Can LLMs Generate User Stories and Assess Their Quality?

Giovanni Quattrocchi, Liliana Pasquale, Paola Spoletini, and Luciano Baresi, *Member, IEEE*

***Abstract*—Requirements elicitation is still one of the most challenging activities of the requirements engineering process due to the difficulty requirements analysts face in understanding and translating complex needs into concrete requirements that directly impact the quality of the software to be developed. Although automated tools allow for assessing the syntactic quality of requirements, evaluating semantic metrics (e.g., language clarity, internal consistency) remains a manual and time-consuming activity. This paper explores how LLMs can help automate requirements elicitation within agile frameworks, where requirements are defined as user stories. We used 10 state-of-the-art LLMs to investigate their ability to generate user stories automatically by emulating customer interviews. We evaluated the quality of user stories generated by LLMs, comparing it with the quality of user stories generated by humans (domain experts and students). We also explored whether and how LLMs can be used to automatically evaluate the semantic quality of user stories. Our results indicate that LLMs can generate user stories similar to humans in terms of coverage and stylistic quality, but exhibit lower diversity and creativity. Although LLM-generated user stories are generally comparable in quality to those created by humans, they tend to meet the acceptance quality criteria less frequently, regardless of the scale of the LLM model. Finally, LLMs can reliably assess the semantic quality of user stories when provided with clear evaluation criteria and have the potential to reduce human effort in large-scale assessments.**

***Index Terms*—Large language models, requirement engineering, requirement elicitation, requirement quality, user stories, agile methods.**

I. Introduction

REQUIREMENTS elicitation is still a challenging activity due to the difficulty analysts face in understanding and translating complex needs into concrete requirements. In addition, their quality directly impacts the speed of development [1], [2] and the quality of the software to be developed [3]. These problems are exacerbated by the transition to agile system development at scale [4], where the distributed nature of teams and the demands on long-term maintenance make the need for high quality requirements even more pressing [5]. Although frameworks [6] and automated tools [2] exist to evaluate the quality of requirements, assessing semantic metrics (e.g., language clarity, internal consistency) remains a manual and time-consuming activity.

Large Language Models (LLMs) [7], [8], [9] demonstrated high-level performance in natural language processing tasks, and their use is becoming increasingly prominent in automating various software engineering tasks [10], such as code generation [11], [12], program repair [13], [14], [15], and comprehension [16]. As system requirements are usually specified in natural language, LLMs also have the potential to automate requirements engineering activities [17], particularly requirements elicitation. Moreover, since requirements quality assessment is usually based on textual analysis, LLMs could also help with that.

The application of LLMs to requirements engineering has gained significant traction, ranging from ambiguity detection [18], [19] to the automation of elicitation interviews [20], [21]. While recent studies suggest that LLMs can produce specifications comparable to novice engineers [22], [23] and even outperform humans in generating follow-up questions [24], significant research gaps remain open. Specifically, existing literature predominantly focuses on: i) elicitation as a distinct, often single-turn task [22], [23], rather than capturing the structured, multi-turn process typical of analyst-customer interactions; ii) general natural language requirements [18], [25] or formal specifications [26], [27], often overlooking the specifics of agile processes and the use of semi-structured user stories; iii) the generation of requirements without a rigorous evaluation of their semantic quality at scale [23], [28]; and iv) the capability of LLMs to act as reliable assessors [19], [29], which is key to mitigating the manual, time-consuming effort traditionally required for quality validation [2].

In order to tackle these challenges, this paper investigates the capabilities of LLMs to support the entire requirements lifecycle—from the iterative elicitation processes to performing large-scale semantic quality assessment—with a specific focus on agile development where requirements are defined as user stories. We selected user stories as the target artifact for two primary reasons. First, user stories are among the most widely

Received 21 April 2025; revised 10 February 2026; accepted 18 February 2026. Date of publication 11 March 2026; date of current version 1 May 2026. This work was supported in part by the Research Ireland under Grant 13/RC/2094 P2 and in part by the Project EMELIOT under MUR PRIN 2020 program (Contract 2020W3A5FY). Recommended for acceptance by J. L. C. Guo. *(Corresponding author: Giovanni Quattrocchi.)*

Giovanni Quattrocchi is with the Dipartimento di Informatica "Giovanni Degli Antoni," Università degli Studi di Milano, 20133 Milano, Italy (e-mail: giovanni.quattrocchi@unimi.it).

Liliana Pasquale is with the University College Dublin, D04C1P1 Dublin, Ireland (e-mail: liliana.pasquale@ucd.ie).

Paola Spoletini is with Kennesaw State University, Kennesaw, GA 30144 USA (e-mail: pspoleti@kennesaw.edu).

Luciano Baresi is with the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: luciano.baresi@polimi.it).

Digital Object Identifier 10.1109/TSE.2026.3670612

adopted specification means in industrial practice and are supported by a rich body of literature and established evaluation frameworks [30], making them a suitable and accessible target for a systematic study. Second, the semi-structured template of user stories offers a controlled format that facilitates a rigorous evaluation while still reflecting real-world practice [31]. Building on this foundation, we examine whether LLMs can: (i) generate user stories that are comparable to those elicited by human experts, (ii) evaluate the overall quality that characterizes the stories they produce, and (iii) reliably assess the semantic quality of user stories in a way that aligns with human judgment.

In our study, we used 10 state-of-the-art LLMs to emulate a real-world interview-based requirements elicitation process, where an analyst conducts a multi-turn interview with a customer to gather, interpret, and formalize user stories. For this purpose, we replicated the experiment conducted by Ferrari et al. [32], which offers a rigorous experimental protocol and an expert-curated ground truth essential for objectively evaluating LLM performance. In the study, 30 students were asked to impersonate the requirements analyst to conduct interviews with a fictitious customer to elicit user stories. The user stories generated by the students were compared with a set of user stories generated by domain experts that were used as the ground truth. We used 30 instances of each LLM as requirements analysts conducting the interview, and one instance impersonated the customer. We asked each instance acting as an analyst to generate a maximum of 50 user stories, producing a total of 14,540 user stories.

In our analysis, we distinguish between syntactic and semantic qualities of user stories. Syntactic qualities concern the form and structure of the story, that is, whether it follows the canonical template ("As a <role>, I want <goal>, so that <rationale>") and avoids surface-level defects such as excessive conjunctions, missing roles, or duplicated acceptance criteria. These aspects of well-formedness can be reliably detected with automated tools including deterministic checks and lightweight NLP-based methods [31], [33]. In our study, semantic qualities include aspects such as the precision with which a feature is described, the clarity of its rationale, the extent to which it is problem-oriented rather than solution-driven, the unambiguity of its language, and its internal consistency. Moreover, we also considered set-level semantic quality in terms of coverage, defined as the extent to which a set of user stories captures the semantics of an expert-curated dataset. Unlike syntactic qualities, semantic qualities require interpretation and cannot be captured by simple checks.

Our results indicate that although LLMs can generate user stories similar to human experts in terms of coverage and stylistic quality, they exhibit lower diversity and creativity and high consistency.

All LLMs, particularly Claude 3 Opus, could reliably assess user stories semantic quality when equipped with clear evaluation criteria (complete codebook) and can potentially reduce the human effort to evaluate a large set of user stories. Although LLMs can generate user stories that have a high average syntactic and semantic quality, they tend to produce fewer user stories that pass the acceptance criteria compared to those generated by humans, regardless of the size of the model. Common defects include excessive conjunctions, low feature specificity, and clarity of rationale, which reveal the limited capacity of LLMs to contextualize user stories. In summary, LLMs can be used to improve the structural and syntactic quality of human-generated user stories. They could support requirements analysts in generating an initial set of user stories, which would still require human oversight to increase their diversity, specificity, and clarity of rationale.

While our empirical evaluation is necessarily grounded in the current generation of LLMs, our goal is not to compare the models. Instead, we aim to identify requirements elicitation and quality assessment tasks that are structurally amenable to automation, and those that remain inherently dependent on human reasoning and creativity.

The remainder of the paper is organized as follows. Section II describes the study design and research questions. Section III presents the replicated experiment on user story generation. Section IV outlines the generation and assessment of user stories. Section V compares LLM-generated stories with those written by experts. Section VI investigates the ability of LLMs to evaluate the quality of user stories. Section VII reports on the quality of the stories generated by different LLMs. Section VIII tries to identify the implications of this study on requirements engineering. Section IX discusses threats to validity. Section X reviews related work, and Section XI concludes the paper.

II. Study Design

As shown in Fig. 1, we designed our study to address the identified research gaps through a structured pipeline divided into two phases: *Generation & Assessment* and *Evaluation*. In the following, we first introduce the Research Questions (RQs) that drive our investigation, and then detail how the two phases are instantiated to answer them.

RQ1: How good are LLMs at generating user stories compared to those elicited by human experts?

RQ2: How effective are LLMs in evaluating the semantic quality of user stories?

RQ3: What is the syntactic and semantic quality of the user stories generated by LLMs?

These questions have been formulated to directly address the research gaps identified in Section I, moving beyond static generation tasks to assess whether LLMs can sustain a structured, multi-turn elicitation process (RQ1, RQ3) and effectively automate quality assessment to mitigate manual effort (RQ2).

Generation & Assessment. This phase (steps ❶–❸ in Fig. 1) produces the artifacts necessary to enable the analysis required to answer the RQs. To investigate the generation capabilities targeted by **RQ1** and **RQ3**, in step ❶ we use an LLM-based multi-agent architecture to automatically generate a large-scale dataset of user stories (*LLM US*). Subsequently, to enable the investigation of reliable automated assessment for **RQ2**, we establish a human baseline in step ❷. Here, human annotators (*Human Labels*) manually evaluate the semantic quality of a

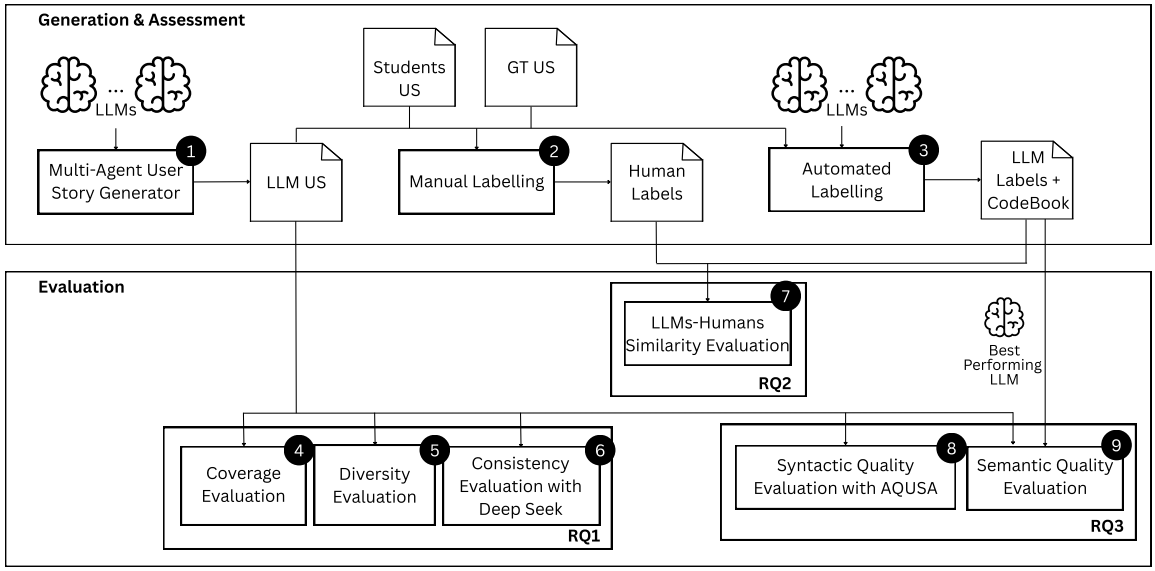


Fig. 1. An overview of our study.

curated subset of user stories generated by domain experts (*GT US*), students (*Students US*), and LLMs. From this manual process, we also curated a *Code Book* containing guidelines and examples, which is then used in step to guide the ❸ LLMs in evaluating the user stories semantic quality (*LLM Labels*).

Evaluation. In this phase (steps ❹–❾), we perform the specific analyses to answer our RQs using the artifacts produced in the previous phase. Addressing the limited focus on structured elicitation in agile contexts (gaps i and ii), **RQ1** studies whether LLMs can generate user stories that have content similar to that elicited by human experts, thus understanding to what extent LLMs can replicate human expertise in a multi-turn interview setting. To evaluate RQ1, we measure whether the *LLM US* cover the content of the user stories in the ground truth (step ❹), examine their diversity across independent runs (step ❺), and assess their internal consistency (step ❻).

Addressing the need to reduce the manual effort required for quality validation, **RQ2** focuses on the potential of LLMs to evaluate the quality of user stories, examining how effectively these models can act as evaluators and how their assessments align with human evaluations. To do this, we compare the *LLM Labels* generated in step ❸—using different levels of guidance based on the codebook—against the ground truth established by manual annotation in step ❷, measuring the agreement between the two (step ❼).

Finally, to overcome the limitations of small-scale qualitative studies, **RQ3** measures the quality of the user stories generated by LLMs at scale, analyzing both syntactic and semantic metrics proposed in the QUS framework [6]. To answer RQ3, we assess the *LLM US* using automated syntactic checks from AQUSA (step ❽) alongside semantic metrics evaluated using the LLM that achieved the best agreement with humans in RQ2 (step ❾). Given the positive results obtained in RQ2, using an LLM to automate this step enabled us to rigorously evaluate the full dataset, which would have been impractical to perform manually.

III. Existing Experiment

To evaluate the ability of LLMs to generate user stories, we build on interview-based learning (IBL) approaches. Interviews are a widely used elicitation technique in requirements engineering [34]. We started from the findings of an existing experiment conducted by Ferrari et al. [32]. In the original experiment, students participated in an Interview-Based Learning (IBL) process, emulating a real-world requirements elicitation process that allowed them to gather, interpret, and formalize user stories. The target application in this experiment was a summer camp management system for which an expert-curated set of user stories already exists[1]. This dataset contains 53 user stories and is henceforth referred to as the ground truth dataset (*GT US*). Each student acted as an analyst and interacted with a fictitious customer, impersonated by one of the authors of the original study. The user story generation process was organized in four steps. Besides resembling a realistic industrial scenario, this experiment provides a well-documented and fully specified benchmark that enables systematic comparison between different classes of analysts under identical elicitation conditions.

Preparation. Students, acting as analysts, received a brief overview of the target application and were asked to prepare questions for customer interviews to elicit product requirements. The customer was equipped with the *GT US* which served as an initial knowledge base that allowed the customer to answer the students' questions consistently and accurately.

Interview. Each student conducted a 15-minute interview with the customer, during which they asked both their prepared questions and their spontaneous follow-up questions.

[1]These user stories are available at https://zenodo.org/records/13880060 in file *g12-camperplus.txt*

The customer's responses were based on the provided *GT US*, which were not disclosed to the students. The students recorded the interviews and took detailed notes for later analysis.

Follow-up Interview. After the initial interview, each student reviewed their notes to identify gaps and generate additional questions for a follow-up interview. Then, a second 15-minute interview was conducted, where each student posed their newly generated questions to the customer to refine and expand their understanding of the requirements.

Requirements Elicitation. Each student used the information collected from both interviews to create 50 to 60 user stories for the system. This range was chosen to roughly match the size of *GT US*, allowing a comparative analysis of the generated data. Students were not intended to represent professional requirements analysts, but rather to provide a reproducible human baseline reflecting novice elicitation performance, which is commonly used in empirical software engineering studies [35].

In total, the final dataset contained 1530 user stories produced by 30 different students (*Students US*). This existing dataset, together with the ground truth (*GT US*) authored by experts, served as the baseline against which we compared the performance of LLM-based analysts.

The experimental setup intentionally adopts a controlled and partially artificial configuration, where students act as requirements analysts, and the customer responses are grounded in a predefined expert-curated set of user stories. However, it allows us to use a stable ground truth against which coverage and semantic alignment can be measured, and compare across analysts and elicitation runs.

While Ferrari et al. [32] provides a well-defined and reproducible instantiation of interview-based learning for user story elicitation, similar interview-centric methodologies have been adopted in other studies to elicit and formalize requirements artifacts. For example, subsequent work [36] has extended the same methodological framework in different empirical settings, while another study [37] relies on interviews as the primary empirical source and applies structured post-hoc analysis to derive system goals.

IV. User Story Generation and Assessment

A. Multi-Agent User Stories Generation

While the original experiment by Ferrari et al. [32] involved students acting as analysts; in our study, we replicated the same methodology using LLMs. We generated user stories by implementing the IBL process (step ❶), with multiple LLM instances taking the role of analysts (instead of students) and an additional LLM acting as a customer (instead of the human expert). This design preserves the structure and rigor of the original experiment while enabling a large-scale comparison between human- and LLM-generated artifacts. To our knowledge, this is the first work that replicates the IBL process with a fully multi-agent LLM architecture.

We developed the multi-agent architecture using the best-performing LLMs available at the time we started this work, namely *Claude Sonnet 3.5*, *Claude Sonnet 3*, *Claude Opus 3*, *Grok Beta*, *GPT-4*, *GPT-3.5 Turbo*, *LLaMA 3.1 8B*, *LLaMA 3.1 405B*, *Gemini 1.5 Flash*, *Gemini 1.5 Pro*. We instantiated each LLM model 30 times to replace the students and the customer, ensuring consistency in each run. We did not fine-tune any model; each LLM instance operated in an independent context configured with the default sampling temperature. We replicated the IBL process as follows.

Preparation. We provided the 30 LLM instances (Analyst LLM) with the same brief description of the app provided to the students in the original experiment. The additional LLM instance (Customer LLM) received the same *GT US* that were originally given to the human expert. Real customers do not possess requirements as user stories, but encoding domain knowledge in this form allows the customer role to respond consistently across interviews and ensures that elicited requirements can be objectively evaluated.

Interview. Each Analyst LLM engaged with Customer LLM by asking a set of questions. The Customer LLM answered such questions with the *GT US* to answer as coherently with the given domain as possible.

Follow-up Interview. Based on the new knowledge acquired, each Analyst LLM generated a new set of questions for further clarification.

Requirements Elicitation. After the second interview, we asked each Analyst LLM to generate up to 50 user stories, matching the size of the dataset created in the original experiment to allow comparative analysis.

For both the Analyst and Customer roles, we adopted a Role-based prompt engineering pattern, instructing each LLM to behave consistently with its assigned role in the IBL workflow. Although the main structure of the prompts was uniform across models, minor adaptations were applied to match the input format of different LLMs. Full prompt templates and implementation details are available in our replication package (see Section *Data Availability* at the end of the manuscript). At the end of the requirements elicitation phase, we obtained a dataset of *LLM US* with $14,540$ user stories, corresponding to about 50 stories per LLM instance. The total is slightly below the expected $15,000$ because some instances produced fewer outputs due to their limited prompt-following capabilities.

B. Manual Labeling

To establish the ground truth to assess whether LLMs can be used to assess the quality of user stories (RQ2), the authors (human annotators) independently assessed the semantic qualities of a dataset of 153 user stories, which included 53 *GT US*, 50 randomly selected user stories created by students, and 50 randomly selected user stories generated by an LLM (GPT-3.5) (step ❷). The human annotators conducted the labeling process blindly, that is, they were unaware of the source of each user story.

They labeled the user stories according to the semantic metrics proposed in the QUS framework [6]. We focused on qualities applied to a single US, not the entire set. Avoiding qualities based on inter-story dependencies allowed us to assess the capabilities of LLMs more directly, yielding clear and

actionable insights into their strengths and weaknesses. We measured **conceptual soundness** through **Feature Specificity (FS)**, which measures the specificity with which the user story describes a feature and highlights how specific and direct the explanation is, and **Rationale Clarity (RC)**, which measures how clearly the reason or justification behind the need for a feature is articulated in a user story. It assesses the clarity with which the user's goal or problem is explained. We considered **Problem-Oriented (PO)** to measure how clearly the problem is specified without implying solutions. Finally, we addressed **unambiguity** by means of **Language Clarity (LC)**, which measures the clarity and precision of the language used, and **Internal Consistency (IC)** to measure whether a user story is internally consistent and free of contradictions.

The human annotators evaluated each metric using a three-point scale: unacceptable (1), acceptable but has margins of improvement (2), or good (3). In addition, they also documented the rules and heuristics that they used during the evaluation process. This additional elicited knowledge captured implicit decision-making strategies and improved the consistency of the assessment framework. To ensure consistency and reliability in the manual annotation process, we conducted a multi-stage reconciliation procedure. Initially, two annotators independently labeled the same set of user stories according to the predefined criteria. They then compared their labels and resolved any major disagreements through discussion. The resulting reconciled set was further reviewed and refined by a third annotator, allowing additional verification and harmonization of the labeling strategy.

We systematically analyzed disagreements among annotators to identify *conflicts*, which we define as substantial mismatches in the perceived quality of a user story. Specifically, a conflict occurred when one annotator assigned a score of 1 (unacceptable) and another assigned a score of 2 (acceptable) or 3 (good) for the same quality criterion on the same user story. This approach avoids over-penalizing minor variations (e.g., scoring a user story as 2 vs. 3), and instead targets more substantial discrepancies.

To illustrate how we conducted the manual labeling, we report two representative examples of user stories annotated by human experts using the QUS framework. The first example represents a low-quality user story, receiving the lowest score (*1*) in all semantic dimensions:

```
As an administrator, I want
the camp site to be enticing,
especially to those applications
who utilize the online application
forms.
```

This story was rated poorly because it lacks a clear and actionable goal, introduces vague and subjective terms such as "enticing", and does not specify what functionality or outcome is expected. The rationale is implicit and the target users are not clearly defined, leading to ambiguity across all QUS dimensions.

Conversely, the following example represents a high-quality user story, rated with the maximum score (*3*) in all semantic dimensions:

```
Rationale Clarity (RC)
- The US should clearly justify the actor's
    rationale.
- To assign score 3, the US should have the
    rationale explicitly expressed through a
    subsentence that could start with "so that," "as
     a" or similar phrases.
- The US should include a rationale or a description
     of the problem; otherwise, it should be
    assigned score 1.
- If the rationale is implied through details given
    in the feature description, score 2 should be
    assigned.

Examples:
- Score 3: "As a parent, I want to be able to enroll
     my children so that they can be admitted to
    camp."
- Score 2: "As the administrator, I would like the
    system to have a forum area where I can
    communicate directly with all employees about
    camp rules or feedback."
- Score 1: "As a camp administrator, I want to be
    able to create and modify rules that campers and
     camp workers have to follow."
```

Listing 1. Codebook Excerpt.

```
As a branch campsite manager, I
want to be able to look at the
schedule for the week, so I know
which counselors I will have
working in my area.
```

This story is clear, specific, and goal-oriented. It explicitly identifies the actor, the desired capability, and the underlying rationale, providing sufficient detail to support implementation and validation.

At the end of the reconciliation process, we used the evaluation rules and guidelines formulated by the annotators to compile a structured *codebook*, that provides a reference framework for assessing user stories quality in future analyses. Listing 1 is an excerpt from the codebook compiled during the reconciliation process, illustrating the evaluation criteria applied to the Rationale Clarity metric.

The excerpt provides both the evaluation criteria and illustrative examples. The examples highlight how different levels of rationale clarity manifest in practice. user stories rated with a score 3 provide a clear and explicit rationale, making the actor's intent fully transparent, as illustrated in the first example. In contrast, a score 2 user story contains an implicit rationale, where the reasoning can be inferred, but is not explicitly stated (see the second example). Lastly, a score 1 user story lacks rationale, making it difficult to understand the underlying motivation for the requirement (as demonstrated in the third example).

C. Automated Labeling With LLMs

To evaluate the assessment capabilities of the 10 selected LLMs (step ❸), we applied three prompting strategies to the set of 153 manually annotated user stories, grounding our approach in established prompt engineering techniques. First, we employed a *basic prompt* to implement a *Zero-Shot* setting, testing

the models' ability to assess quality relying solely on their pre-trained knowledge without provided examples. Second, we utilized a *codebook-augmented prompt* that instantiates the *Context Manager* pattern [38]. It leverages *Few-Shot Learning* principles [9] by injecting the full expert-curated codebook to explicitly guide reasoning through definitions and examples. Third, we designed a *partial codebook-augmented prompt* to mitigate potential *information overload* phenomena in long contexts [39], providing a concise summary of the rules to investigate the trade-off between contextual informativeness and alignment with human experts.

V. Comparing LLM and Expert User Stories

To evaluate the similarity between LLM- and human-generated user stories (RQ1), we used three metrics: i) coverage, ii) diversity, and iii) consistency.

A. Coverage

Coverage measures how well a set of user stories captures the requirements expressed in the *GT US*. Specifically, it quantifies the semantic overlap between the two sets of user stories using embedding-based similarity.

Metrics. To calculate coverage, we first converted all the user stories (*GT US*, *LLM US*, *Students US*) into a vector representation using OpenAI's *text-embedding-3-small* model. For any two sets of user stories X and Y, we define the coverage of X with respect to Y as the percentage of user stories in Y with at least one semantically similar counterpart in X. Two user stories are considered semantically similar if their *cosine similarity* equals or exceeds a set threshold c.

We measured cosine similarity within a range of [-1,1], where 0 indicates no similarity, 1 represents identical texts, and -1 texts with opposite meanings. To determine an appropriate threshold for considering two user stories as semantically similar, we empirically identified two possible thresholds: 0.8 (less strict) and 0.85 (more conservative). We selected these thresholds based on a human evaluation study designed to calibrate similarity assessments. We used human annotators (the four authors, two of whom worked in pairs) to manually assess pairs of user stories. For each similarity bin (ranging from 0.6 to 0.85 in steps of 0.05), we randomly selected 20 pairs from the full dataset of user stories including (*GT US*, *LLM US*, and *Students US*) for a total of 120 pairs. The annotators independently determined whether each pair of user stories conveyed the same meaning without being informed of the actual similarity score or the respective authors. A pair was considered a match if at least two annotators agreed that the two user stories were similar. Table I summarizes the results of this human evaluation study where columns $\%Match$ and $\#Match$ are, respectively, the percentage and the number of pairs marked as similar by annotators, while column $Count$ is the total amount of user stories in a given cosine similarity bin.

The results show that at a cosine similarity threshold of 0.8, human annotators considered 72.5% of the user stories pairs similar, making it a reasonable threshold to capture semantic equivalence. At 0.85, the agreement increased to 95.0%, indicating a stricter but also more reliable threshold. Lower thresholds, such as 0.6 or 0.65, resulted in considerably lower agreement levels, suggesting a higher risk of including semantically dissimilar stories. We therefore use c=0.80 as a balanced threshold and c=0.85 as a conservative setting in subsequent analyses. Note that Table I quantifies only the *precision* of the cosine similarity threshold c (i.e., how trustworthy a high similarity score is). The coverage is then calculated as the fraction of *GT US* that has at least one match above c, and results are reported and described in Section V-A. In practice, selecting c=0.85 yields stricter matching: fewer false positives and higher false negatives (since pairs below 0.85 are all considered non-matching), On the contrary, c=0.80 offers a less conservative trade-off with more false positives and fewer false negatives.

TABLE I
Determining the Similarity Thresholds

Bin	% Match	# Match	Count
0.6-1	28.3%	34	120
0.65-1	34.0%	34	100
0.7-1	42.5%	34	80
0.75-1	53.3%	32	60
0.8-1	72.5%	29	40
0.85-1	95.0%	19	20

TABLE II
Coverage Results of LLMs Against Ground Truth (Sorted by $c = 0.8$)

LLM	c = 0.8	c = 0.85
Claude 3 Sonnet	96.23	88.68
Claude 3 Opus	96.23	84.91
Gemini 1.5 Pro	90.57	77.36
Llama 3.1 405B	81.13	64.15
Grok	81.13	43.40
Claude 3.5 Sonnet	79.25	49.06
Gemini 1.5 Flash	77.36	50.94
GPT-4	75.47	43.40
GPT-3.5	73.58	37.74
Llama 3.1 8B	58.49	16.98
Students Coverage	52.83	15.09

Results. We evaluated the coverage of the set of user stories generated by the 10 LLMs and the students of the original experiment (step ❹). Table II shows that LLMs consistently outperform students in generating user stories that align with the ground truth (*GT US*). At both similarity thresholds (0.8 and 0.85), LLMs demonstrate significantly higher coverage, meaning they produce user stories more similar to the reference set of validated requirements. For example, *Claude 3 Sonnet* and *Claude 3 Opus* achieve a top coverage of 96.23% at the 0.8 threshold, while even the lowest-scoring LLM (*LLama 3.1 8B*) reaches 58.49%. In contrast, students only reach 52.83% at the same threshold. When the threshold is raised to 0.85, the gap widens further: *Claude 3 Sonnet* and *Claude 3 Opus* retain high coverage (88.68% and 84.91%), but students drop sharply to just 15.09%.

This trend suggests that LLMs are highly capable of capturing the *core* requirements contained in the *GT US*. Their

output tends to be semantically consistent with the reference set. This is likely due to general-purpose training in large corpora of structured text and task-oriented patterns. In contrast, students struggle more with structural consistency and may miss some fundamental aspects of the original requirements.

However, these results must be interpreted correctly. The ground truth used for evaluation only represents a portion of the full requirement space—typically, the more standard or fundamental needs of the system. High coverage with respect to this reference set is not necessarily indicative of a model's ability to explore the broader, more creative, or domain-specific requirements that stakeholders may expect. The tendency of LLMs to stay close to known patterns may limit their ability to propose novel or unconventional user stories. Therefore, while the high coverage with the *GT US* shows precision and alignment with the application domain, it could also reflect a conservative generation strategy that favors familiarity over exploration. In terms of temperature, we used the default one for all the LLMs, which guarantees a balanced trade-off between creativity and consistency in the generated outputs. Higher temperature values may have led to lower coverage values that favor the generation of more diverse and nuanced user stories.

Moreover, the higher coverage of LLMs might lead to over-reliance on automated tools, potentially reducing the involvement and creativity of stakeholders in the requirements engineering process. Despite their lower performance in this experiment, human analysts may still contribute more nuanced, contextual, or innovative perspectives that are difficult to evaluate through coverage alone. This hypothesis is supported by the results we obtained for the diversity metric.

B. Diversity

While *coverage* quantifies how well one set captures another, *diversity* captures the semantic variability between different generators of user stories. High diversity indicates that the generators produce complementary and varied content, while low diversity suggests a convergence toward similar outputs.

Metrics. To compute diversity, we build on coverage and define the diversity between two sets of user stories X and Y as the complement of their coverage: $Diversity(X, Y) = 100 - Coverage(X, Y)$, where the coverage is calculated using $c = 0.8$. To estimate the overall diversity of a group—such as a cohort of students or multiple instantiations of the same LLM—we compute pairwise diversity scores across all distinct pairs of user stories sets within the group. Let $\mathcal{S} = S_1, S_2, \ldots, S_n$ be the collection of n user story sets generated by n individuals or instances. The average pairwise diversity within the group is given by:

$$AvgDiversity(\mathcal{S}) = \frac{2}{n(n-1)} \sum_{1 \le i < j \le n} Diversity(S_i, S_j) \tag{1}$$

This formulation captures how varied the user story sets are within different sources. In our study, we apply this metric to compare the diversity among different instances of the same LLM and among user stories produced by different students.

TABLE III
AVERAGE DIVERSITY OF LLMs AND STUDENTS

Generator	*AvgDiversity* (%)
Students	98.58
Grok Beta	74.74
LLaMA 3.1 8B	73.36
LLaMA 3.1 405B	65.67
GPT-4	62.21
Claude 3.5 Sonnet	56.08
Claude 3 Opus	55.35
Gemini 1.5 Flash	52.85
Gemini 1.5 Pro	49.38
GPT-3.5	48.38
Claude 3 Sonnet	44.23

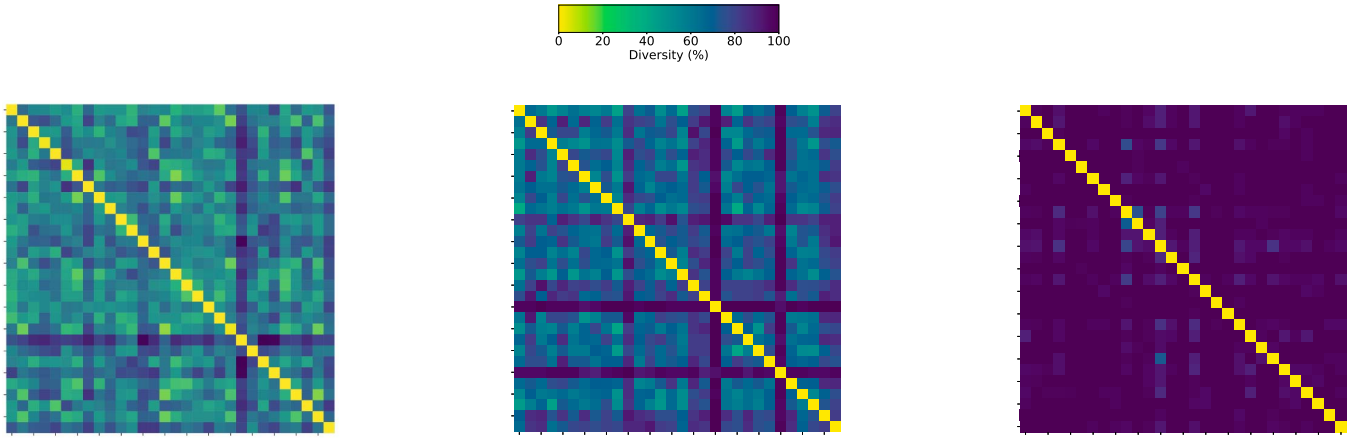
Results. Table III shows the average diversity of the user stories produced by the most and least diverse LLMs and by students (step ❺). Fig. 2, where darker colors indicate greater pairwise diversity, while lighter colors point to low diversity. The complete figure can be found in the replication package (see Section Data Availability at the end of the paper).

The students exhibit the highest diversity (98.58%), far exceeding any LLM. This is clearly visible in Fig. 2(c), where the non-diagonal elements of the matrix are almost uniformly dark, indicating minimal redundancy. In contrast, most LLMs (in the figure we show Claude Sonnet 3.5 as an example) show brighter horizontal or vertical bands, revealing runs that consistently produce similar outputs.

This result confirms the intuition that humans, even when faced with the same task, tend to express requirements in highly distinct ways. Their variation stems not only from vocabulary and sentence structure, but also from different interpretations of domain concepts, priorities, and expression styles. Among LLMs, the highest diversity is observed in Grok Beta (74.74%) and LLaMA 3.1 8B (73.36%), followed by the larger LLaMA 3.1 405B (65.67%) and GPT-4 (62.21%). This suggests that these models introduce more internal variability across independent generations, possibly due to architectural differences, decoding strategies, or higher sensitivity to random sampling noise. In contrast, models like Claude 3 Sonnet (44.23%) and GPT-3.5 (48.38%) produce significantly more homogeneous outputs across different runs.

Note that, in our setting, diversity does not arise from a single one-shot prompt but from a multi-step workflow that replicates the interview-based process. Each Analyst LLM instance prepares questions, conducts two rounds of interviews with the Customer LLM, and then synthesizes user stories from collected information. At every step, the non-determinism inherent to LLMs—such as stochastic sampling, contextual interpretation, and phrasing variability—can introduce subtle deviations that accumulate across runs.

The diversity in generated requirements has complex implications. On the one hand, high diversity can be advantageous in the early stages of requirements elicitation, where a wider set of ideas is valuable. It allows a broader coverage of the requirements space and may surface alternative perspectives. For example, the behavior of Grok Beta and LLaMA 3.1 8B



(a) Claude Sonnet 3.5 (b) Grok Beta (c) Students

Fig. 2. Diversity between the most and least diverse LLMs and Students.

in this respect may be desirable in contexts where creative variability is important. However, excessive diversity might be problematic for standardization or alignment with domain-specific terminology. A more consistent generation (as seen with Claude 3 Sonnet or GPT-3.5) could be preferable in such settings, particularly when the LLM is integrated into structured pipelines. Moreover, diversity must be evaluated jointly with coverage: a model that produces diverse but irrelevant stories is not necessarily useful.

Interestingly, larger models do not always produce more diverse outputs. For example, LLaMA 3.1 405B (405 billion parameters) shows lower diversity than its smaller counterpart, LLaMA 3.1 8B. This may indicate that larger models are better at converging toward high-probability, canonical phrasings, effectively reducing internal variability. This observation cautions against assuming that scaling up model size inherently improves the diversity of generated content.

The sharp contrast between student diversity and that of all LLMs underscores the gap in internal variability. Although LLMs may approach human-level performance in fluency, they appear to lack the range of semantic and expressive variation seen in real users. Increasing the temperature parameter beyond the default value can enhance the diversity of LLM outputs, potentially leading to more varied generations.

C. Consistency

Consistency refers to the absence of contradictions or mutually exclusive requirements within the set of user stories generated by a single LLM instance, ensuring that the stories can coexist without logical conflicts (step ❻).

Metrics. To assess the consistency of a set of generated user stories, we sampled three Analyst instances for each LLM and considered all possible pairs of their generated user stories. We used DeepSeek[2], which is an LLM with state-of-the-art intelligence capabilities and that was not part of the cohort of models analyzed in our study, to do it. Following a set-up similar to [40], we asked DeepSeek to classify each pair as: NEUTRAL, when the two stories are unrelated; ENTAILED, when one story logically implies the other (indicating some redundancy); and CONTRADICTING, when the stories contain conflicting requirements. We instructed DeepSeek using a role-based prompt pattern in a zero-shot setting.

[2]https://api-docs.deepseek.com

TABLE IV
CONSISTENCY

Model	# User Story	#P	C%	E%	N%
Claude 3.5 Sonnet	149	3626	0.00	0.55	99.45
GPT-3.5	142	3291	0.00	4.31	95.69
GPT-4	146	3482	0.00	1.55	98.45
Gemini 1.5 Pro	160	4188	0.00	2.08	97.92
Grok Beta	149	3626	0.00	1.24	98.76
LLaMA 3.1 405B	141	3268	0.00	3.43	96.57
LLaMA 3.1 8B	170	4790	0.00	3.90	96.10
Claude 3 Opus	147	3528	0.03	1.73	98.24
Claude 3 Sonnet	110	2031	0.05	1.33	98.62
Gemini 1.5 Flash	149	3632	0.06	2.15	97.80

Results. The results reported in Table IV show that the overwhelming majority of pairs (#P) are NEUTRAL (N) (always above 95%), a small fraction are ENTAILED (E) (typically 1–4%), and almost none are CONTRADICTING (C). This indicates that LLM-generated datasets are not only semantically complete with respect to the ground truth, but also largely free from internal contradictions.

D. Answering RQ1

Our analysis shows that LLMs can effectively generate user stories that match the requirements represented in *GT US*, achieving high coverage scores across models. This suggests that LLMs can align well with fundamental requirement patterns, often producing syntactically clean and semantically precise outputs. However, this high coverage may come at the cost of variability and originality. Compared to students, who display much higher output diversity, LLMs tend to generate more homogeneous stories, pointing to a narrower exploration of the

requirement space. This lack of variability might limit their usefulness in early ideation phases where breadth is essential.

Another set-wise property in addition to coverage is consistency. The results show that most of the user stories generated by LLMs are not related to each other, some repeat similar content, and almost none introduce conflicting requirements. This suggests that the user stories generated by LLMs achieve high coverage of the ground truth while remaining largely free of internal contradictions.

In summary, LLMs exhibit strong potential in supporting user story generation. However, their tendency toward structural regularity and limited variation distinguishes them from human experts, who naturally infuse creativity and contextual nuance into requirements specification.

RQ1: How good are LLMs at generating user stories compared to those elicited by human experts?

LLMs can generate user stories similar to human experts in terms of coverage and stylistic quality, particularly for basic requirements. However, they exhibit lower diversity and creativity, which may limit their effectiveness in exploratory or nuanced elicitation tasks. The generated user stories exhibit a high level of consistency.

VI. LLM-based Assessment of User Stories

To understand whether LLMs can be used to evaluate the quality of user stories (RQ2), we calculated the agreement between the LLM-generated labels and those created by humans (the authors) after conflict resolution. We used Cohen's Kappa metric (κ) [41], which quantifies the level of agreement between two raters while accounting for the agreement that could occur by chance. A value of κ of 1 indicates perfect agreement, 0 corresponds to agreement at the chance level, and negative values imply systematic disagreement. In our context, higher values of κ indicate that the behavior of the LLM as an evaluator is more similar to that of human annotators, reflecting a better ability to assess the quality of user stories.

To focus on the most meaningful cases of disagreement, we adopted a conflict-aware formulation of κ. Specifically, a disagreement between an LLM and human annotation was only counted when there was a conflict as defined in Section IV-B. In this setting, κ better reflects the alignment of an LLM with the decision boundaries that humans deem critical to determine the adequacy of a user story.

Cohen's Kappa was calculated separately for each of the five quality dimensions: FS (Feature Specificity), RC (Rational Clarity), PO (Problem Oriented), LC (Language Clarity), and IC (Internal Consistency) as defined in Section IV-B. This allows us to identify which quality aspects are more or less reliably assessed by different models. In general, a higher κ indicates a greater ability of the LLM to act as a reliable evaluator of the quality of user stories, potentially reducing the effort required in manual validation workflows. In contrast, lower agreement highlights areas where LLMs struggle to interpret or apply quality criteria in a manner consistent with human experts, even when detailed codebooks are provided. Given the task, we configured all the LLMs with a temperature of 0 (as suggested by Renze [42]) to ensure deterministic behavior and eliminate randomness in generation, allowing for a more controlled and reproducible comparison across models.

TABLE V
Inter-Annotator Agreement Before Conflict Resolution (Cohen's Kappa)

Criterion	E1 vs E2	E1 vs E3	E2 vs E3
FS	SU (0.72)	SU (0.79)	SU (0.76)
RC	SU (0.75)	SU (0.67)	AP (0.81)
PO	SU (0.72)	SU (0.72)	MO (0.52)
LC	SU (0.70)	AP (0.83)	SU (0.79)
IC	SU (0.69)	AP (0.85)	SU (0.72)
Overall κ	SU (0.76)	SU (0.78)	SU (0.74)

A. Agreement Between Human Experts

To understand the results obtained by LLMs, here we show the agreement among human experts (the four authors of the paper with two working together) during the manual labeling phase and before the reconciliation process.

The values reported in Table V were interpreted according to standard qualitative thresholds for Cohen's Kappa: *Poor* (PO) ($\kappa < 0$), *Slight* (SL) ($0 \leq \kappa < 0.2$), *Fair* (FA) ($0.2 \leq \kappa < 0.4$), *Moderate* (MO) ($0.4 \leq \kappa < 0.6$), *Substantial* (SU) ($0.6 \leq \kappa \leq 0.8$), and *Almost Perfect* (AP) ($\kappa > 0.8$). The table shows that most of the comparisons yielded *Substantial* agreement, with several criteria —particularly LC and IC— reaching *Almost Perfect* agreement for some pairs of annotators. Only one case fell in the *Moderate* range, indicating a slightly lower consistency in judging PO between two experts. Overall, these results confirm a strong level of alignment among the annotators, although the task inherently involves a degree of subjectivity. Assessing the quality of user stories requires interpretative judgment, especially when distinguishing between acceptable and good requirements or evaluating the rationale or conceptual clarity of user stories. The observed levels of agreement demonstrate that, despite this subjectivity, the evaluation framework supported reasonably consistent scores among different human experts.

B. LLM-Human Agreement in User Story Assessment

Fig. 3 shows the Cohen's Kappa scores obtained by comparing LLM-generated labels and human-annotated ground truth. High values indicate that LLM-produced judgments align well with those of human experts. The LLMs were instructed with three different prompts: without codebook, with partial codebook, and with complete codebook, that is, from a minimal task description to detailed evaluation criteria (step ❼). The prompts and the different variations of the codebook are available in the replication package.

In general, all LLMs exhibit at least a *Substantial* level of agreement in the three cases, several reaching *Almost Perfect*

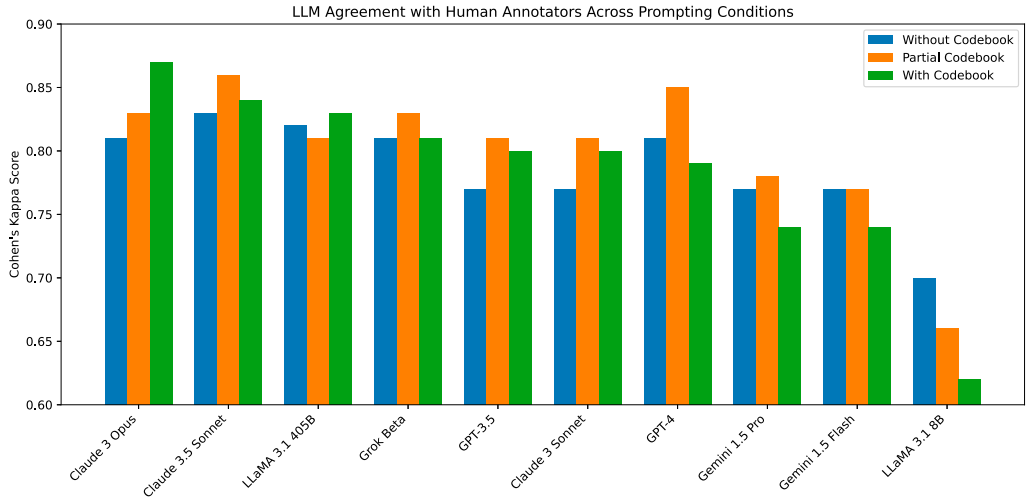


Fig. 3. Cohen's Kappa between each LLM and the human-annotated ground truth (across all quality criteria).

agreement ($\kappa > 0.8$) when provided with the complete codebook. This confirms that LLMs can mirror human decision making in evaluating the quality of user stories, especially when provided with structured and explicit criteria. In some cases, such as Claude 3 Opus ($\kappa = 0.87$) and Claude 3.5 Sonnet ($\kappa = 0.84$), the agreement even exceeds the average inter-human reliability observed in our manual annotation phase, suggesting that LLMs can act as robust evaluators when adequately instructed.

The comparison of the results with different prompts reveals a consistent trend: the codebook acts as an anchor that generally helps models interpret ambiguous or under-specified evaluation tasks.

For example, Claude 3 Opus obtained $\kappa = 0.81$ without the codebook, $\kappa = 0.83$ with partial codebook, and $\kappa = 0.87$ with the full codebook. Interestingly, while all models except LLaMA 3.1 8B and Gemini 1.5 Flash benefit from the codebook, some show diminishing returns or slight reversals. For example, Claude 3.5 Sonnet and GPT-4 perform better with partial codebook guidance ($\kappa = 0.86/0.85$) than with the full version ($\kappa = 0.84/0.78$). This may indicate that excessive prompting or overly detailed instructions can sometimes interfere with the model's internal heuristics, introducing noise or unnecessary constraints. Compared to human-human agreement levels, which were typically between 0.74 and 0.78, the majority of the highest performing LLMs match or exceed this threshold when properly guided. This alignment suggests that LLMs not only reproduce surface-level patterns, but are capable of modeling evaluative behavior comparable to that of domain experts. However, the extent of this capability depends on the clarity of the task, as demonstrated by improvements in prompt conditions.

The lowest performing model in all conditions is LLaMA 3.1 8B, which reaches a maximum κ of only 0.70, still within the substantial range but trailing behind its larger counterpart LLaMA 3.1 405B (which reaches 0.83). This reinforces the observation that the size of the model and the scope of training play a role in the ability to generalize and follow abstract evaluative criteria. More powerful models exhibit better semantic understanding and are more sensitive to prompt structure.

TABLE VI
LLM-HUMAN AGREEMENT (COHEN'S KAPPA) BY QUALITY CRITERION – WITH CODEBOOK

LLM	FS	RC	PO	LC	IC	Overall
Claude 3 Opus	0.89	0.86	0.72	0.92	0.97	**0.87**
Claude 3.5 Sonnet	0.90	0.87	0.60	0.87	0.97	**0.84**
LLaMA 3.1 405B	0.91	0.73	0.68	0.94	1.00	**0.83**
Grok Beta	0.88	0.68	0.72	0.89	0.94	**0.81**
Claude 3 Sonnet	0.86	0.64	0.68	0.90	0.95	**0.80**
GPT-3.5	0.87	0.67	0.70	0.88	0.85	**0.80**
GPT-4	0.81	0.72	0.69	0.85	1.00	**0.79**
Gemini 1.5 Pro	0.77	0.71	0.52	0.92	0.90	**0.74**
Gemini 1.5 Flash	0.88	0.66	0.50	0.87	0.86	**0.74**
LLaMA 3.1 8B	0.77	0.63	0.67	0.54	0.50	**0.62**

The results show that LLMs are already competent assessors of the quality of user stories and can replicate human judgment to a high degree of fidelity. The provision of structured evaluation criteria significantly enhances their performance, confirming that prompt engineering and task framing are critical levers in the deployment of LLMs for quality assurance in requirements engineering. These findings position LLMs as promising tools for (partially) automating human-centered validation workflows.

Table VI provides a detailed view of the agreement between each LLM and the human-established ground truth with the use of a codebook-based prompt. The values represent κ scores for each of the five quality criteria and an overall score calculated as their average.

High-performing LLMs tend to exhibit strong consistency across dimensions, with particularly strong agreement in LC and IC —two relatively less interpretative dimensions. Several models (e.g., GPT-4 and LLaMA 3.1 405B) achieve a perfect score ($\kappa = 1.0$) in Internal Consistency, likely due to the model's ability to recognize basic logical coherence within a short text. More semantically nuanced criteria, such as RC and PO, show greater variability among LLMs. This aligns with the observation that these dimensions require deeper contextual understanding and often reflect implicit intentions.

For example, Claude 3.5 Sonnet, while very strong in RC (0.87), drops to 0.60 in PO, suggesting that it may prioritize clarity of purpose over critical assessment of solution bias. In contrast, Gemini 1.5 Flash and LLaMA 3.1 8B fall behind, especially in LC and IC. The latter performs weakest overall ($\kappa = 0.62$), showing limited alignment across all five dimensions. These gaps may arise from a smaller model size (e.g., 8B vs. 405B), architectural limitations, or differences in alignment and fine-tuning processes. Although model size often correlates with stronger performance, this is not always the case. For example, GPT-3.5 performs as well as GPT-4 overall (0.80 vs. 0.79) and is better in several individual dimensions. This suggests that alignment techniques and prompt compatibility may be as significant as scale in supporting high-quality assessment.

These results show that LLM–human agreement is inherently bounded by the quality of human annotations themselves. Our annotators achieved substantial but not perfect agreement between the raters (Table V), so even the best LLMs can only approximate human reliability within that limit. The performance gains with the full codebook underscore how prompt framing and structured criteria can dramatically shift apparent model performance, and suggest that evaluation protocols and annotation quality are as influential as the choice of the LLM. These nuances indicate that less structured contexts or different annotation schemes may yield a lower alignment than reported here.

C. Answering RQ2

The results presented in this section show that LLMs can act as effective evaluators of the quality of user stories when supplemented with appropriate guidance. Using a conflict-aware Cohen's Kappa metric, we measured the agreement between each LLM and the human-annotated ground truth. By embedding the expert curated codebook, several LLMs achieved agreement scores on par with or even exceeding human-human agreement.

These findings suggest that LLMs can reliably identify user stories that do not meet basic acceptability thresholds, particularly in more surface-level semantic quality indicators such as LC and IC. However, performance remains more variable in semantically demanding areas such as Rationale Clarity and Problem Orientation, where domain understanding and implicit reasoning play a more significant role. Thus, LLMs offer a viable support mechanism for quality assurance in requirements engineering, particularly for tasks that involve detecting basic structural and logical issues.

Note, however, that this effectiveness relies on humans in the loop. The evaluation codebook used in our study was created through rounds of human annotation and reflects expert judgment. Although developing such guidelines requires an initial investment of human effort, once established they can be reused and adapted in many datasets and contexts. In this way, LLMs can help reduce the cost of scaling quality assessment, while humans remain essential for defining and refining evaluation criteria.

RQ2: How effective are LLMs in evaluating the semantic quality of user stories?

LLMs can reliably assess the quality of user stories when equipped with clear evaluation criteria. They perform best on structurally grounded dimensions and can reduce human effort in large-scale assessments. However, their reliability depends on prompt structure, and human oversight remains important for semantically nuanced judgments.

VII. Evaluating Quality of Generated User Stories

We evaluated the syntactic and semantic qualities of the user stories generated by the ten LLMs to answer RQ3.

To evaluate the syntactic quality of LLM-generated user stories, we used the *AQUSA* [31] (*Automatic Quality User per Story Artisan*) framework (step ❸). It applies a structured set of static checks based on a formal taxonomy of quality criteria, targeting syntactic and structural correctness. The tool outputs detailed metrics that capture the presence and distribution of quality defects.

Table VII presents a comprehensive view of the AQUSA-based defect analysis in all LLMs evaluated. We focus on the total number of user stories generated by each model (*Total*), the number of user stories that contained at least one defect (*# Def*), and the corresponding proportion of such user stories over the total (*% Def*, computed as $\frac{\#\text{Def}}{\text{Total}} \times 100$). To assess how concentrated defects are within the faulty user stories, we also compute the *Average Defects* (*Avg Def*), defined as the number of defects divided by the number of defective user stories.

AQUSA categorizes defects into the following distinct types, each representing a different class of syntactic quality issues.

- *atomic.conjunctions* (*conj*): excessive or improper use of conjunctions (e.g., "and") that compromise atomicity.
- *uniform.uniform* (*uniform*): inconsistencies in the format or structure of acceptance criteria among user stories.
- *minimal.brackets* (*brackets*): incorrect use of brackets or parentheses.
- *minimal.indicator_repetition* (*rep*): redundant specification of role or goal indicators.
- *unique.identical* (*identical*): repeated or duplicated acceptance criteria items that should be unique.
- *well_formed.no_means* (*no_means*): missing specification of the "means" in the canonical form: "As a *role*, I want *goal*, so that *reason*".
- *well_formed.no_role* (*no_role*): omission of the user role.
- *minimal.punctuation* (*punct*): punctuation-related issues such as missing or duplicated symbols.

We report all metrics as % of the total number of user stories, except for the absolute counts (Total, # Defected) and the averages.

To evaluate the semantic quality of user stories, we performed a large-scale annotation experiment using the best-performing

TABLE VII
DEFECT METRICS (SORTED BY % DEFECTED)

Model	Total	% Def	# Def	Avg Def	conj	uniform	brackets	rep	identical	no_means	no_role	punct
Claude Sonnet 3.5	1500	2.20	33	1.00	1.67	0.47	0.00	0.00	0.07	0.00	0.00	0.00
GPT-3.5 Turbo	1360	16.69	232	1.02	13.68	2.79	0.00	0.00	0.59	0.00	0.00	0.00
Gemini 1.5 Pro	1595	18.18	326	1.12	5.27	12.66	1.57	0.44	0.19	0.31	0.00	0.07
GPT-4	1490	21.14	378	1.20	10.87	11.01	0.00	0.27	1.21	2.01	0.00	0.00
Gemini 1.5 Flash	1506	23.31	358	1.02	21.05	1.59	0.80	0.27	0.07	0.00	0.00	0.00
Llama 3.1 405B	1452	30.37	579	1.31	24.24	0.07	0.00	0.00	7.02	4.27	4.27	0.00
Claude Opus 3	1481	38.28	641	1.13	28.22	13.91	0.95	0.14	0.00	0.00	0.00	0.00
Llama 3.1 8B	1460	41.58	793	1.31	29.38	1.10	0.14	0.14	15.14	4.25	4.11	0.00
Grok Beta	1500	51.47	1525	1.98	22.40	19.13	0.20	0.60	23.00	26.27	10.07	0.07
Claude Sonnet 3	1196	55.94	798	1.19	57.78	8.03	0.75	0.17	0.00	0.00	0.00	0.00

RQ2 model (Claude 3 Opus) (step ❾), which showed the highest agreement with human experts in structured assessment tasks and, therefore, we selected it as a reference evaluator to label systematically user stories generated by LLMs, those written by students, and those present in the ground truth set. Claude 3 Opus was prompted with the complete codebook (see Section IV-B) and asked to assign scores to each user story according to the five predefined quality criteria.

As for the previous steps, each criterion was scored on a 3-point scale: 1 (non-acceptable), 2 (acceptable), or 3 (good). Based on these annotations, we computed two aggregate measures for each source.

- *Rejection Rate (%)*: the percentage of user stories that received a score of 1 (non-acceptable) on at least one of the five quality criteria. This metric reflects how many user stories fail to meet minimal quality expectations.
- *Avg Score* [[1, 3]: the average of all quality scores assigned to all criteria and user stories from a given source. This provides an overall indicator of the typical quality level observed.

This setup allowed us to compare the perceived quality of the user stories genereted by different sources using an automated and fine-grained approach.

A. Quantitative Evaluation

The results (Table VII) indicate substantial differences in the quality obtained with the different LLMs in terms of the quantity and nature of the defects identified.

Claude Sonnet 3.5 stands out with a remarkably low defect rate, with only 2.20% of user stories containing any issue and 33 defects over 1500 user stories. This reflects a high level of structural and syntactic correctness, further supported by its low average defect count per defective user story (1.00). At the opposite end of the spectrum, Claude Sonnet 3 exhibits the highest defect rate at 55.94%, suggesting a significant decrease in quality relative to its updated counterpart. In particular, this earlier variant shows a very high percentage of atomic conjunction issues (57.78%), which alone may explain much of the inflated defect rate.

Grok Beta also shows critical quality concerns. Despite generating the same number of user stories as Claude Sonnet 3.5 (1500), it produces the highest total number of defects (1525) and a high average of 1.98 defects per defective user story. These defects are frequent and structurally severe. Grok exhibits elevated rates of `unique.identical` (23.00%), `well_formed.no_means` (26.27%), and `well_formed.no_role` (10.07%), pointing to systemic issues in the composition of user stories. Looking at large models, LLaMA 3.1 405B shows a moderate defect rate (30.37%), with notable issues related to identical acceptance criteria (7.02%) and missing components (`no_means` and `no_role` at 4.27% each). Despite its size, it performs worse than some smaller models, such as Gemini 1.5 Flash, which achieves a lower defect rate (23.31%). This suggests that size alone is not a sufficient indicator of reliability in the generation of user stories. GPT-3.5 Turbo and Gemini 1.5 Pro exhibit balanced behavior with relatively low % Def (16.69% and 18.18%, respectively) and average defect counts close to 1. These models suffer from milder issues, such as inconsistent formatting (`uniform`) and atomic conjunctions. GPT-4, although a more capable model in general NLP tasks, shows a higher defect rate (21.14%) than GPT-3.5 Turbo, although with moderate defect density per user story (1.20).

Another key trend concerns the nature of defects. Atomic conjunctions (`conj`) are the most frequently reported issue in general, representing more than 57% in some models. This suggests that while models often generate syntactically correct user stories, they tend to concatenate multiple user intents, compromising atomicity, a known challenge in user stories authoring. Models such as LLaMA 3.1 8B and Claude Opus 3 also exhibit high conjunction rates (29.38% and 28.22%, respectively), confirming that this is a widespread defect class even among the top performing systems.

B. Qualitative Evaluation

Table VIII presents the measured quality of user stories produced by the LLMs, by the students, and *GT US*, as evaluated by Claude 3 Opus, the best performing LLM in the annotation task, in terms of the rejection rate (Rej%), the average score (μS) and the ratings per criterion. The evaluation reveals several trends, particularly in terms of rejection rate, average score, and dimension-specific strengths and weaknesses.

GT US achieved the best overall performance, with the lowest rejection rate (9.43%) and one of the highest average

TABLE VIII
QUALITATIVE EVALUATION PRODUCED BY CLAUDE 3 OPUS

LLM	Rej%	μS	FS	RC	PO	LC	IC
Ground Truth	9.4	2.78	2.74	2.66	2.85	2.68	2.98
Students	14.64	2.70	2.54	2.68	2.68	2.67	2.92
Claude 3 Opus	16.55	2.78	2.60	2.71	2.71	2.92	2.99
LLaMA 3.1 405B	19.23	2.81	2.74	2.65	2.81	2.88	2.98
Gemini 1.5 Pro	20.33	2.77	2.63	2.58	2.81	2.85	2.99
LLaMA 3.1 8B	22.10	2.76	2.78	2.39	2.83	2.87	2.94
Gemini 1.5 Flash	24.85	2.81	2.80	2.36	2.94	2.93	3.00
Grok Beta	25.70	2.70	2.71	2.13	2.82	2.88	2.98
Claude 3.5 Sonnet	33.24	2.76	2.78	2.22	2.96	2.88	2.98
Claude 3 Sonnet	48.37	2.61	2.56	1.95	2.70	2.83	2.99
GPT-4	63.47	2.48	2.48	1.54	2.75	2.71	2.91
GPT-3.5	77.34	2.23	2.15	1.25	2.51	2.47	2.77

scores (2.78). This confirms the quality of these manually curated user stories, which serve as the upper limit in our comparison. *Students US* followed closely, with a rejection rate of 14.64% and an average score of 2.70. This suggests that, despite being non-experts, students can produce reasonably high-quality requirements when guided by instructional material and structure. Among the LLMs, Claude 3 Opus, the model used as the evaluator, scored the user stories it generated very high, with an average score of 2.78 and a rejection rate of 16.55%, closely aligning with human-created content. Some LLMs outperformed Claude 3 Opus in terms of average score: both LLaMA 3.1 405B and Gemini 1.5 Flash reached the highest average score of 2.81. However, this came at the cost of higher rejection rates (19.23% and 24.85%, respectively). This indicates that while these LLMs can produce highly rated user stories, they are also more prone to generating user stories that do not meet the minimum quality standards.

A more detailed view emerges when considering the per-criterion scores. Almost all models perform well in IC and LC, with values often approaching or even reaching 3.00. This is expected, as these dimensions are highly dependent on surface-level linguistic patterns, coherence, and grammar, areas in which LLMs excel. In contrast, models perform significantly worse in RC, reflecting the ability to explain the "why" behind the requirement. This is where human-created content has a noticeable advantage. *GT US* and *Students US* achieve RC scores of 2.66 and 2.68, respectively, while most LLMs lag behind. For example, GPT-4 scores only 1.54 and GPT-3.5 only 1.25, suggesting a systematic difficulty in making explicit the underlying rationale, even when the rest of the user stories is structurally sound. GPT-3.5 has the highest rejection rate (77.34%) and the lowest average score (2.23), with low performance in RC (1.25) and LC (2.47). GPT-4, while stronger overall, also has a high rejection rate (63.47%) and a similarly low RC (1.54). These results suggest that, despite their strong performance in many benchmarks, these models may be less reliable in structured requirements tasks without domain adaptation. Interestingly, even among stronger models like Claude 3.5 Sonnet, the rejection rate remains high (33.24%) despite a solid average score (2.76). This again points to inconsistency: some user stories are rated very well, while others fall below acceptable thresholds.

C. Answering RQ3

The combined evaluation reveals a clear picture of LLM performance in the generation of user stories. The syntactic and structural quality is generally strong, especially in terms of linguistic clarity and consistency. However, models often struggle with deeper semantic aspects, such as articulating the rationale behind features, where human-written user stories retain a clear advantage. Quality variability remains a concern: some LLMs produce excellent outputs but lack consistency, with a non-negligible share of unacceptable user stories. These fluctuations highlight the importance of controlled deployment and the need for human-in-the-loop validation. Model size or recency does not uniformly predict quality: training objectives and alignment matter equally, if not more. Overall, LLMs are promising requirements specification tools, but their limitations call for careful integration and monitoring in practical workflows.

RQ3: What is the syntactic and semantic quality of the user stories generated by LLMs?

LLMs can generate high-quality user stories, especially in terms of clarity and structure. However, their performance decreases on aspects that require reasoning and justification. Quality varies significantly between and within LLMs, with no guarantee that the scale of the model ensures the generation of high quality user stories. This finding suggests that human oversight is necessary.

VIII. KEY FINDINGS AND IMPLICATIONS FOR REQUIREMENTS ENGINEERING

Based on our large-scale analysis, we synthesize our findings into four key observations that explain the current capabilities of LLMs and offer a perspective on their future trajectory in requirements engineering

- LLMs have already reached a level of maturity where they can automate the structural and syntactic aspects of user story generation. They consistently outperform novice analysts in adhering to templates and avoiding surface-level defects. This suggests that LLMs can serve as robust drafting assistants that standardize the output format across large teams.
- A fundamental insight from our study is the dichotomy between LLM convergence and human divergence. While LLMs achieve high coverage of known requirements (ground truth), they exhibit significantly lower diversity compared to human analysts. LLMs tend to converge on the most likely content, whereas humans naturally introduce creative variations that may drive innovation. This suggests that while LLMs can be a valid starting point for generating core requirements, especially in a well-known domain, human-driven development remains crucial for proposing innovative functionalities and qualities, and exploring novel domains.

- We demonstrated that LLMs can reliably assess semantic quality when guided by expert-curated codebooks, often aligning with human agreement levels. This finding has long-term implications for scalability. It enables a shift from time-consuming human reviews and periodic inter-rater reliability checks to automated, comprehensive quality assurance processes that can evolve simply by updating the LLM prompt.
- Although future generations of NLP models will likely improve in reasoning, context retention, and multimodal grounding, the results observed in this study—such as the tension between convergence and creative diversity—are likely to persist. Therefore, our findings should be interpreted not as statements about specific models, but as indicators of how different classes of RE tasks interact with generative automation. Consequently, the role of the requirements engineer is poised to shift from drafting specifications to *orchestrating* the elicitation process. Future analysts will focus on pushing the creative boundaries of the solution space, where models tend to converge, and performing strategic sample checks to validate automated outputs, relying on LLMs to handle the bulk of formalization and structural verification.

IX. Threats to Validity

This section discusses the main threats that could impact the validity of our results [43].

Internal Validity. The outputs of LLMs are inherently non-deterministic and may vary between runs. To mitigate this, we generated 30 independent instances of user stories for each LLM, reducing the impact of random fluctuations in individual outputs. Furthermore, when employing LLMs for the evaluation of user stories, we set the generation temperature at 0, following best practices [42], to ensure deterministic behavior and reproducibility. Regarding qualitative evaluation, given the large scale of the dataset (14,540 user stories), we automated the annotation process using Claude 3 Opus, the best performing model from our first evaluation. This choice could introduce a bias in favor of Claude 3 Opus when it evaluates its own generated user stories. However, it is important to emphasize that the objective of this study is not to declare the absolute best-performing model, but rather to investigate the broader potential and limitations of LLMs in supporting the requirements elicitation process. Although there may be slight biases in the evaluation of specific models, overall findings regarding LLM capabilities and weaknesses remain robust and are not substantially affected by this choice.

Our semantic quality evaluation was mainly applied to individual user stories. However, the coverage experiment directly assesses completeness across sets of stories, thus also capturing set-level properties.

Another potential internal threat to validity concerns the possibility that some LLMs may have been exposed to the *GT US* dataset during training. Although the nature of LLMs inherently reduces the risk of direct memorization of small datasets, since they rely on statistical patterns learned from massive-scale data and typically do not store exact instances of individual samples [44], this risk cannot be entirely excluded. This is particularly relevant given that the case study adopted in our experiment is well known in the requirements engineering community and publicly available. While we have no evidence of verbatim reuse of *GT US* content, prior exposure could have influenced the behavior of the different LLMs in subtle ways. This represents an inherent limitation of studies involving pre-trained LLMs and should be taken into account when interpreting the results.

External Validity. Our study is carried out within the context of a single application domain, that is, a camping management system, which may limit the generalizability of the results to other domains. Requirements in different domains (e.g., finance, healthcare, embedded systems) may exhibit different linguistic structures, abstraction levels, or domain-specific terminology that could affect LLM performance. However, we deliberately selected a realistic and sufficiently complex system with *GT US* as ground truth user stories manually curated by domain experts. This choice ensures that the evaluation is grounded in a concrete and representative scenario. Moreover, the methodology and evaluation pipeline we propose (e.g., coverage, diversity, defect analysis, qualitative scoring) are domain-agnostic and can be reused in future studies to assess generalizability across multiple contexts.

Another external concern regards the choice of comparing LLMs and students performance. We reused the students results available in Ferrari et al. [32] which offers a complete replication package, providing the necessary ground truth for a systematic and reproducible comparison. Recruiting professional analysts at scale is significantly more challenging, while student participation allows controlled experimentation with a sufficient sample size. Importantly, our study also includes a ground truth data set curated by experts (*GT US*), which serves as a benchmark for coverage and completeness, and the semantic evaluation was performed by the authors, all experienced in RE. This combination mitigates the threat of relying solely on students, although we acknowledge that there may be differences with professional analysts.

We must also mention the exclusive focus on user stories as RE artifacts. We chose this specification means because they are one of the most widely adopted artifacts in agile requirements engineering, supported by a rich literature and established evaluation frameworks [30]. user stories are also the starting point of our study, inherited from the work by Ferrari et al. [32]. Compared to free-form natural language requirements, user stories provide a semi-structured format that facilitates controlled evaluation while still reflecting real-world practice. We did not consider formal specification languages in this study, as they require specialized expertise not typically assumed in agile teams and introduce an abstraction layer beyond the scope of our evaluation. Although we expect similar results to hold for other semi-structured artifacts, we acknowledge that the generalization of our findings to different types of RE artifacts requires further empirical study.

We acknowledge that the only expert-authored set of user stories in our study is the ground truth (*GT US*). The datasets

produced by students and LLM instances are comparable in size compared to the *GT US* and serve as a solid basis for our analyses. We plan to extend our study in future work by including larger collections of expert-produced stories.

Finally, a further limitation concerns the rapid evolution of LLMs. While absolute performance levels may change, our analysis focuses on relative strengths and weaknesses across task types and quality dimensions. We therefore expect the results of this study to remain informative even as underlying models improve, though future replications will be necessary.

Construct Validity. Construct validity concerns whether the evaluation metrics and procedures accurately capture the intended concepts. In our study, we rely on well-established frameworks: AQUSA for syntactic quality and the Quality User Story (QUS) framework for qualitative assessment. Although these frameworks are grounded in literature and practice, the application of their criteria, particularly in QUS when using manual and model-based labeling, can be inherently subjective. To reduce interpretation variability, we developed a detailed evaluation codebook that was iteratively refined and reviewed by all authors. A potential threat to construct validity lies in the fact that, although inter-rater agreement among human evaluators was substantial (Table V), it was not perfect. As a result, LLM–human alignment is ultimately constrained by this human baseline. In addition, model performance is sensitive to prompt design and the level of structure provided in the evaluation criteria, meaning that less controlled or differently framed setups could lead to lower agreement than observed in this study.

X. Related Work

A. LLMs in Requirements Engineering

LLMs have been used extensively in software engineering, particularly to support code generation, testing, and maintenance [10]. LLMs can help generate complete software requirements specifications significantly faster [45] and, more generally, can improve the efficiency and accuracy of requirements-related tasks [17]. Despite their potential in supporting requirements engineering activities (e.g., modeling with different formalisms and domains [46], [47], [48], [49] or traceability [50], [51], [52]) and the initial exploration of their effectiveness in addressing software engineering problems [53], [54], LLMs have not been widely applied to support requirements engineering activities.

Previous work has focused on using LLMs to reduce requirements ambiguity and support requirements elicitation. For example, Luitel et al. [18] propose an approach to reduce omissions in software requirements by using BERT [55] to predict missing keywords and concepts. Fantechi et al. [19] measure the performance of ChatGPT in finding inconsistencies in requirements. They notice that LLMs cannot substitute expert judgment but could rather complement manual analysis and speed up human annotation. White et al. [25] present a set of prompt patterns aimed at evaluating completeness (i.e., whether the specification covers relevant system features) and the ambiguity of a requirement specification. Similarly, Lubos et al. [29] use LLMs to detect quality issues in software requirements, such as ambiguity, incompleteness, and other categories defined in ISO 29148, highlighting the ability of LLMs to improve requirements-related quality assurance processes.

Görer et al. [56] use LLMs (ChatGPT and Bard) to generate interview scripts for requirements elicitation. The authors explore an iterative approach in which the LLM is successively presented with initial segments of the interview script, enabling the step-by-step generation of subsequent content. This iterative approach allows the LLM to avoid common mistakes made by analysts during stakeholder interviews (e.g., asking long or irrelevant questions) [34]. Elicitron [21] uses LLMs to simulate a diverse set of user agents and automate requirements elicitation interviews. The approach proposes a context-aware generation method that maximizes the diversity of requirements. It also allows for the creation of user agents that can support the identification of latent needs (i.e., unarticulated and unexpected factors that strongly influence product desirability).

Recent RE 2025 studies show a growing interest in LLM-based elicitation. For example, Shen et al. [24] examine how GPT-4o can automatically generate follow-up interview questions, outperforming human-authored questions in clarity and informativeness. Korn et al. [20] present LLMREI, a chatbot that autonomously conducts requirements interviews and reduces common interviewer errors. Ullrich et al. [57] investigate how developers integrate requirements into LLM-assisted code generation, finding that requirements must be decomposed to be usable in prompts. These works collectively illustrate the emergence of LLMs as active participants in the elicitation process, either as interviewers or as assistants.

Our work differs from these studies in several key respects. We focus specifically on using LLMs to automate the generation of user stories from the IBL process and to assess the semantic quality of the generated stories. To achieve this aim, we use LLMs to emulate the IBL process conducted by students in previous work [32]. Unlike Elicitron [21] or LLMREI [20], we do not explore the ability of LLMs to diversify user agents but instead benchmark multiple LLMs against student-generated datasets using a large-scale replication protocol. This allows us to analyze not only single stories but also set-level properties (coverage and consistency) and to systematically examine the impact of the evaluation framework itself (human annotation quality, inter-rater agreement, and prompt framing) on LLM performance.

Recent work has explored the use of LLMs to bridge the gap between informal natural language requirements and formal specifications. For example, Guidotti et al. [26] introduced an approach that leverages LLMs to translate natural language requirements into Property Specification Patterns, facilitating the formalization process for domain experts without deep expertise in formal methods. Similarly, Cosler et al. [27] developed nl2spec, a framework that interactively translates unstructured natural language into temporal logic specifications using LLMs, enhancing the accessibility of formal verification. Reinpold et al. [58] investigated the potential of LLMs to verify technical system specifications, demonstrating that models such as GPT-4o and Claude 3.5 Sonnet can effectively assess the fulfillment of requirements in the system specifications. A general

overview of the application of LLMs to promote the usability of formal requirements has been presented in [59], which proposes a roadmap for the use of LLMs to assist in understanding and generating formal requirements. This work highlights the promising role of LLMs in making formal requirements engineering more approachable and efficient.

Other work leveraged deep learning models and LLMs to automate the elicitation of software requirements. For example, Gudaparthi et al. [60] generate adversarial examples to identify candidates for novel requirements. They do so by applying small changes (perturbations) to descriptions of original requirements using a deep neural network. The generated requirements are traceable to existing software features, ensuring transparency and facilitating discussions among requirements engineers and developers. Bencheikh and Höglund [23] studied how ChatGPT-generated requirements align with human-written requirements. Humans and ChatGPT were provided with a domain description and requested to generate seven functional and two non-functional requirements, without receiving additional contextual information of how requirements should be specified. ChatGPT demonstrated remarkable time efficiency compared to human participants without significantly compromising the quality of generated requirements. Similarly, Ronanki et al. [28] formulated six questions to elicit requirements using ChatGPT. Using the same six questions, the authors obtained requirements from five RE experts from academia and industry through interview-based surveys. Compared to the requirements generated by RE experts, ChatGPT-generated requirements were more abstract, atomic, consistent, correct, and understandable. Krishna et al. [22] conducted an empirical study comparing GPT-4 and CodeLlama with entry-level engineers to produce software requirements specifications, finding that LLMs can produce SRS drafts of comparable quality to those created by novice engineers, with GPT-4, demonstrating a greater ability to identify and rectify issues within requirements documents.

B. User Stories Evaluation

The Quality User Story (QUS) framework [6] is a collection of 13 criteria that determine the quality of user stories in terms of syntax (well-formed, atomic, minimal), pragmatics (full sentence, estimatable, unique, uniform, independent and complete), and semantics (conceptually sound, problem-oriented, unambiguous, conflict-free). Lucassen et al. [2] also proposed an automated tool (AQUSA) to automatically evaluate the syntactical and pragmatic qualities of user stories using Stanford CoreNLP and the Natural Language Toolkit. The QUS framework and related toolkit have been extensively used in previous work (e.g., [61], [62], [63]) to evaluate the quality of user stories. In contrast, in this paper, we focus on the semantic qualities of user stories and their automated assessment using LLMs, which has not been systematically explored in previous studies. Our large-scale evaluation of multiple LLMs against human-created datasets and our analysis of set-level properties extend the state of the art in user story evaluation.

XI. Conclusion

This study shows that LLMs can support requirements engineering by generating user stories with high syntactic and stylistic quality and reliably assessing their semantic quality when guided by structured criteria. LLM-generated user stories achieved higher coverage than those written by students but showed lower diversity and creativity, limiting their value in exploratory contexts. Although newer models produce more human-like user stories, they still meet acceptance criteria less consistently than those authored by humans. For assessment, several LLMs matched or exceeded inter-human agreement when using a detailed codebook. In general, LLMs offer useful support for semi-automated elicitation and quality assessment, but prompt design and human oversight remain essential. Future work will investigate reasoning models and assess their ability to generate and evaluate user stories. Moreover, we will integrate user stories with acceptance tests, for instance in the form of Behavior-Driven Development scenarios, to better connect requirements quality with validation and testing.

Data Availability Statement

We provide a complete replication package that contains all the code, datasets, our codebook, and evaluation scripts used in this study. The materials are publicly available at: https://doi.org/10.5281/zenodo.15240142.

References

[1] F. Dalpiaz, I. Van Der Schalk, S. Brinkkemper, F. B. Aydemir, and G. Lucassen, "Detecting terminological ambiguity in user stories: Tool and experimentation," *Inf. Softw. Technol.*, vol. 110, pp. 3–16, Jun. 2019.
[2] G. Lucassen, F. Dalpiaz, J. M. E. M. van der Werf, and S. Brinkkemper, "Improving agile requirements: The quality user story framework and tool," *Requirements Eng.*, vol. 21, pp. 383–403, Apr. 2016.
[3] A. R. Amna and G. Poels, "Systematic literature mapping of user story research," *IEEE Access*, vol. 10, pp. 51723–51746, 2022.
[4] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband, "A systematic literature review on agile requirements engineering practices and challenges," *Comput. Human Behav.*, vol. 51, pp. 915–929, Oct. 2015.
[5] R. Kasauli, E. Knauss, J. Horkoff, G. Liebel, and F. G. de Oliveira Neto, "Requirements engineering challenges and practices in large-scale agile system development," *J. Syst. Softw.*, vol. 172, 2021, Art. no. 110851.
[6] G. Lucassen, F. Dalpiaz, J. M. E. Van Der Werf, and S. Brinkkemper, "Forging high-quality user stories: Towards a discipline for agile requirements," in *Proc. Int. Requirements Eng. Conf.*, Piscataway, NJ, USA: IEEE Press, 2015, pp. 126–135.
[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[8] A. Radford et al., "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, p. 9, 2019.
[9] T. Brown et al., "Language models are few-shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 1877–1901.
[10] A. Fan et al., " Large language models for software engineering: Survey and open problems," in *Proc. Int. Conf. Softw. Eng., Future Softw. Eng.*, Piscataway, NJ, USA: IEEE Press, May 2023, pp. 31–53.
[11] X. Du et al., "Evaluating large language models in class-level code generation," in *Proc. Int. Conf. Softw. Eng.*, Piscataway, NJ, USA: IEEE Press, 2024, pp. 865–865.
[12] T. Ahmed, C. Bird, P. Devanbu, and S. Chakraborty, "Studying LLM performance on closed-and open-source data," 2024, *arXiv:2402.15100*.
[13] M. Jin et al., "InferFix: End-to-end program repair with LLMs," in *Proc. Int. Conf. Found. Softw. Eng.*, 2023, pp. 1646–1656.
[14] C. S. Xia, Y. Wei, and L. Zhang, "Automated program repair in the era of large pre-trained language models," in *Proc. Int. Conf. Softw. Eng.*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1482–1494.

[15] Y. Wei, C. S. Xia, and L. Zhang, "Copiloting the copilots: Fusing large language models with completion engines for automated program repair," in *Proc. Int. Conf. Found. Softw. Eng.*, 2023, pp. 172–184.
[16] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, "Using an LLM to help with code understanding," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng.,* Piscataway, NJ, USA: IEEE Press, 2024, pp. 881–881.
[17] C. Arora, J. Grundy, and M. Abdelrazek, "Advancing requirements engineering through generative AI: Assessing the role of LLMs," in *Generative AI for Effective Software Development.* New York, NY, USA: Springer-Verlag, 2024, pp. 129–148.
[18] D. Luitel, S. Hassani, and M. Sabetzadeh, "Improving requirements completeness: Automated assistance through large language models," *Requirements Eng.*, vol. 29, no. 1, pp. 73–95, 2024.
[19] A. Fantechi, S. Gnesi, L. Passaro, and L. Semini, "Inconsistency detection in natural language requirements using ChatGPT: A preliminary evaluation," in *Proc. Int. Requirements Eng. Conf.*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 335–340.
[20] D. Korn, E. Knauss, D. Mendez, and N. Seyff, "LLMREI: Large language model requirements. elicitation interviews," in *Proc. 33rd IEEE Int. Requirements Eng. Conf.*, 2025, pp. 19–30.
[21] M. Ataei, H. Cheong, D. Grandi, Y. Wang, N. Morris, and A. Tessier, "Elicitron: An LLM agent-based simulation framework for design requirements elicitation," 2024, *arXiv:2404.16045*.
[22] M. Krishna, B. Gaur, A. Verma, and P. Jalote, "Using LLMs in software requirements specifications: An empirical evaluation," in *Proc. IEEE 32nd Int. Requirements Eng. Conf.*, 2024, pp. 475–483.
[23] L. Bencheikh and N. Hoglund, "Exploring the efficacy of ChatGPT in generating requirements: An experimental study," Master's thesis, Univ. of Gothenburg/Dept. of Computer Sci. and Eng, 2023.
[24] Y. Shen, A. Singhal, and T. Breaux, "Requirements elicitation follow-up question generation," in *Proc. 33rd IEEE Int. Requirements Eng. Conf.*, 2025, pp. 117–129.
[25] J. White, S. Hays, Q. Fu, J. Spencer-Smith, and D. C. Schmidt, "ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design," in *Generative AI Effective Software Development*, New York, NY, USA: Springer-Verlag, 2024, pp. 71–108.
[26] D. Guidotti, L. Pandolfo, T. Fanni, K. Zedda, and L. Pulina, "Translating requirements in property specification patterns using LLMs," in *Proc. Int. Workshop Artif. Intell. Climate Change*, 2024, vol. 3883, pp. 1–13.
[27] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: Interactively translating unstructured natural language to temporal logics with large language models," in *Proc. Int. Conf. Comput. Aided Verification*, 2023, pp. 383–396.
[28] K. Ronanki, C. Berger, and J. Horkoff, "Investigating ChatGPT's potential to assist in requirements elicitation processes," in *Proc. 49th Euromicro Conf. Softw. Eng. Adv. Appl.*, 2023, pp. 354–361.
[29] S. Lubos et al., "Leveraging LLMs for the quality assurance of software requirements," in *Proc. IEEE Int. Requirements Eng. Conf.*, 2024, pp. 389–397.
[30] J. Medeiros, A. Vasconcelos, C. Silva, and M. Goulão, "Requirements specification for developers in agile projects: Evaluation by two industrial case studies," *Inf. Softw. Technol.*, vol. 117, Jan. 2020, Art. no. 106194. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584919302010
[31] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving agile requirements: the quality user story framework and tool," *Requirements Eng.*, vol. 21, pp. 383–403, Apr. 2016.
[32] A. Ferrari, P. Spoletini, and S. Debnath, "How do requirements evolve during elicitation? An empirical study combining interviews and app store analysis," *Requirements Eng.*, vol. 27, no. 4, pp. 489–519, Dec. 2022, doi: 10.1007/s00766-022-00383-7.
[33] B. Niharika and S. Chopra, "A survey on user stories using NLP for agile development," in *Computer Science Engineering and Emerging Technologies.* Boca Raton, FL, USA: CRC Press, 2024, pp. 592–598.
[34] M. Bano, D. Zowghi, A. Ferrari, P. Spoletini, and B. Donati, "Teaching requirements elicitation interviews: An empirical study of learning from mistakes," *Requirements Eng.*, vol. 24, pp. 259–289, May 2019.
[35] M. Höst, B. Regnell, and C. Wohlin, "Using students as subjects—A comparative study of students and professionals in lead-time impact assessment," *Empirical Softw. Eng.*, vol. 5, no. 3, pp. 201–214, 2000.
[36] F. A., and S. P., "Strategies, benefits and challenges of app store-inspired requirements elicitation," in *Proc. Int. Conf. Softw. Eng.,* Piscataway, NJ, USA: IEEE Press, 2023, pp. 1290–1302.
[37] A. Sharfuddin and T. Breaux, "Generative goal modeling," in *Proc. IEEE 33rd Int. Requirements Eng. Conf. (RE)*, Piscataway, NJ, USA: IEEE Press, 2025, pp. 92–103.
[38] J. White, et al., "A prompt pattern catalog to enhance prompt engineering with ChatGPT," 2023. [Online]. Available: https://arxiv.org/abs/2302.11382
[39] N. F. Liu et al., "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, Dec. 2024.
[40] J. Li, V. Raheja, and D. Kumar, "ContraDoc: Understanding self-contradictions in documents with large language models," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Human Lang. Technol. (Volume 1: Long Papers)*, K. Duh, H. Gomez, and S. Bethard, Eds. ACL, Jun. 2024, pp. 6509–6523.
[41] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, 2012.
[42] M. Renze, "The effect of sampling temperature on problem solving in large language models," in *Proc. Findings Assoc. Comput. Linguistics (EMNLP)*, 2024, pp. 7346–7356.
[43] C. Wohlin, M. Höst, and K. Henningsson, "Empirical research methods in web and software engineering," in *Web Eng.* New York, NY, USA: Springer-Verlag, 2006, pp. 409–430.
[44] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang, "Quantifying memorization across neural language models," in *Proc. 11th Int. Conf. Learn. Representations (ICLR)*, 2023, pp. 1–19.
[45] I. Ozkaya, "Application of large language models to software engineering tasks: Opportunities, risks, and implications," *IEEE Softw.*, vol. 40, no. 3, pp. 4–8, May–Jun. 2023.
[46] B. Chen, et al., "On the use of GPT-4 for creating goal models: An exploratory study," in *Proc. Int. Requirements Eng. Conf. Workshops,* Piscataway, NJ, USA: IEEE Press, 2023, pp. 262–271.
[47] K. Chen, Y. Yang, B. Chen, J. A. H. López, G. Mussbacher, and D. Varró, "Automated domain modeling with large language models: A comparative study," in *Proc. Int. Conf. Model Driven Eng. Lang.s Syst.*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 162–172.
[48] S. de Kinderen and K. Winter, "Towards taming large language models with prompt templates for legal GRL modeling," in *Business Process Modeling, Development, and Support*, New York, NY, USA: Springer-Verlag, 2024, pp. 213–228.
[49] J. Cámara, J. Troya, L. Burgueño, and A. Vallecillo, "On the assessment of generative ai in modeling tasks: An experience report with ChatGPT and UML," *Softw. Syst. Model.*, vol. 22, no. 3, pp. 781–793, 2023.
[50] A. Rodriguez et al., "Prompts matter: Insights and strategies for prompt engineering in automated software traceability," in *Proc. IEEE 31st Int. Requirements Eng. Conf. Workshops (REW)*, 2023, pp. 455–464.
[51] M. North, A. Atapour-Abarghouei, and N. Bencomo, "Code gradients: Towards Automated traceability of LLM-generated code," in *Proc. IEEE 32nd Int. Requirements Eng. Conf. (RE)*, 2024, pp. 321–329.
[52] B. Wei, "Requirements are all you need: From requirements to code with LLMs," in *Proc. IEEE 32nd Int. Requirements Eng. Conf. (RE)*, 2024, pp. 416–422.
[53] A. Hemmat, M. Sharbaf, S. Kolahdouz-Rahimi, K. Lano, and S. Y. Tehrani, "Research directions for using LLM in software requirement engineering: A systematic review," *Front. Comput. Sci.*, vol. 7, Mar. 2025, Art. no. 1519437.
[54] N. Marques, R. R. Silva, and J. Bernardino, "Using ChatGPT in software requirements engineering: A comprehensive review," *Future Internet*, vol. 16, no. 6, p. 180, 2024.
[55] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019, *arXiv:1810.04805*.
[56] B. Görer and F. B. Aydemir, "Generating requirements elicitation interview scripts with large language models," in *Proc. Int. Requirements Eng. Conf. Workshops*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 44–51.
[57] A. Ullrich, D. Winterer, J. Sliwinski, and D. Mendez, "From requirements to code: Understanding LLM integration of requirements in code generation," in *Proc. 33rd IEEE Int. Requirements Eng. Conf.*, 2025, pp. 257–266.

[58] L. M. Reinpold, M. Schieseck, L. P. Wagner, F. Gehlhoff, and A. Fay, "Exploring LLMs for verifying technical system specifications against requirements," 2024, *arXiv:2411.11582*.

[59] A. Ferrari and P. Spoletini, "Formal requirements engineering and large language models: A two-way roadmap," *Inf. Softw. Technol.*, vol. 181, May 2025, Art. no. 107697.

[60] H. Gudaparthi et al., "Prompting creative requirements via traceable and adversarial examples in deep learning," in *Proc. Int. Requirements Eng. Conf.,* Piscataway, NJ, USA: IEEE Press, 2023, pp. 134–145.

[61] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving user story practice with the Grimm method: A multiple case study in the software industry," in *Proc. Requirements Eng., Found. Softw. Qual., Int. Working Conf.*, New York, NY, USA: Springer-Verlag, 2017, pp. 235–252.

[62] Y. Wautelet, D. Gielis, S. Poelmans, and S. Heng, "Evaluating the impact of user stories quality on the ability to understand and structure requirements," in *Proc. 12th IFIP Working Conf Pract. Enterprise Model.*, New York, NY, USA: Springer-Verlag, 2019, pp. 3–19.

[63] J. Wouters, A. Menkveld, S. Brinkkemper, and F. Dalpiaz, "Crowd-based requirements elicitation via pull feedback: Method and case studies," *Requirements Eng*, vol. 27, no. 4, pp. 429–455, 2022.