance* 21, 1 (2011), 1–38.

[34] Unnati S Shah and Devesh C Jinwala. 2015. Resolving ambiguities in natural language software requirements: a comprehensive survey. *ACM SIGSOFT Software Engineering Notes* 40, 5 (2015), 1–7.

[35] Álvaro F Silva, Alexandra Mendes, and João F Ferreira. 2024. Leveraging large language models to boost Dafny's developers productivity. In *Proceedings of the 2024 IEEE/ACM 12th International Conference on Formal Methods in Software Engineering (FormaliSE)*. 138–142.

[36] Ian Sommerville and Pete Sawyer. 1997. *Requirements Engineering: A Good Practice Guide* (1st ed.). John Wiley & Sons, Inc., USA.

[37] Marco Tulio Valente. 2020. Engenharia de software moderna. *Princípios e práticas para desenvolvimento de software com produtividade* 1, 24 (2020).

[38] Poonkuzhali Vasudevan and Sandeep Reddivari. 2025. The Role of Generative AI Models in Requirements Engineering: A Systematic Literature Review. In *Proceedings of the 2025 ACM Southeast Conference* (Southeast Missouri State University, Cape Girardeau, MO, USA) *(ACMSE 2025)*. Association for Computing Machinery, New York, NY, USA, 188–194. https://doi.org/10.1145/3696673.3723053

[39] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, and Björn Regnell. 2012. *Experimentation in Software Engineering*. Springer.

[40] Yuankai Xue, Hanlin Chen, Gina R. Bai, Robert Tairas, and Yu Huang. 2024. Does ChatGPT Help With Introductory Programming?An Experiment of Students Using ChatGPT in CS1. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training* (Lisbon, Portugal) *(ICSE-SEET'24)*. Association for Computing Machinery, New York, NY, USA, 331–341. https://doi.org/10.1145/3639474.3640076

Received 30 July 2025; revised 19 September 2025; accepted 13 October 2025