Evaluating Generative Artificial Intelligence for Teaching User Stories

Claudia Marcos[1], Guillermo Rodriguez[2,3], and Delfina Garrido[3]

[1] ISISTAN (UNICEN-CIC), Tandil, Argentina
cmarcos@exa.unicen.edu.ar
[2] Instituto de Tecnología (INTEC),
Universidad Argentina de la Empresa (UADE),
Buenos Aires, Argentina
guirodriguez@uade.edu.ar
[3] UNCPBA, Tandil, Argentina
dvillarrealgarrido@alumnos.exa.unicen.edu.ar

Abstract. Generative Artificial Intelligence (GAI) has demonstrated significant potential as a pedagogical assistant across various disciplines. In Software Engineering, User Stories (US) constitute a fundamental technique for requirement specification from the user's perspective, relying on the "Three Cs" framework (Card, Conversation, and Confirmation) to ensure functional and non-functional understanding. Nevertheless, teaching this technique in massive university environments poses a significant challenge due to the requirement for continuous feedback. This paper presents a study structured into four experiments with incremental context levels to evaluate GAI's capability to generate US aligned with specific pedagogical objectives. The methodology ranged from the exclusive provision of problem statements to the inclusion of detailed theoretical material and faculty-resolved examples. The results, evaluated through direct comparison with official solutions and compliance with the INVEST criteria, demonstrate that model precision significantly improves as domain-specific information is provided. The scenario integrating both theory and practice achieved the highest performance. However, persistent limitations were detected regarding the models' ability to adjust granularity and establish coherent relationships between stories. It is concluded that GAI is a valuable tool for fostering autonomous learning, although its use does not replace the supervision of a specialized instructor.

Keywords: Generative Artificial Intelligence · User Stories · Software Engineering Education · INVEST criteria · Pedagogical Assistance.

1 Introduction

Requirements Engineering (RE) plays a critical role in the success of software systems, as it defines the functional and non-functional expectations that guide the entire development process. Within agile development methodologies, User

Proceedings of the 29th Workshop on Requirements Engineering (WER2026),
August 11-13, 2026, La Plata, Argentina.
https://doi.org/10.29327/1884517.29-7

Stories (US) [2] have become one of the most widely used techniques for representing requirements in a lightweight and user-centered manner. User stories allow development teams to capture stakeholder needs through simple natural language statements that describe who requires a feature, what functionality is needed, and why the functionality provides value.

Despite their apparent simplicity, writing effective user stories requires a set of analytical skills that novice practitioners often find difficult to develop. Students must learn how to interpret informal problem descriptions, identify relevant actors, infer system goals, and articulate the value delivered by the functionality. These tasks require both conceptual understanding and practice with continuous feedback. However, in university environments with large class sizes, providing individualized feedback for each student exercise becomes challenging for instructors.

Recent advances in Generative Artificial Intelligence (GAI), particularly systems based on Large Language Models (LLMs), offer new opportunities to support learning activities in software engineering education [5,3,7]. These models are capable of processing natural language descriptions and generating structured textual outputs, making them particularly suitable for tasks related to requirements interpretation and documentation. In educational contexts, GAI systems can function as interactive assistants capable of generating explanations, examples, and alternative formulations that help students understand complex concepts. In the context of Requirements Engineering education, generative AI systems could assist students in transforming informal textual descriptions into structured requirement artifacts such as user stories [10,11].

For instance, when provided with a problem statement, an LLM may identify potential actors, infer goals, and propose candidate user stories. This capability could support students in understanding how textual descriptions can be interpreted and translated into agile requirement specifications. However, an important open question remains: how much contextual information is required for generative AI systems to produce pedagogically useful user stories? While LLMs can generate plausible outputs with minimal input, their responses may vary significantly depending on the context and guidance provided in the prompt. In particular, it is unclear how the inclusion of theoretical material, examples, and domain explanations influences the quality of generated user stories.

This work investigates the effectiveness of ChatGPT, a generative AI, for supporting the generation of user stories in a software engineering educational context. The study evaluates how different levels of contextual information influence the quality of the generated stories. To achieve this objective, we designed a series of controlled experiments in which a generative AI model receives progressively richer input information, ranging from simple problem statements to detailed theoretical explanations and solved examples provided by the course instructors.

The contributions of this paper are the following:

– An experimental evaluation of generative AI, ChatGPT, for generating user stories from textual problem descriptions.

– A comparative analysis of different levels of contextual information provided to the model.
– An assessment of the generated user stories using the INVEST quality criteria and comparison with instructor-defined solutions [4].
– Insights into the potential role of generative AI as a pedagogical assistant in requirements engineering education.

The remainder of this paper is organized as follows. Section 2 presents the background. Section 3 compares related work on generative AI in education and requirements engineering. Section 4 describes the experimental methodology used in this study. Section 5 presents the results and analysis of the experiments. Section 6 discusses the implications for software engineering education. Section 7 presents the threats to validity. Finally, Section 8 concludes the paper and outlines future research directions.

2 Generative Artificial Intelligence in Education and its Potential for User Story Identification

Generative Artificial Intelligence (GAI) serves as a powerful educational assistant for structuring User Stories, particularly when supported by formal theory (INVEST) and expert examples.

2.1 Generative Artificial Intelligence in Education

Generative Artificial Intelligence, particularly systems based on large language models (LLMs), has recently attracted significant attention in the educational domain. These models are capable of generating natural language explanations, examples, and structured information in response to textual prompts, enabling new forms of interaction between learners and intelligent systems. In educational settings, generative AI can function as an interactive learning assistant that supports activities such as concept explanation, problem solving, and formative feedback.

Unlike traditional rule-based educational technologies, LLMs can generate context-aware responses and maintain conversational interactions, which allows students to iteratively explore complex concepts. Recent studies highlight that these systems can support personalized learning processes and enhance knowledge exploration through dialogue-based interaction [6]. In software engineering education, generative AI is particularly relevant for tasks involving the interpretation and transformation of natural language descriptions. Requirements engineering frequently involves analyzing informal textual information provided by stakeholders and translating it into structured representations that guide system development. Due to their strong natural language processing capabilities, generative models can assist students in analyzing textual information and identifying relevant elements that may represent system requirements [1].

2.2 User Stories in Agile Requirements Engineering

User Stories (US) are a commonly used technique for representing requirements in agile software development [8]. They provide a lightweight and user-centered approach for describing system functionality by focusing on the needs and goals of end users. User stories typically follow the template: As a <type of user>, I want <some goal> so that <some benefit>. This structure explicitly represents three key elements: the actor, the goal or functionality, and the value that the functionality provides. By emphasizing the value delivered to users, user stories encourage development teams to focus on stakeholder needs rather than solely on technical implementation [2].

In practice, user stories are usually complemented by acceptance criteria, which define the conditions that must be satisfied for the story to be considered complete. These criteria help clarify the requirement, guide testing activities, and ensure a shared understanding among development team members. Although the user story format is simple, identifying and writing effective user stories can be challenging for novice practitioners. Students often experience difficulties in identifying the appropriate actors, defining clear goals, and articulating the expected value of the functionality, particularly when requirements originate from unstructured textual descriptions.

2.3 Supporting User Story Identification with Generative AI

Generative AI offers promising opportunities to support the learning of requirements engineering practices, particularly the identification and formulation of user stories. Because large language models can analyze and generate natural language, they can assist students in identifying potential user needs embedded within textual descriptions. For example, generative models can process domain narratives or problem descriptions and propose candidate user stories by identifying actors, goals, and potential benefits. Such functionality can help students understand how informal textual information can be transformed into structured requirements artifacts.

From an educational perspective, generative AI can support both learning and practice. Students can use these systems to explore alternative formulations of user stories, receive feedback on the structure of their own stories, and analyze how different interpretations of the same textual input may lead to different requirements. This process can help develop analytical skills that are fundamental for requirements engineering. However, generative AI should be regarded as a supportive tool rather than a replacement for human analysis in requirements engineering. Although large language models are capable of processing and generating natural language, their outputs may contain inaccuracies, misinterpretations or hallucinations, which means they cannot be fully relied upon for producing correct requirements artifacts. When appropriately guided and contextualized, however, generative AI can be used as an educational assistant that helps explain concepts and illustrate how textual descriptions can be transformed into requirements.

In the case of user stories, this support is particularly valuable because they do not have a single canonical formulation: the same requirement can be expressed in different ways while still being valid, as long as the actor, goal, and value are clearly represented. Consequently, generative AI can assist learners by suggesting possible formulations and explanations, while the interpretation and validation of the resulting user stories remain a human responsibility [8].

3 Related Work

Recent research has explored the use of Natural Language Processing (NLP) and Artificial Intelligence (AI) techniques to support activities in Requirements Engineering, particularly the automatic extraction of requirements or user stories from unstructured textual sources. These approaches aim to reduce the manual effort required from requirements engineers and facilitate the identification of user needs from large volumes of textual data.

One of the most relevant contributions in this area is presented by Siahaan et al. [13], who propose a method for automatically extracting user stories from natural language texts, particularly online news articles. Their approach relies on several NLP techniques, including part-of-speech tagging, named entity recognition, and dependency parsing, to identify the classical components of a user story: actor, action, and goal. The results indicate that relevant software requirements can be derived from non-technical textual sources, demonstrating the potential of leveraging publicly available textual data for requirements elicitation.

Similarly, Ngaliah et al. [9] present a machine learning-based approach for extracting user stories from online news articles. Their method employs a Maximum Entropy classifier combined with linguistic feature extraction to identify actors and actions within textual content. The experimental results report high precision and recall values for actor identification, suggesting that supervised learning techniques can effectively support the transformation of narrative text into structured software requirements.

More recent studies have explored the application of large language models (LLMs) and advanced AI techniques to automate additional aspects of the requirements engineering process. In this context, Voria et al. [14] introduce RECOVER, a system designed to generate software requirements from stakeholder conversations recorded during elicitation meetings. The system analyzes dialogue transcripts, identifies relevant requirement fragments, and generates structured requirement descriptions. Their evaluation shows that the generated requirements achieve acceptable levels of correctness and completeness, suggesting that automated analysis of stakeholder conversations can complement the work of requirements engineers.

Finally, Seibert [12] proposes a dialogue-based approach to computer-aided requirements elicitation. Rather than fully automating the extraction of requirements, this work advocates for interactive systems that assist analysts through conversational interfaces. In this model, AI systems collaborate with stakeholders during elicitation sessions, helping to identify, refine, and structure requirements

iteratively. This perspective highlights the importance of maintaining human involvement in the requirements engineering process while leveraging AI capabilities for text analysis and knowledge extraction.

Overall, these studies illustrate the evolution of research in automated requirements extraction, moving from rule-based NLP techniques and traditional machine learning models to approaches based on large language models and conversational AI. Despite these advances, several challenges remain, such as ensuring semantic accuracy and effectively integrating these techniques into agile development processes. Additionally, it would be valuable to investigate how much domain and pedagogical information generative AI systems require to learn the teaching strategies of a specific course. This could enable generative AI to function as an educational assistant that supports students and complements the instructor's role in the learning process.

4 Experimental Design

To evaluate the ability of generative AI to generate pedagogically useful user stories, we designed a series of controlled experiments that progressively increase the contextual information provided to the model. We used ChatGPT as a generative AI tool. The objective of these experiments is to analyze how different levels of instructional guidance affect the quality of the generated user stories.

The experiments simulate scenarios that reflect how students typically interact with course materials in a requirements engineering class. In particular, we analyze four situations that vary according to the amount of information available to the generative model.

The research goal of this experiment is to answer the following research questions (RQ):

- **RQ1.** How does the incorporation of solved exercises and theory influence the solution proposed by the AI?
- **RQ2.** How does the use of expert-resolved examples influence the AI's ability to propose additional requirements ("unidentified activities") that are not explicit in the original statement?
- **RQ3.** Is there a correlation between the amount of context and "hallucinations"?
- **RQ4.** What information is most frequently omitted in the Acceptance Criteria (AC)?

4.1 Experimental Setup

Four experiments were designed to evaluate how ChatGPT generates User Stories (US) and their respective Acceptance Criteria (AC). The research focused on identifying how the gradual addition of context (including domain statements, theoretical frameworks, and practical examples) influences the quality, structure, and logical coherence of the requirements produced by the model. This

was conducted across five varied domains used in the Software Development Methodologies course of the Systems Engineering degree at UNCPBA (Universidad Nacional del Centro de la Provincia de Buenos Aires). For the experiments, we used the set of problem statements provided by the course staff, which were originally used to design the practical assignment on requirements specification using user stories. The problem statements correspond to 5 domains: *AdoptaYa, Calamuchita, FitnessPro, MinisterioSalud, and SeguroFiel* [4].

A comparative manual analysis was performed to detect recurring error patterns and areas for improvement in the automation of requirements engineering. The methodology was structured into four incremental phases to observe the evolution of the model's performance:

- **Experiment 1: Isolated Statements (Baseline)**
  Inputs: Only the textual domain statements were provided to the AI.
  Objective: To evaluate the model's native ability to identify requirements and draft US and Acceptance Criteria (AC) without external guidance.
- **Experiment 2: Statements + Theoretical Framework**
  Inputs: Domain statements accompanied by the formal theory of User Stories, including the INVEST criteria, the "As a/I want/So that" format, and the "Three Cs" (Card, Conversation, Confirmation).
  Objective: To observe if knowledge of the methodological framework improved the hierarchy (epics), formatting, and relationships between the stories.
- **Experiment 3: Statements + Solved Examples**
  Inputs: Domain statements along with the course's Solved Examples file (cases such as "Seguro Fiel" and "Ministerio de Salud").
  Objective: To determine if the AI could mimic the style, level of detail, and technical rigor of a human expert by having direct references to "ideal solutions".
- **Experiment 4: Integral Configuration (Theory + Examples)**
  Inputs: The full context was provided: domain statements, US theory (INVEST), and the course's solved examples.
  Objective: To verify if the combination of theoretical and practical support allowed for achieving a professional standard, minimizing recurring errors such as the lack of security validations or functional redundancies.

4.2 Evaluation Criteria

The generated user stories were evaluated using two complementary approaches.

- 1. Comparison with Instructor Solutions.
  Each experiment used problem statements for which instructors had previously defined reference user stories. The generated stories were compared with these reference solutions to determine whether the AI identified similar actors, goals, and value propositions.

[4] https://shorturl.at/AbLeZ

– 2. INVEST Criteria.
The quality of the generated stories was also evaluated according to the INVEST criteria, which define desirable properties for agile user stories:
  - Independent: Stories should not strongly depend on other stories.
  - Negotiable: Stories should allow discussion and refinement.
  - Valuable: Stories must deliver clear value to users.
  - Estimable: Stories should be understandable enough to estimate.
  - Small: Stories should represent manageable units of work.
  - Testable: Stories should allow the definition of acceptance criteria.

Each generated story was manually reviewed to determine whether it satisfied these characteristics.

5 Results and Analysis

This section presents the experimental results and a comparative analysis to assess how providing incremental contextual information to the model influences the quality and structural coherence of the generated user stories. We show the results in terms of evaluation criteria: first, comparison with instructor solutions; second, INVEST criteria.

5.1 Comparison with instructor solutions

Across all experiments, ChatGPT demonstrates the ability to approximate the structure of the instructor-provided solutions, but consistently falls short in terms of logical coherence and technical precision (Table 1).

In Experiment 1, the generated user stories diverge significantly from the instructor solutions, showing fragmented structures, missing relationships, and incomplete acceptance criteria. The AI captures surface-level functionality but fails to reproduce the integrated system view present in the official solutions.

In Experiment 2, the inclusion of theory improves alignment with instructor solutions in terms of organization (e.g., use of epics and better grouping). However, discrepancies persist due to redundancy, incorrect role assignments, and especially the introduction of non-existent constraints, which are absent in the instructor versions.

In Experiment 3, the use of solved examples leads to outputs that closely resemble the instructor solutions in format and style. Nevertheless, this similarity is largely superficial: the AI still fails to replicate the underlying logic, including correct story relationships, completeness of acceptance criteria, and essential preconditions.

In Experiment 4, combining theory and examples produces the closest approximation to instructor solutions in terms of structure, hierarchy, and level of detail. Despite this, important gaps remain, such as redundant or over-specified stories, missing logical links, and critical omissions (e.g., authentication checks), which prevent full equivalence with the instructor's solutions.

Table 1. Experimental design for evaluating context configurations

Exp.	#Stmts	Theory	Examples	Objective
E1	5	No	No	Analyze model behavior without context
E2	5	Yes	No	Assess the impact of theory
E3	3	No	Yes	Analyze learning from examples
E4	3	Yes	Yes	Evaluate performance with maximum context

Overall, while each intervention (theory, examples, or both) incrementally improves the resemblance to instructor solutions, none fully achieves their level of coherence, correctness, and domain fidelity. This confirms that human expertise remains essential to validate and refine AI-generated requirements.

5.2 INVEST Criteria

The analysis reveals clear differences across experiments in terms of compliance with the INVEST criteria. Table 2 shows that, applying the INVEST framework significantly improves the structure and testability of the artifacts, particularly in Experiments 2 and 4. However, it also introduces a trade-off, as it may reduce negotiability by encouraging the model to invent or over-specify constraints that were not originally present.

Regarding Independence, Experiments 1 and 3 produced fragmented user stories with little or no logical linkage, resulting in apparent independence due to missing dependencies and poor traceability. In contrast, Experiments 2 and 4 introduced explicit relationships between stories, improving traceability and structure, but reducing strict independence by correctly capturing interdependencies.

For Negotiability, Experiment 1 showed high flexibility due to a lack of detail, though this stemmed from incomplete specifications rather than intentional design. In Experiments 2 and 4, negotiability decreased as the model tended to hallucinate constraints or business rules not present in the original requirements, thereby limiting flexibility.

In terms of Value, Experiments 1 and 2 remained at a standard level, while Experiments 3 and 4 demonstrated increased value by identifying implicit requirements and proposing additional functionality, supported by the inclusion of examples and richer context.

For Estimability and Size, Experiment 1 produced incomplete and difficult-to-estimate stories. Experiments 2 and 4 improved significantly by structuring requirements into epics and sprint-sized user stories. However, Experiment 4 introduced redundancy, which negatively affected accurate effort estimation.

Finally, regarding Testability, Experiment 1 lacked sufficient acceptance criteria, hindering test design. Experiment 3 improved descriptive detail, increasing apparent testability, although not always ensuring logical correctness. Experiment 4 achieved the highest level of detail, but still exhibited critical gaps, such as missing preconditions (e.g., authentication), which undermine the validity of acceptance testing.

Table 2. Comparison Across Experiments

Criterion	Exp. 1	Exp. 2	Exp. 3	Exp. 4
Independent	Fragmented	Manual links	Inconsistent	Explicit relationships
Negotiable	By omission	Low (rule invention)	Inconsistent	Low (over-specified)
Valuable	Standard	Standard	High (discovers US)	Maximum (proactive)
Estimable	Difficult	Sprint-sized	Improved	Redundant
Small	Isolated atoms	Epic structure	Mimetic	Hierarchical / Epics
Testable	Poor	Improved	Detailed	Exhaustive (login issues)

These findings suggest that generative AI systems can substantially benefit from the inclusion of pedagogical context, especially when supported by clear theoretical explanations and illustrative examples. At the same time, they highlight the need for careful prompt design to balance guidance with flexibility, avoiding unintended rigidity in the generated outputs.

6 Discussion

The findings of this study suggest that generative AI can function as an educational assistant for requirements engineering tasks. When provided with sufficient contextual information—ranging from formal theory like the INVEST criteria to expert-resolved examples—the model generates user stories that closely approximate instructor-defined solutions and respect common agile practices, such as the organization of backlogs into Epics.

However, the results also highlight that generative AI should not be considered a replacement for human analysis. While the generated stories are often structurally plausible and visually professional, they frequently contain inconsistencies, omit critical logical dependencies, or introduce additional functionality not explicitly required in the problem description. From an educational perspective, this limitation represents a significant opportunity: students can use generative AI outputs as a starting point for analysis and discussion, critically evaluating whether the proposed stories correctly reflect the intended requirements and identifying the logical "blind spots" that the AI fails to address.

Research demonstrates that the gradual incorporation of context, such as theoretical frameworks and expert examples (RQ1), significantly optimizes the hierarchical structure and professional formatting of user stories generated by AI. Nevertheless, this increase in information carries a risk of "over-specification" or "hallucination," where the model invents business constraints and acceptance criteria not present in the original problem statements. Despite providing methodological support, systemic "blind spots" in deep logic persist, such as the recurring omission of basic security validations.

Consequently, while AI acts as a valuable assistant for structuring initial requirements and proposing additional activities, expert human supervision is indispensable to ensure systemic coherence and correct fundamental logical errors. With respect to RQ2, it is possible to analyze how the use of expert-resolved examples serves as a catalyst for proactivity in the model. In Experiment 3, having direct references for style and depth led the AI to move beyond mere summaries and propose requirements not originally contemplated by the instructors. For instance, in the "Fitness Pro" domain, expert references prompted the AI to declare additional user stories that were absent in the official solution. Similarly, in Experiment 4, the model successfully identified functional activities that the teaching staff had not initially included in their baseline.

There is a clear trend indicating that as the amount of contextual information (theory and examples) increases, so does the risk of the AI "hallucinating" or inventing business rules (RQ3). While the initial experiments often suffered from data omissions due to limited context, the integral configuration in Experiment 4 showed a distinct trend toward over-specification. With increased support, the AI becomes more prone to deducing or inventing constraints and non-existent business rules that were neither present in the original text nor required by the domain expert.

Finally, with respect to RQ4, manual analysis reveals that the AI consistently struggles to detail the specific technical information and precise data validations required to close a functional flow. Even when provided with high-quality examples, the quality of the generated criteria remains inconsistent and often insufficient depending on the complexity of the domain. A notable and recurring systemic "blind spot" across all four experiments—even when theory and examples were provided—was the failure to verify essential security preconditions, such as ensuring a user is logged in before performing sensitive actions, a requirement that was explicitly described in statements like "Fitness Pro". Additionally, the AI frequently fails to account for necessary validations involving external systems or the mandatory nature of specific data fields required by the business logic

7 Threats to validity

As with any empirical study, this work is subject to several threats to validity. Following common empirical research guidelines in software engineering, we discuss threats related to construct validity, internal validity, external validity, and conclusion validity.

7.1 Construct validity

Construct validity concerns whether the measurements used in the study accurately capture the concepts being evaluated. In this work, the quality of the generated user stories was evaluated using two mechanisms: comparison with instructor-defined reference solutions and assessment according to the INVEST

criteria. Although the INVEST framework is widely used in agile development to evaluate user stories, it does not capture all aspects of requirement quality, such as domain completeness or stakeholder alignment. Therefore, the evaluation may not fully represent the overall usefulness of the generated requirements artifacts. Additionally, the comparison with instructor-defined solutions assumes that these solutions represent an appropriate or optimal interpretation of the problem statements. In practice, multiple valid sets of user stories may exist for the same problem.

7.2 Internal validity

Internal validity concerns factors that may influence the observed results independently of the variables being studied. In this research, the primary independent variable is the amount of contextual information provided to the generative AI model. One potential threat arises from the prompt design used in each experiment. Small variations in prompt wording may influence the responses produced by large language models. Although efforts were made to maintain consistent prompt structures across experiments, it is possible that some results are partially influenced by prompt phrasing rather than by contextual information alone. Another potential threat relates to the manual evaluation of the generated user stories. The qualitative assessment of INVEST properties involves a degree of subjective judgment, which may introduce evaluator bias.

7.3 External validity

External validity refers to the extent to which the results can be generalized beyond the context of the study. The experiments were conducted using a limited set of problem descriptions typically used in undergraduate software engineering courses. As a result, the findings may not generalize to more complex industrial requirements or to domains with highly specialized terminology. Furthermore, the study focuses on the generation of user stories within an educational context. The effectiveness of generative AI in professional requirements engineering processes may differ due to additional factors such as stakeholder negotiation, domain expertise, and organizational constraints. Another limitation is that the experiments were conducted using a single generative AI model, ChatGPT. Different models or model versions may produce different results, particularly as generative AI technologies continue to evolve rapidly.

7.4 Conclusion validity

Conclusion validity concerns whether the conclusions drawn from the study are supported by the available evidence. The experimental evaluation relies primarily on qualitative analysis and limited quantitative comparison of generated user stories. Because the dataset of problem descriptions and generated stories is relatively small, statistical generalizations should be interpreted with caution.

Future work should replicate the study with larger datasets, multiple evaluators, and additional generative models in order to strengthen the reliability of the conclusions

8 Conclusion and future work

Generative Artificial Intelligence has recently emerged as a promising tool for supporting activities in software engineering education. In the context of Requirements Engineering, the ability of Large Language Models to interpret natural language descriptions and generate structured artifacts suggests potential applications as pedagogical assistants for students learning to write requirements.

This paper presented an empirical study evaluating the effectiveness of generative AI, specifically ChatGPT, in generating user stories from textual problem descriptions. The experiments analyzed how different levels of contextual information provided to the model—ranging from minimal problem statements to theoretical explanations and instructor-provided examples—affect the quality of the generated user stories.

The results indicate that the amount of contextual information significantly influences the quality of the generated artifacts. When only the problem description was provided, the model produced syntactically valid but conceptually inconsistent user stories. As additional contextual elements were introduced, the generated stories showed progressively better alignment with instructor-defined solutions and with the INVEST quality criteria. However, increased context also raises the risk of hallucinations and over-specification of non-existent business rules, and some stories did not fully satisfy all INVEST criteria—reinforcing the need for human oversight when using generative AI in requirements engineering tasks.

These findings suggest that generative AI systems can benefit substantially from pedagogical context when applied to educational tasks in Requirements Engineering. Rather than acting as fully autonomous requirement generators, generative models appear to be more effective when used as guided assistants that operate within a well-defined instructional framework. From an educational perspective, this opens interesting opportunities: AI-generated user stories could serve as examples for classroom discussion, as starting points for student refinement, or as automated feedback mechanisms in practical exercises, helping instructors scale feedback in large classes while still encouraging critical thinking about requirement quality.

Future work will extend this research in several directions. First, we plan to conduct classroom studies involving undergraduate students to evaluate how interaction with generative AI tools influences learning outcomes in requirements engineering. Second, future experiments will analyze different generative AI models and prompting strategies to better understand how model capabilities affect requirements generation. Finally, additional evaluation methods, including quantitative scoring and multi-evaluator assessments, will be explored to strengthen the reliability of the results.

Overall, the study provides initial evidence that generative AI can play a valuable role in supporting the teaching and learning of user story specification. As generative technologies continue to evolve, understanding how they can be effectively integrated into requirements engineering education represents an important direction for future research.

References

1. Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y.T., Li, Y., Lundberg, S., et al.: Paper review:'sparks of artificial general intelligence: Early experiments with gpt-4' (2023)
2. Cohn, M.: User stories applied: For agile software development. Addison-Wesley Professional (2004)
3. Denny, P., Prather, J., Becker, B.A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B.N., Santos, E.A., Sarsa, S.: Computing education in the era of generative ai. Communications of the ACM **67**(2), 56–67 (2024)
4. Grigorian, A.: INVEST in Success: A Deep Dive into User Story Optimization. Master's thesis (2024)
5. Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., Wang, H.: Large language models for software engineering: A systematic literature review. ACM Transactions on Software Engineering and Methodology **33**(8), 1–79 (2024)
6. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: Chatgpt for good? on opportunities and challenges of large language models for education. Learning and individual differences **103**, 102274 (2023)
7. Mohamed, A., Assi, M., Guizani, M.: The impact of llm-assistants on software developer productivity: A systematic literature review. arXiv preprint arXiv:2507.03156 (2025)
8. Mollick, E.R., Mollick, L.: Using ai to implement effective teaching strategies in classrooms: Five strategies, including prompts. The Wharton School Research Paper (2023)
9. Ngaliah, N., Siahaan, D., Raharjana, I.K.: User story extraction from online news with featurebased and maximum entropy method for software requirements elicitation. IPTEK The Journal for Technology and Science **32**(3), 125–135 (2022)
10. Norheim, J.J., Rebentisch, E., Xiao, D., Draeger, L., Kerbrat, A., de Weck, O.L.: Challenges in applying large language models to requirements engineering tasks. Design Science **10**, e16 (2024)
11. Ronanki, K., Cabrero-Daniel, B., Berger, C.: Chatgpt as a tool for user story quality evaluation: Trustworthy out of the box? In: International Conference on Agile Software Development. pp. 173–181. Springer (2022)
12. Seibert, V.: Towards dialogue based, computer aided software requirements elicitation. arXiv preprint arXiv:2310.13953 (2023)
13. Siahaan, D., Raharjana, I.K., Fatichah, C.: User story extraction from natural language for requirements elicitation: Identify software-related information from online news. Information and software technology **158**, 107195 (2023)
14. Voria, G., Casillo, F., Gravino, C., Catolino, G., Palomba, F.: Recover: Toward requirements generation from stakeholders' conversations. IEEE Transactions on Software Engineering (2025)