A Large Language Model as a Support Tool for User Story Creation with Use Cases: An Empirical Study

Pedro Garcia
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
pedro.clair@dcc.ufmg.br

Eduardo Figueiredo
Federal University of Minas Gerais (UFMG)
Belo Horizonte, Brazil
figueiredo@dcc.ufmg.br

Kattiana Constantino
Federal University of Jequitinhonha and Mucuri Valleys (UFVJM)
Diamantina, Brazil
kattiana.constantino@ufvjm.edu.br

Filipe Fernandes
Federal Institute of the Southeast of Minas Gerais (IF Sudeste MG)
Manhuaçu, Brazil
filipe.fernandes@ifsudestemg.edu.br

Filipe Roseiro Côgo
Queen's University
Kingston, Canada
filipe.cogo@gmail.com

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 ChatGPT, not only on the productivity and quality of requirements artifacts, such as user stories, but also as a requirements elicitation tool. 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 with use cases. Based on the Latin Square design, students created user stories with and without LLM support. We then analyzed the effects in terms of completion time, solution quality, students reported benefits and challenges, and interactions between students and the LLM. 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 issues with using the LLM, such as becoming overly dependent on it and losing creativity. We identified a recurrent support-oriented interaction pattern between users and the LLM. Our findings suggest that educators and software professionals should exercise caution when using LLMs, as their use may negatively affect student learning and undermine a thorough understanding of specific software requirements.

Keywords

Support Tool, LLM, Requirements, User Story, Use Cases

1 Introduction

Requirements engineering is a key activity aiming at discovering, documenting and managing the requirements of a software system [48]. This activity involves intense human interaction through interviews, observations, workshops, and document analysis, making it both time-consuming and susceptible to misinterpretation [27]. The widespread adoption of agile software development practices, such as Scrum [45] and Kanban [2], has led to increased use of user stories for requirements elicitation [21]. In addition, previous research [5] indicated that 59% of agile software developers used the Connextra [13] structured template as a supporting tool to write user stories.

Recent advances in Large Language Models (LLMs) have boosted interest in their use for software engineering (SE) problems [18, 19, 34], such as code generation [37], software quality [38], and software documentation [32]. Their ability to understand complex contexts allows LLMs to generate flexible and adaptable solutions [38] for such problems. In particular, LLMs offer a promising avenue for enhancing human-intensive SE activities [35], 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 of templated user stories by LLMs [21].

Previous work [3, 36, 51] conducted literature reviews focusing on the application of LLMs into requirements engineering contexts and educational contexts [24] identified key opportunities and limitations. Previous research [44] also conducted empirical studies to evaluate the use of LLMs in requirements engineering education. However, to our knowledge, no prior work has quantitatively and qualitatively evaluated the benefits and challenges of using LLMs to document use stories.

Beyond the technical evaluation of LLMs as requirements engineering tools, this paper is fundamentally motivated by an educational concern: understanding how the use of LLMs impacts the learning process of the next generation of software engineers. Requirements elicitation is a discipline that demands analytical reasoning, creative thinking, and contextual judgment — competencies that are developed through deliberate practice and critical engagement with problem domains. As LLMs become increasingly accessible to students, there is a growing risk that their adoption in educational settings may shortcut this learning process rather than support it.

This concern is consistent with broader discussions in the Software Engineering Education literature. Studies such as [22] have highlighted both the opportunities and risks of integrating generative AI into SE courses, while [11] and [40] have raised concerns about how automation may negatively affect active learning in computing education. More specifically, [44] investigated student perceptions of LLM-generated requirements, and [53] examined the

impact of ChatGPT on introductory programming courses. However, to our knowledge, no prior work has empirically evaluated — through a controlled experiment with specialist quality assessment and interaction analysis — how LLM use affects both the artifacts and the learning behavior of students in requirements engineering education.

This paper presents an experimental study investigating the impact of using LLMs on the documentation of software requirements. In particular, we designed a controlled study in which 28 undergraduate students enrolled in SE courses documented two use cases using the Connextra template for user stories [13]. We employed the Latin Square design, allowing each participant to perform the tasks with and without LLM support (i.e., the Web interface of 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 produced user stories, (iii) students' perceptions of using an LLM, and (iv) interaction patterns between users and the model during user story and use case construction.

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 other, more surprising results. Manual inspections of the quality of the generated requirements, conducted by two SE professors, uncover major quality problems in the LLM-supported use stories. For instance, we observed that students using the LLM not only documented requirements in a more generic way but also provided 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 were overly dependent and lacked creativity in creating user stories and use cases. We also observed a recurring pattern in the conversations between users and the LLM: During the interaction between the user and the model, the user first provides the task statement, context, and template. The model then uses this information to contextualize the task, generate responses, and provide support. So, students should avoid relying too heavily on LLM-generated user stories because, while these models can produce quick solutions, they lack the thorough understanding and creativity required to properly document software requirements.

The rest of this paper is structured as follows. Section 2 presents background information about requirements engineering and LLMs in SE activities. Section 3 describes the research method of this experimental study, including its goal and research questions, materials, and methods. Section 4 discusses our results, focusing on answering four research questions, followed by Section 5, which discusses our results. Section 6 discusses some threats to the study's validity, while Section 8 compares our study to related work. Finally, Section 8 concludes this paper with directions for future work. All materials and data used in this study are available in the accompanying replication package [25].

2 Background

This section presents an overview of requirements elicitation, the role and structure of use cases in SE, and the emergence and application of LLMs in software development and education.

2.1 Requirements Elicitation

Requirements elicitation is a fundamental activity in SE, during which the stakeholders' needs and expectations are discovered, examined, and formally recorded [48]. 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 [28]. Requirements elicitation is a complex, multifaceted, and iterative activity that relies heavily on the communication-rich interaction between analysts and stakeholders [7]. Additionally, the inherent ambiguity of natural language can lead to misinterpretations of requirements, resulting in downstream development issues [46]. A widely adopted strategy to reduce such ambiguity is the use of structured requirement representations, such as user stories [13]:

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

and use cases expressed through Behavior-Driven Development (BDD) scenarios [9]:

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

These artifacts describe functional requirements by explicitly modeling interactions between users (actors) and the system to achieve specific goals. By emphasizing system behavior from the user's perspective, they support clearer identification, organization, and validation of system functionality [30].

2.2 LLMs for SE

Generative Pre-trained Transformer, such as GPT-4, has demonstrated capabilities in natural language understanding, context comprehension, and text generation that make them potentially valuable tools in various SE contexts [6]. Research has investigated LLM applications in different domains of SE, including code generation [29], bug fixing [31], and documentation creation [8]. These models can provide contextually relevant information, generate code snippets from natural language descriptions, and automate documentation tasks [8]. Their ability to process natural language makes them particularly suitable for tasks involving communication and documentation [4].

3 Study Design

This section explains our study's design, detailing the research goals and questions, participants' demographics, and hypotheses. It also outlines the methods and materials used for data collection, as well as the plan for data analysis.

3.1 Goal and Research Questions

In particular, the study analyzes the influence of using ChatGPT on the creation of user stories and their corresponding use case scenarios, following the Conextra format [13] and writing BDD specifications using Gherkin templates [9]. The study adopts a

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

mixed-method approach. Quantitatively, it evaluates student performance with and without LLM support in terms of production time, quality of the generated artifacts, and student perceptions of LLM assistance. Qualitatively, it adopts thematic analysis [12], inspired by human–machine interaction studies and visualization-based approaches [10], to examine how students interact with LLMs during conversational sessions. This qualitative analysis focuses on the student–LLM interaction process, including how students formulate prompts, interpret AI-generated responses, and iteratively refine their dialogue strategies as they create user stories and use case scenarios. Based on these goals, the study is guided by the following research questions:

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 how much 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 user story creation?

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

RQ4: How do students interact with LLMs when creating user stories and use cases?

This question explores the interaction between students and LLMs through thematic analysis, focusing on prompt formulation, response interpretation, iterative refinement, and adaptation of strategies during the activity.

3.2 Hypotheses

Based on the research questions, we defined the following hypotheses for RQ1 and RQ2. We do not define hypotheses for RQ3 and RQ4 because they are based in qualitative analysis.

RQ1: Impact on completion time:

- **H_0 (Null Hypothesis):** *The use of LLMs does not impact the time required to produce requirements.*
- **H_1 (Alternative Hypothesis):** *The use of LLMs impacts the time required to produce requirements.*

RQ2: Impact on requirement quality:

- **H_0 (Null Hypothesis):** *The use of LLMs does not impact the quality of the produced requirements.*
- **H_1 (Alternative Hypothesis):** *The use of LLMs impacts the quality of the produced requirements.*

3.3 Methods

Table 1 presents the activity schedule followed during the experiment. We employed the Latin Square study design [1], allowing each participant to perform the tasks both with and without LLM support in alternating orders. We chose this design in our study, aiming to minimize learning and sequence effects, in line with prior studies using Latin Square [14, 16, 17]. The procedure was structured into four activities (lines in Table 1): pre-experiment, two task sessions, and post-experiment. The first column lists the conducted activities, while the second column describes the instructor's role or actions. The third and fourth columns outline the specific actions taken by Group 01 (e.g., Task 01 without LLM support) and Group 02 (Task 01 with LLM support) of participants.

During the pre-experiment activity, the instructor reviewed 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, G1 created the user story without the LLM, while 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 tasks, all participants completed an experience evaluation questionnaire.

3.4 Materials

Figure 1 illustrates the materials and how they 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.

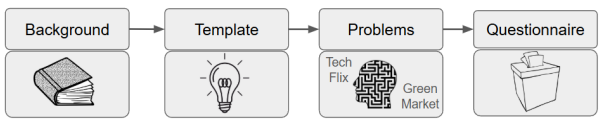

Figure 1: Chronological sequence of used material

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 demographic information, academic level, and self-assessed familiarity with SE topics. The participants also reported their practical experience in software development. These data support bias control and subgroup analyses during results interpretation.

Template. To standardize requirement artifacts and facilitate later evaluation, we provided participants with a structured template [9, 13]. The template guided them to describe user stories using the Conextra template [13] as defined in the background section. 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 BDD [9]. The instructors introduced and discussed this template during the review of fundamental concepts, after students had been exposed to requirements engineering in the course in which they were enrolled.

Problems: Our research group is composed of four PhD holders and one Master's degree holder in the field of software engineering developed and adapted two subject systems described on a prominent SE textbook [48], with the goal of reinforcing theoretical concepts through practical requirements elicitation exercises. The first subject system, *TechFlix*, described a collaborative technical-support platform in which users report issues and specialists propose solutions. The second subject system, *GreenMarket*, portrayed an application for purchasing organic products directly from local producers. The researchers linked each practical task to one of these scenarios, eEnsuring that all participants worked with both contexts under different conditions (see Table 1).

Questionnaire : At the end of the intervention, the instructor administered a structured questionnaire to collect students' 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 to learn templated requirements documentation. Additionally, each student (who consented) shared the link to their conversations with ChatGPT, allowing the researchers to analyze the interactions in context and better understand whether and how the tool supported or hindered their learning.

3.5 Participants Demographics

The experiment involved 28 students from two universities, with Group 01 at each university having two more members than Group 02. Specifically, 15 students in Group 01 began the task without the support of a LLM, while 13 students in Group 02 commenced the task using the support of an LLM. Figure 2 shows the distribution of participants' self-assessed knowledge levels across eight different areas of engineering and quality, for example: Object-Oriented Programming (OOP), Web Technologies, 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):

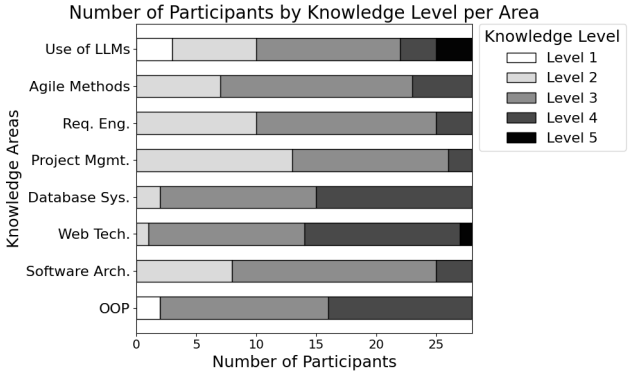

Figure 2: Details about the participants' knowledge.

- 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.

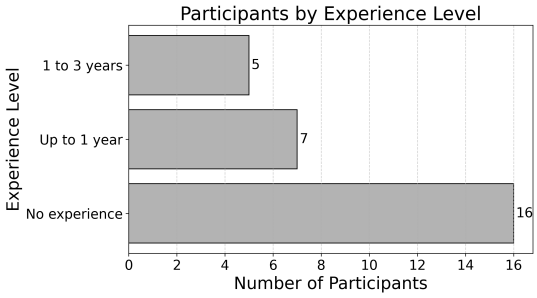

Figure 3: Professional experience of the participants.

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 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 PhDs who currently work as university professors in software engineering. The specialists applied a predefined set of 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. We calculated Cohen's Kappa to assess the inter-rater reliability between the specialists. The Cohen's Kappa score of 0.295 indicates a 'fair' level of agreement. We do not apply any scorer calibration. So, the group decided to use the mean of the evaluations. This collaborative validation process aimed to reduce subjective bias and ensure alignment with the study's goals. The two PhD professors evaluated each response by assigning a score to each task completed by each participant in their respective groups, and we used the Wilcoxon signed-rank test to test our hypothesis (see Section 3.2).

To investigate the perceived benefits and challenges associated with the use of LLMs, we applied a Thematic Analysis [12] approach to analyze 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 analysis served as the primary method for addressing RQ3 and RQ4 (see Section 3.1). We began by familiarizing ourselves with the dataset, with each researcher reviewing the responses to gain an overall understanding. The dataset was then shared among the research team, followed by collaborative meetings to construct and refine a shared codebook. Each researcher identified initial codes independently, which were then reviewed, merged, and reorganized through two rounds of analysis: the first to elaborate the codes and the second to consolidate them. In these rounds, at least three researchers participated, leading to the development and refinement of codes. In the final phase, related codes were grouped into higher-level thematic categories that represent the main domains of perceived benefits and challenges associated with the use of LLMs in software engineering tasks. 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. Both our analysis code and our documentation are available in our replication package [25].

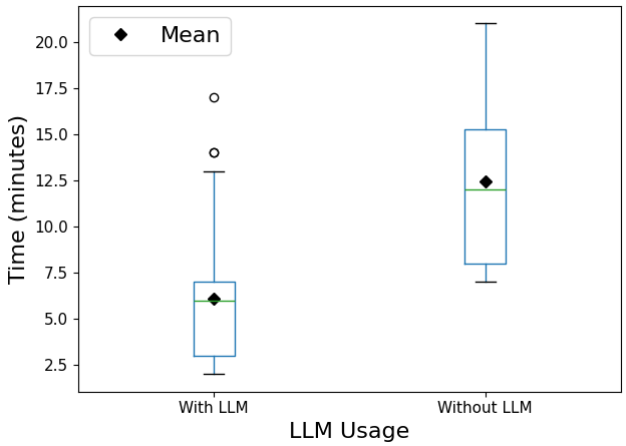

Figure 4: Task completion time with and without LLM.

4 Results

This section presents the results obtained from the experimental study in relation to our four research questions.

4.1 Task Time (RQ1)

Figure 4 shows that the task completion times for the condition without LLM support exhibit a higher median and greater inter-quartile spread. This observation shows that participants took longer to complete the tasks without LLM support. The results for the condition with LLM support indicate more consistent and faster task completion times. These results are supported by a Wilcoxon test ($p < 0.05$), suggesting a time efficiency gain associated with the use of LLMs in the documentation of software requirements using templated user stories and use cases. Our findings aligning with findings reported in the literature that highlight the potential of LLMs to enhance productivity in SE tasks [47].

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.

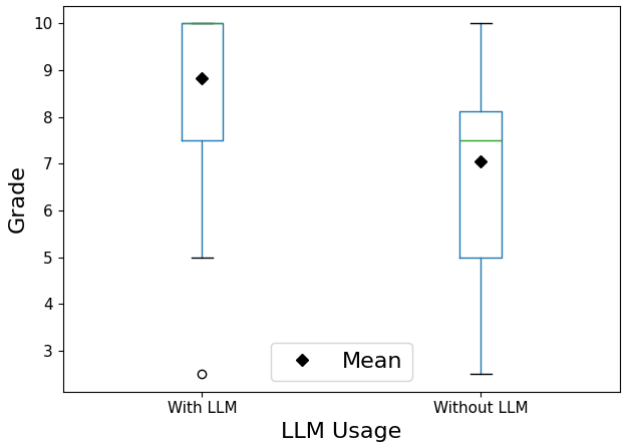

Figure 5: Mean of artifact quality scores between conditions with and without LLM.

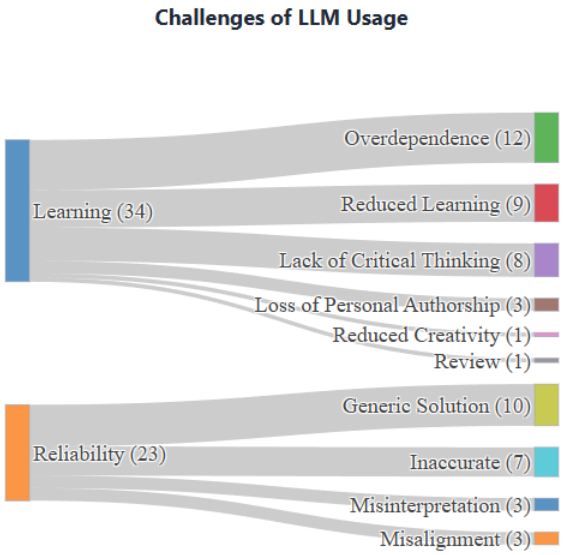

Figure 6: Perceived challenges of using LLMs in user story documentation.

4.2 Artifact Quality (RQ2)

Figure 5 presents the distribution of students' performance in each group, based on the task grade. 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 measures (mean and 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.

RQ2 Summary – Artifact Quality

Artifacts produced with LLM support received higher scores (mean = 8.8) compared to those created without LLM support (mean = 7.0) when evaluated by SE specialists. The Wilcoxon signed-rank test revealed a statistically significant difference in artifact scores between conditions ($p < 0.05$).

4.3 Benefits and Challenges of LLMs (RQ3)

Figure 6 presents a Sankey diagram with ten categories of challenges, which we organized based on our thematic analysis described in Section 3.6 with two top challenge categories: (i) Learning and (ii) Reliability. The number after the category code represents the count of participants who mentioned that category. The most commonly reported issue was *Overdependence* (12 mentions). Closely related to this, participants frequently noted *Generic Solutions or Inaccurate Responses* (10 and 7 mentions, respectively). Another critical concern was the *Reduced Learning* and *The Lack of Critical Thinking* (9 and 8 mentions, respectively). Some responses also pointed out *Authorship* concerns (3 mentions). Similarly, participants raised concerns about *Misinterpretation or Misalignment* with context (6 mentions).

Most reported challenges were associated with the Learning top category. Participants referred to *Overdependence* and *Reduced Learning* in descriptions of relying on LLM-generated outputs during task execution, such as reporting *"excessive dependence on an external tool"*, *"reduced learning because individuals tend to rely on the use of LLMs and consequently think less about the activity"*, and cases in which *"not making enough effort to solve problems limited independent reasoning"*. Representative participants' quotes are available in the replication package [25]. Mentions of Lack of Critical Thinking appeared in responses describing limited verification, reduced cross-checking, or uncritical acceptance of generated content, including statements that participants took the LLM's response as true without even reading it or searching for other information sources, or noted a *"lack of correction and validation of requirements"*. Some participants also reported concerns related to *Loss of Personal Authorship*, indicating a reduced presence of their own ideas or reasoning in the produced artifacts, for example, by stating that *"sticking to an idea that is not your own"* made it difficult to maintain a coherent line of reasoning, or mentioning a *'lack of personalization in thinking."* Reduced Creativity was mentioned in fewer responses, with participants reporting that reliance on generated suggestions resulted in *"reduced creativity"* or impaired originality. Challenges related to the Reliability dimension were primarily linked to *Generic Solutions* and *Inaccurate Responses*. Participants reported that some outputs were superficial, overly generalized, or incorrect despite appearing plausible, describing cases of *"generic or superficial responses," "generic use cases,"* or content that *"seems correct but contains inaccuracies,"* requiring attention from the student. In some responses, participants noted the need for manual review

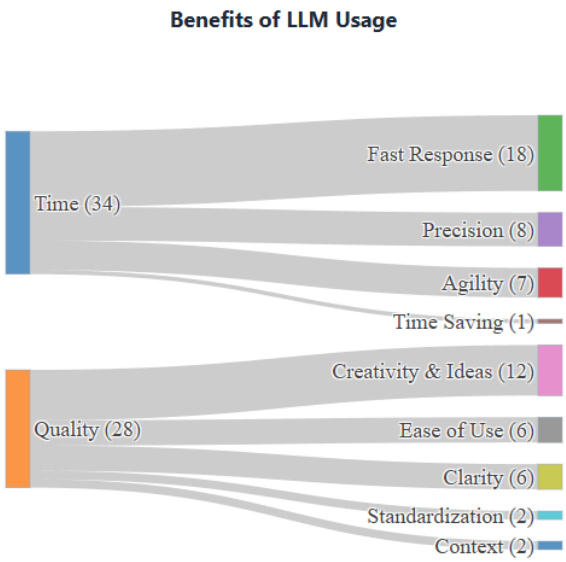

Figure 7: Perceived benefits of using LLMs in user story documentation.

and correction of generated artifacts, emphasizing that responses were *"not always correct"* and therefore required additional review to ensure accuracy and suitability.

Additional reliability-related challenges included *Misinterpretation* and *Misalignment*, reported in situations where the model did not fully capture the intended request, business rules, or contextual constraints, such as when *"the AI may misinterpret the requests,"* or when the model did not understand specific constraints defined by the instructor. Some participants explicitly mentioned instances in which the model produced incorrect information due to misunderstandings of prompts or the generation of hallucinated content, noting that the model *"often hallucinates"* and may deliver incorrect answers if not carefully validated.

Figure 7 presents a Sankey diagram classification of the perceived benefits reported by students when using Large Language Models (LLMs). We grouped the responses into two main categories, time and quality, reflecting the nature of the benefits participants experienced during the tasks. Overall, perceptions related to Time efficiency were the most prominent. In particular, *Fast Response* emerged as the most frequently cited benefit (18 mentions), highlighting the role of LLMs in providing immediate feedback and accelerating task completion. Closely related aspects such as *Precision* (8 mentions) and *Agility* (7 mentions) further reinforce students' perception that LLMs contribute to more efficient workflows, reducing delays in requirements-related activities. Regarding the Quality dimension, participants most frequently emphasized *Creativity and Idea Generation* (12 mentions). Additional benefits included *Ease of Use* and *Clarity* (6 mentions each). Less frequently, participants highlighted *Standardization* and improved *Contextual Understanding* (2 mentions each).

Most positive perceptions were related to Time. Participants repeatedly mentioned speed, efficiency, and agility during task execution. Several responses explicitly highlighted faster completion of activities, such as "speed and efficiency," "speed and practicality," and "speeding up the work process." Other participants referred to time savings in repetitive tasks and drafting activities, including comments such as "time savings in repetitive tasks" and "agility in drafting descriptions and use cases." Some responses also emphasized the immediacy of interaction, for example, stating that the tool "quickly generates responses" and allows "small adjustments if the result does not meet expectations."

Benefits associated with Quality were also frequently reported. Several responses highlighted clarity and understanding, including comments about "clear and detailed explanations," "ease of understanding what is being explained," and the ability to "clarify several questions mayt may arise during task execution. Moreover, ease of use and practicality were also mentioned, often in combination with speed, for example, "ease of use, time savings, and practicality."

Mentions related to Standardization appeared in responses describing the use of templates and consistent structures. Participants reported that the LLM helped maintain uniformity across multiple artifacts, such as "generating descriptions with the same standard for multiple user stories and use cases" and "support in organizing ideas based on templates." Some participants also noted that predefined structures reduced rework and improved consistency across documents. Finally, a smaller number of responses referred to Context and adaptability. Participants mentioned that the model provided suggestions based on the provided context and helped identify scenarios and usage situations, for example, "clear explanations and suggestions based on the provided context" and "ease in obtaining ideas for possible scenarios and usage contexts." Some responses also referred to customization of the detail level and improved communication through clearer documentation.

RQ3 Summary – Benefits and Challenges Perceived

The primary benefit is the increased agility in completing tasks. However, some major challenges identified with using 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 Interactions between students and LLM (RQ4)

To answer RQ4, we performed a categorization of user–model interaction turns across all collected dialogues. Each turn was coded according to its functional role in the interaction (e.g., prompt formulation, context provision, template definition, solution generation, refinement, and reinforcement), enabling a structured representation of interaction behaviors as explained in Section 3.6.

Table 2 provides the definitions of the most common identified codes, and Figure 8 summarizes the frequency of these codes, separated by actor (user vs. model). The frequency analysis reveals a clear division of roles between users and the LLM during interaction. For the model, the three most frequent categories are *Model Offers Reinforcements* (36), *Model Adequates Solution* (31), and *Model Confirms Instructions* (29). This pattern indicates that the LLM is

Table 2: Definitions of the most frequent interaction codes per persona

Persona	Code	Definition
User	Defines Template	The user specifies an explicit structure, format, or template.
	Provides Context	The user supplies domain-specific information.
	States Prompt	The user presents the question statement.
Model	Offers Reinforcements	The model offers more concepts or examples.
	Aligns Solution	The model outputs a hopeful solution.
	Confirms Instructions	The model explicitly confirms its understanding.

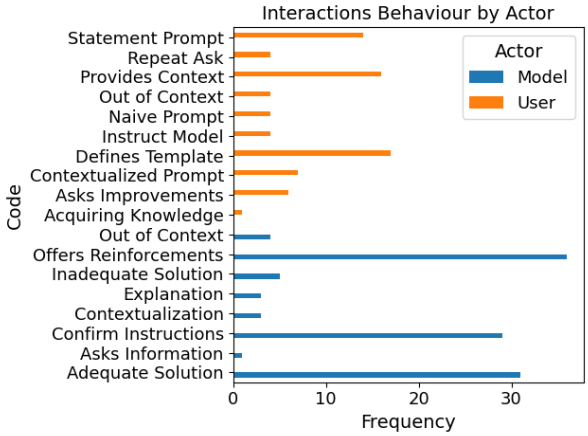

Figure 8: Open coding result.

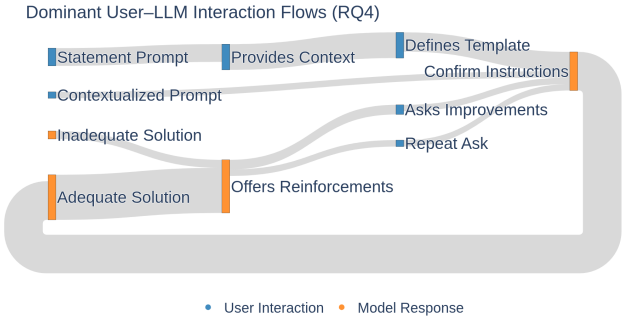

Figure 9: Dominant user–model interaction flows derived from the code of interaction turns.

primarily used to produce solutions, while also validating its understanding and extending or reinforcing outputs, suggesting a supportive and responsive role rather than proactive task initiation. In contrast, the user's most frequent categories are *User Defines Template* (17), *User Provides Context* (16), and *User Statement Prompt* (14). These behaviors reflect that students consistently engage in structuring the problem space, specifying constraints, and formulating task requirements before relying on the LLM's assistance.

Figure 9 presents a Sankey diagram that represents the dominant interaction flows between users and the model, derived from the categorization step of the thematic analysis. The diagram highlights recurrent sequential patterns of interaction codes, making visible how students and the model collaboratively construct task solutions over turns. The most prominent flow begins with user-driven actions, *Statement Prompt* followed by *Provides Context* and *Defines Template*. This sequence indicates that users actively shape the problem space by clarifying requirements, supplying domain information, and specifying structural constraints. Such behavior reinforces the user's role as the primary agent responsible for framing the task.

Following these preparatory actions, the interaction commonly transitions to the LLM *Confirms Instructions*, reflecting the model's tendency to explicitly validate its understanding of user intent before proceeding. This confirmation phase frequently leads to *Aligns Solution*, which represents the model's main problem-solving contribution. The thickness of this flow suggests that successful task progression often relies on this confirmation–solution pairing. Another dominant pathway extends from *Aligns Solution* to *Offers Reinforcements*, indicating that the LLM not only provides an initial answer but also elaborates on it with additional explanations, refinements, or examples. Subsequent user actions, such as *Asks Improvements* or *Repeat Ask*, feed back into this reinforcement loop, illustrating an iterative refinement process rather than a single-turn exchange.

Less frequent but visible transitions toward *Inadequate Solution* followed by *Offers Reinforcements* highlight recovery strategies, where the model compensates for partial or misaligned outputs through clarification or expansion. Overall, the Sankey diagram reveals a support-oriented interaction pattern, in which users initiate and structure tasks, while the LLM primarily responds by validating understanding, generating solutions, and iteratively reinforcing outputs. This sequential organization provides concrete evidence of how students employ LLMs as assistive tools during task execution, directly addressing RQ4.

RQ4 Summary – User–Model Interaction Patterns

Users consistently assume an active role in structuring the problem space by formulating prompts, providing contextual information, and defining templates. In response, the LLM predominantly validates user intent, generates solutions, and extends outputs through iterative reinforcement.

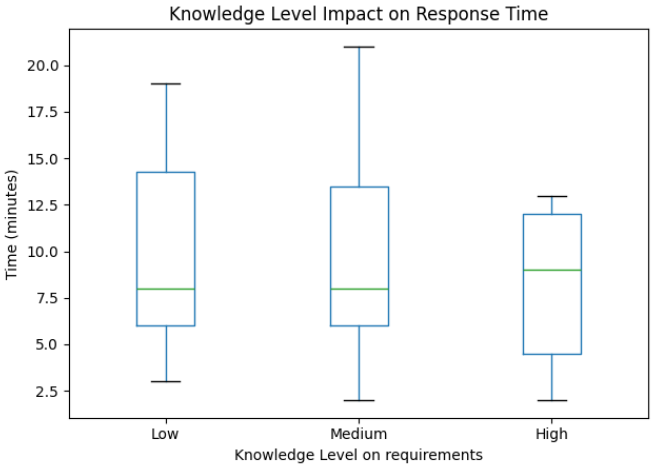

Figure 10: Task completion time with LLM by knowledge of requirements.

5 Discussion

The students' feedback (RQ3) 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 [42] that indicates that LLMs facilitate repetitive tasks and initial text formulation, thereby freeing students to focus on higher-order cognitive activities. On the other hand, the learning concern aligns with broader discussions in the literature on how automation may negatively affect active learning processes [11, 33, 40].

Figure 10 illustrates that participants with advanced requirements engineering knowledge dedicated more time to the task compared to students with limited understanding of this topic. This trend also appears with other software engineering knowledge areas, such as LLM use and Agile Software Development. This observation 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.

The dominant interaction flow progressing from *Statement Prompt*, *Provides Context*, and *Defines Template* to *Confirm Instructions* and *Adequate Solution*, indicate that effective task execution relies on explicit user guidance and alignment checks prior to solution generation. This pattern is consistent with prior work showing that LLMs can generate user stories comparable to human-authored artifacts when provided with clear structure, contextual grounding, and explicit quality criteria, particularly to mitigate limitations in diversity, creativity, and compliance with acceptance standards [41]. Complementarily, research on LLM-generated user stories for AI systems demonstrates that such guidance enables models to produce stakeholder-oriented artifacts and identify non-functional and ethical requirements during early elicitation phases, reinforcing the role of LLMs as assistive rather than autonomous requirements engineers [54]. Finally, this interaction dynamic aligns with earlier tool-based approaches such as User Story Tutor, which emphasize automated feedback and guidance to support readability, quality, and continuous learning in user story construction, highlighting that structured, feedback-driven assistance has long been effective in supporting requirements authorship, independent of generative capabilities [20].

5.1 Contributions for Teachers, Students, and the SE Community

Our results show that while LLMs significantly reduce task completion time (from 12 to 6 minutes, $p < 0.05$), this comes at a learning cost. Overdependence (most-cited challenge), reduced learning (second-most-cited), and lack of critical thinking (third-most-cited) were the most reported concerns. These findings suggest that: (i) LLM-assisted tasks should not replace independent foundational exercises; (ii) the quality gain observed (mean 8.8 vs. 7.0) must be interpreted cautiously, as higher scores do not necessarily reflect deeper understanding; and (iii) assessments should evaluate the reasoning process behind requirements, not just the final artifact.

Given these implications, we offer concrete advice for professors. We recommend that educators: (i) introduce LLMs only after students complete at least one independent requirements task, ensuring they can critically evaluate AI-generated content; (ii) treat prompt engineering as an explicit learning objective, since our RQ4 analysis shows that students who provided richer context and templates obtained better results; (iii) require mandatory critique and revision of LLM outputs, given students' tendency to accept responses uncritically; and (iv) use generic or incorrect LLM-generated user stories as classroom examples for discussing requirements quality criteria.

Beyond requirements engineering education, our contribution extends to the broader SE education community in three ways: (i) methodologically, our controlled Latin Square design combining quantitative and qualitative measures is transferable to other SE education contexts; (ii) our RQ4 interaction coding schema (Table 2) provides a reusable framework for studying LLM use in other SE tasks such as code review or testing; and (iii) our findings on overdependence and reduced critical thinking complement broader SE education concerns raised in the literature [12, 35], adding task-specific empirical evidence. We will revise the Discussion and Related Work sections to make these implications and our positioning explicit.

6 Threats to Validity

This exploratory study on the use of LLMs in user story creation with students 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. [52].

Construct Validity. This kind of threat can occur in formulating the questionnaire with the set of tasks and questions for each group to answer with or/and without an 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 this 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. 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 study focused on students who were already trained on the topic of user story creation/requirements elicitation. 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 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.

7 Related Work

LLM have been used to provide solutions to traditional software engineering problems in several areas, such as development, testing, design, and maintenance [26].

Regarding software requirements, there is a significant number of works that address this topic. For instance, Marques et al. [36] and Vasudevan and Reddivari [51] 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. [44] carried out an experiment involving the use of LLM in requirements engineering education. In their study [44], 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 [22] 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. [53] 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. [39] 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 [15, 23, 43, 49, 50]. 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 [22, 39, 44, 53], 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.

8 Conclusion

Our study provides empirical evidence that LLMs accelerate requirements documentation but introduce significant learning costs, including overdependence, reduced critical thinking, and loss of personal authorship. While LLMs are best positioned as assistive tools — complementing rather than replacing human judgment — their adoption in educational settings must be accompanied by pedagogical strategies that emphasize prompt formulation, critical validation, and reflective use to preserve student autonomy and intellectual development. Future work should extend this investigation through larger and more diverse samples, longitudinal designs, and experiments with other LLMs to assess generalizability.

Artifact Availability

The artifacts (including materials and datasets) supporting this work are publicly available in the replication package [25].

Acknowledgments

This research was partially supported by the Brazilian funding agencies: CAPES, CNPq (Grants 406089/2025-6 and 305171/2025-9), and FAPEMIG (Grant APQ-01488-24).

References

[1] 2008. *Latin Square Designs.* Springer New York, New York, NY, 297–297. https://doi.org/10.1007/978-0-387-32833-1_223

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

[3] Gabriel Amaral, Henrique Gomes, Eduardo Figueiredo, Carla Bezerra, and Larissa Rocha. 2025. Improving JavaScript Test Quality with Large Language Models: Lessons from Test Smell Refactoring. In *Brazilian Symposium on Software Engineering (SBES)*. SBC, 776–782.

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

[5] 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

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

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

[8] 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.

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

[10] Zixin Chen, Jiachen Wang, Meng Xia, Kento Shigyo, Dingdong Liu, Rong Zhang, and Huamin Qu. 2024. StuGPTViz: A visual analytics approach to understand student-ChatGPT interactions. *IEEE Transactions on Visualization and Computer Graphics* (2024).

[11] 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.

[12] Victoria Clarke and Virginia Braun. 2017. Thematic analysis. *The Journal of Positive Psychology* 12, 3 (2017), 297–298. https://doi.org/10.1080/17439760.2016.1262613 arXiv:https://doi.org/10.1080/17439760.2016.1262613

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

[14] Kattiana Constantino and Eduardo Figueiredo. 2022. CoopFinder: Finding Collaborators Based on Co–Changed Files. In *2022 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 1–3. https://doi.org/10.1109/VL/HCC53370.2022.9833126

[15] Kattiana Constantino, Pedro Garcia, and Eduardo Figueiredo. 2025. A Long-Term Study of the Pandemic Impact on Education: A Software Engineering Case. In *Proceedings of the 17th International Conference on Computer Supported Education (CSEDU)*. Porto, Portugal.

[16] Kattiana Constantino, Raquel Prates, and Eduardo Figueiredo. 2023. Recommending collaborators based on co–changed files: A controlled experiment. In *Simpósio Brasileiro de Sistemas Colaborativos (SBSC)*. SBC, 154–168.

[17] Kattiana Constantino, Raquel Prates, and Eduardo Figueiredo. 2024. A user evaluation of a collaborator recommender based on co-changed files. *Journal on Interactive Systems* 15, 1 (2024), 157–169.

[18] Kattiana Constantino, Mauricio Souza, Shurui Zhou, Eduardo Figueiredo, and Christian Kastner. 2021. Perceptions of Open-Source Software Developers on Collaborations: An Interview and Survey Study. *Journal of Software: Evolution and Process (JSEP)* 35, 5 (2021).

[19] Kattiana Constantino, Shurui Zhou, Mauricio Souza, Eduardo Figueiredo, and Christian Kästner. 2020. Understanding collaborative software development: An interview study. In *Proceedings of the 15th international conference on global software engineering*. 55–65.

[20] Giseldo da Silva Neo, José Antão Beltrão Moura, Hyggo Oliveira de Almeida, Alana Viana Borges da Silva Neo, and Olival de Gusmão Freitas Júnior. 2024. User Story Tutor (UST) to Support Agile Software Developers. *ArXiv* abs/2406.16259 (2024). https://api.semanticscholar.org/CorpusID:269680474

[21] 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

[22] 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

[23] Pedro Garcia, Jose Ferreira, Matheus Goncalves, Tiago Carneiro, Eduardo Figueiredo, and Igor Pereira. 2024. Current DevOps Teaching Techniques: A Systematic Literature Review. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES), Education Track*. Curitiba, Brazil.

[24] Pedro Garcia, José Ferreira, Matheus Gonçalves, Tiago Carneiro, Eduardo Figueiredo, and Igor Pereira. 2024. Current DevOps Teaching Techniques: A Systematic Literature Review. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 389–398. https://doi.org/10.5753/sbes.2024.3503

[25] Pedro S. C. Garcia. 2026. Replication package for "A Large Language Model as a Support Tool for User Story Creation with Use Cases: An Empirical Study". https://github.com/LabSoft-UFMG-Digitall-Cell/2026-LLMReq.

[26] 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

[27] 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.

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

[29] 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).

[30] 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.

[31] 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.

[32] 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

[33] 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.

[34] Jefferson Lopes, Johnatan Oliveira, and Eduardo Figueiredo. 2025. Aged to Perfection? Analyzing the Impact of Years of Experience on Code Quality. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES)*. Recife, Brazil.

[35] 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

[36] 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

[37] 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

[38] 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)*.

[39] 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

[40] 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.

[41] Giovanni Quattrocchi, Liliana Pasquale, Paola Spoletini, and Luciano Baresi. 2025. Can LLMs Generate User Stories and Assess Their Quality? arXiv:2507.15157 [cs.SE] https://arxiv.org/abs/2507.15157

[42] 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.

[43] Yasmim Rosa, Pedro Garcia, Kattiana Constantino, and Eduardo Figueiredo. 2025. Reflexões sobre o Uso de LLMs no Ensino de Programação. In *Anais do V Simpósio Brasileiro de Educação em Computação* (Juiz de Fora/MG). SBC, Porto Alegre, RS, Brasil, 741–749. https://doi.org/10.5753/educomp.2025.5366

[44] 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)*. 624–634. https://doi.org/10.1145/3701625.3701687

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

[46] 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.

[47] 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.

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

[49] Mauricio Souza, Renata Moreira, and Eduardo Figueiredo. 2019. Students' Perception on the Use of Project-Based Learning in Software Engineering Education. In *Proceedings of the Brazilian Symposium on Software Engineering (SBES), Education Track*. Salvador, Brazil.

[50] Mauricio Souza, Lucas Veado, Renata Moreira, Eduardo Figueiredo, and Heitor Costa. 2018. A Systematic Mapping Study on Game-related Methods for Software Engineering Education. *Information and Software Technology* 85 (2018), 201–218.

[51] 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

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

[53] 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

[54] Asma Yamani, Malak Baslyman, and Moataz Ahmed. 2025. Leveraging LLMs for User Stories in AI Systems: UStAI Dataset. In *Proceedings of the 21st International Conference on Predictive Models and Data Analytics in Software Engineering* (Trondheim, Norway) *(PROMISE '25)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/3727582.3728689