Towards Automating User Story Classification with Large Language Models Using a Reuse-Oriented Taxonomy

Carlos E. M. de Souza
Federal University of Campina Grande
Campina Grande, PB, Brazil
carlos.souza@virtus.ufcg.edu.br

Mirko Perkusich
Federal University of Campina Grande
Campina Grande, PB, Brazil
mirko@virtus.ufcg.edu.br

Emanuel Filho
Federal Institute of Pernambuco
Jaboatão dos Guararapes, PE, Brazil
emanuel.filho@jaboatao.ifpe.edu.br

Danyllo W. Albuquerque
Federal University of Campina Grande
Campina Grande, PB, Brazil
danyllo.albuquerque@virtus.ufcg.edu.br

Kyller Costa Gorgônio
Federal University of Campina Grande
Campina Grande, PB, Brazil
kyller@virtus.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande
Campina Grande, PB, Brazil
perkusich@virtus.ufcg.edu.br

ABSTRACT

[Context] Agile Software Development (ASD) and reuse strategies are increasingly used to improve software productivity and maintainability. However, while reuse relies on structured and traceable artifacts, ASD often depends on informal elements such as user stories, limiting opportunities for systematic reuse. A recent taxonomy proposes classifying user stories to support traceability and asset reuse, but manual classification remains labor-intensive and error-prone. **[Objective]** This study investigates whether Large Language Models (LLMs) can automate the classification of user stories using a reuse-oriented taxonomy, reducing manual effort while preserving annotation quality. **[Method]** We adopted an explanatory sequential mixed-methods approach. First, a two-step prompting protocol was applied to classify user stories from 12 real-world projects using GPT-4-turbo. Then, we compared model outputs to expert annotations, measuring agreement and qualitatively analyzing disagreements to identify causes and propose corrective actions. **[Results]** The LLM achieved a 48.1% agreement rate with human labels, with project-specific performance ranging from 14.0% to 84.4%. Notably, in 46% of disagreement cases, the LLM's classifications were judged more appropriate than the human label, and in only 25% the human labels were judged to be correct, highlighting inconsistencies in the human annotation process despite prior validation. **[Conclusion]** These initial findings suggest that LLMs can effectively assist in classifying user stories for reuse purposes. Beyond reducing labeling effort, they offer the potential as reviewers in collaborative workflows to improve consistency, transparency, and the overall quality of software artifact organization.

KEYWORDS

Agile Software Development; User Stories Classification; Software Requirements Classification; Large Language Models; Prompt Engineering; Software Reuse.

1 Introduction

Agile Software Development (ASD) has become the dominant paradigm in modern software engineering, emphasizing adaptability, rapid feedback, and incremental delivery [8]. Its lightweight processes, particularly the use of user stories to capture functional goals, have enhanced team collaboration and responsiveness to change [23]. However, this flexibility often comes at the cost of structure: user stories are typically informal, inconsistently phrased, and tied to local project contexts, which limits their potential for systematic reuse, traceability, and integration with other development artifacts [22, 27].

Recent empirical studies highlight this issue, revealing that agile teams frequently manage fragmented and semantically inconsistent requirements data [5, 9, 15]. Without a shared semantic structure, it becomes challenging to link user stories to downstream elements such as implementation tasks, test cases, and architectural decisions [11]. This lack of formalization hinders the development of intelligent tools that rely on structured knowledge for traceability and automation [10].

To address this, researchers have proposed using functional taxonomies to classify user stories into predefined categories, facilitating reuse and supporting semantic enrichment [20, 26]. One such initiative is the Web Information Systems (WIS) taxonomy. Web Information Systems (WIS) refer to web applications that manage and process structured information, and the WIS taxonomy organizes user stories according to reusable, domain-specific (Module, Operation) labels [7]. This taxonomy has been manually applied in previous work to enhance the discoverability and traceability of software development assets.

In parallel, advances in Large Language Models (LLMs) have transformed how developers approach programming tasks, including code completion, code review [2], and Technical Debt management [1]. More recently, these models have also shown promise in earlier stages of the software life cycle—such as requirements structuring, classification, and defect detection—enabling partial automation of traditionally manual and cognitively demanding activities. For instance, a recent review of 395 LLM-focused software engineering studies found that over 20% addressed classification tasks [14]. Models such as BERT and its variants have shown high performance in classifying functional and non-functional requirements [13, 18, 24], especially in zero-shot scenarios. However, the application of LLMs to classify user stories—despite their prevalence in agile practice—remains largely unexplored.

To bridge this gap, this preliminary study investigates whether general-purpose LLMs can perform zero-shot classification of user stories using the WIS taxonomy [7], without domain-specific tuning. We evaluate the performance of this approach using real-world agile project data and compare model-generated classifications to expert annotations. This paper presents a mixed-methods study combining quantitative agreement metrics with a qualitative review of disagreement cases. Our findings provide empirical evidence on the feasibility of integrating LLMs into requirements structuring

workflows, and they highlight both opportunities and challenges for combining taxonomies with generative AI to support intelligent software engineering practices.

The remainder of this paper is structured as follows: Section 2 introduces the WIS taxonomy and summarizes related work. Section 3 details our classification protocol. Section 4 presents and discusses the results. Section 5 points out the implications of our findings. Section 6 outlines threats to validity. Finally, Section 7 concludes the paper and paves the way for future work opportunities.

2 Background and Related Work

This section introduces the functional taxonomy used in our study to guide the classification of user stories and related work on requirements classification with LLMs.

The WIS Taxonomy for User Story Classification. To structure the functional classification of user stories, we adopted the taxonomy proposed by Dilorenzo et al. [7], which was originally designed to support the reuse of development artifacts in the context of WIS. This taxonomy emerged from the systematic analysis of real-world development artifacts—such as issue tracker entries and user stories—collected from over 25 software projects.

The WIS taxonomy provides a domain-specific schema to annotate and organize user stories based on their functionality. It was created to improve semantic traceability, facilitate reuse, and reduce inconsistencies in requirements documentation across teams and projects. The taxonomy is structured along two axes:

(1) **Module:** Defines a high-level functional area of the system. The taxonomy includes three mutually exclusive modules:
   - *Registration:* Encompasses CRUD operations (Create, Read, Update, Delete) related to persistent entities in the system.
   - *Authentication:* Covers identity and access management operations, including login, account creation, password recovery, and permission validation.
   - *Management:* Includes support and administrative features, such as viewing dashboards, exporting reports, and sending notifications.

(2) **Operation:** Specifies the functional action described by the user story. Each operation belongs to one and only one module. For example:
   - The operation "Insert Data" belongs to the *Registration* module.
   - The operation "View Dashboard" belongs to the *Management* module.
   - The operation "Reset Password" belongs to the *Authentication* module.

A single user story may be associated with multiple (Module, Operation) pairs, especially when it spans different functional concerns. This allows for multi-label classification and supports richer semantic annotation.

In the original validation study [7], the taxonomy was applied to label over 1,000 user stories and development tasks. Experts performed annotations manually and followed a structured guideline to ensure consistency. In our study, we reuse this validated dataset and adopt the same labeling scheme to evaluate whether LLMs can accurately replicate or complement human annotations. By integrating this taxonomy into our prompting strategy, we aim to explore its suitability as a scaffold for LLM-based classification, especially in agile contexts where requirements are often informal and unstructured. This approach also allows linking user stories to other software artifacts via semantic labels, enabling more intelligent tooling and enhanced reuse.

Related Work. Classification tasks represent one of the most extensively explored and impactful applications of LLMs in Software Engineering. A recent systematic review of 395 studies found that over 20% addressed classification problems [14]. Within this space, requirements classification has emerged as a central research focus, particularly for its potential to improve early-stage analysis, risk detection, and automation.

Most existing work centers on classifying requirements into functional and non-functional types or identifying fine-grained quality attributes. Approaches such as NoRBERT [13], PRCBERT [18], and FNReq-Net [24] have demonstrated state-of-the-art performance using pre-trained models like BERT, often via fine-tuning or prompt-based strategies. These models achieve F1-scores exceeding 90% on benchmark datasets such as PROMISE and NFR-Review, substantially outperforming traditional machine learning baselines.

For instance, Hey et al. [13] demonstrate that transfer learning with BERT improves robustness across unseen projects. In contrast, Luo et al. [18] apply prompt learning to enable competitive zero-shot performance. Other studies explore hybrid pipelines that combine deep learning with feature selection or data balancing to enhance accuracy and interpretability [19, 24].

Recent research has also investigated classification as a precursor to traceability link recovery. NoRBERT, for example, has been applied to filter irrelevant requirement segments before link generation, improving precision [12]. Similarly, T-BERT [16] leverages knowledge from related tasks, such as code search, to infer traceability links without requiring retraining on each project.

Despite this progress, limited attention has been given to classifying user stories—the primary format for expressing requirements in agile development settings. One notable exception is the work of Alawaji et al. [4], which demonstrates that instruction-tuned LLMs, guided by structured prompts and reusable taxonomies, can outperform traditional fine-tuned models on this task.

Beyond requirements engineering, thematic analysis and content classification studies have shown that LLM performance improves significantly when supported by structured semantic scaffolds. Research by Dai et al. [6], Qiao et al. [21], and Zhang et al. [28] consistently finds that taxonomies, codebooks, and ontologies enhance interpretability, consistency, and quality in human-in-the-loop settings.

These findings underscore two key insights: (i) classification is a mature and high-impact domain for LLMs in Software Engineering, particularly in requirements analysis; and (ii) reusable semantic structures, such as domain-specific taxonomies, enhance the reliability, consistency, and generalizability of LLM-based classifiers. Building on this foundation, our study investigates the zero-shot classification of user stories using a validated functional taxonomy for WIS [7]—a direction that remains underexplored in current literature. Such studies are particularly relevant as they bridge the gap between informal agile artifacts and structured software reuse practices, fostering more intelligent and automated requirements engineering workflows.

3 Research Methodology

To evaluate the ability of LLMs to classify user stories according to a functional reuse-oriented taxonomy, we followed a structured methodology composed of four sequential steps. These steps combine automated classification with human judgment and are designed to support both quantitative evaluation and interpretive

analysis. Figure 1 summarizes the overall process, which includes dataset preparation, prompt-based classification, result comparison, and qualitative review of disagreements. This hybrid approach aims to balance automation with expert insight, enabling us to assess model behavior, detect labeling inconsistencies, and generate actionable recommendations for refinement.

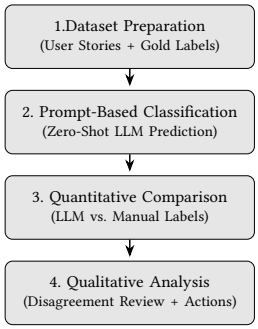

Figure 1: Methodological Workflow.

3.1 Dataset and Annotation Reference

We reused the dataset originally annotated by Dilorenzo et al. [7], comprising 238 user stories and 1099 associated implementation tasks drawn from real-world projects in the domain. Each row in the dataset corresponds to a (User Story, Task) pair, with a human-annotated classification consisting of one (Module, Operation) label from the WIS taxonomy. As such, a single user story may appear multiple times in the dataset, once for each associated task, and may receive different labels across those instances depending on the task context. The dataset is organized as follows:

Structure:

- User stories follow the "As a [role], I want [goal]" format.
- Tasks consist of informal implementation notes extracted from issue trackers or development logs.

Regarding the annotation process:

Gold standard:

- Each (User Story, Task) pair was manually labeled with a (Module, Operation) entry from the WIS taxonomy.
- Annotations were performed by experts, with disagreements resolved through discussion (see Dilorenzo et al. [7]).

The dataset, gold standard annotations, LLM predictions, and prompt templates are available online in our Supplementary Material (See Artifacts Availability Section).

3.2 Classification Protocol and Prompt Design

To simulate a semi-automated annotation-review loop, we designed a two-step prompting protocol with GPT-4-turbo.

Step 1 – Initial Classification. In the first step, the LLM receives the full text of a user story along with all its associated implementation tasks (combined into a single input). It must return one or more (Module, Operation) pairs that describe the functionality addressed by the user story, according to the WIS taxonomy.

Prompt constraints:

- Classifications must be selected from the WIS taxonomy only.
- Multiple classifications are allowed when well justified.
- Format: tabular output with columns [User Story, Module, Operation].
- If classification is not possible, return `n/a`.

Step 2 – LLM-Based Validation. In the second step, since empirically we observed that, in a few instances, the LLM created classifications not in line with the reference taxonomy, the LLM is prompted to validate and revise its output:

- It must ensure all labels conform to the taxonomy.
- It may reclassify items deemed inconsistent or erroneous.
- No external knowledge or inferred categories are permitted.

The full prompt templates are included in our Supplementary Material (See Artifacts Availability Section).

3.3 Quantitative Analysis

We computed the number of classifications generated manually and automatically for each user story by aggregating over the (User Story, Task) pairs in the dataset. That is, if a user story appeared in three rows, it had three manual classifications, which could be redundant. Additionally, the LLM was free to classify the user stories using as many labels as seemed necessary.

Agreement was computed as follows:

- For each user story, we compared the set of (Module, Operation) pairs generated by the LLM against the set of human-annotated pairs.
- A classification was counted as a **match** only when the pair matched exactly.
- We then computed the agreement rate as the proportion of matched classifications over the total number of gold standard labels.

This analysis was performed at the user story level to reflect the model's ability to semantically interpret stories in context, rather than per (User Story, Task) row.

3.4 Qualitative Analysis

Following the quantitative phase, we conducted a manual review of disagreement cases. For each user story where the LLM and the gold standard disagreed, one of the authors examined:

- The full user story text.
- The list of associated tasks.
- The LLM's classification versus the human annotations.

Each disagreement was categorized into one of the following:

- *LLM Error:* clear mistake by the model given the available context.
- *Human Annotation Error:* the gold standard was judged incorrect or inconsistent.
- *Ambiguous Case:* multiple valid interpretations exist; no definitive judgment possible.

As part of the qualitative analysis, each disagreement case was also assigned a recommended follow-up action to guide future iterations of the classification process and dataset refinement. The following categories were used:

- *Fix manual classification:* Human annotation was judged to be incorrect based on LLM evidence.
- *Refine prompt:* The prompt failed to provide sufficient clarity or guidance for accurate classification.
- *Accept justified disagreement:* Both human and LLM classifications were deemed reasonable due to ambiguity in the user story description; disagreement is tolerable.

- *Breakdown the user story:* The user story addressed multiple concerns that hindered classification.
- *Extend taxonomy:* The taxonomy did not support an otherwise valid classification.
- *Fix both classifications:* Both classifications were deemed incorrect.

These recommendations were derived during expert review and are intended to inform prompt engineering, taxonomy evolution, and dataset curation.

4 Results and Discussion

This section presents the results of applying a two-step zero-shot LLM classification protocol to a dataset of 238 user stories and 1099 associated implementation tasks. We report both quantitative agreement metrics and qualitative insights, including suggested actions for improving future classification cycles.

4.1 Quantitative Analysis

We computed agreement rates between LLM-generated classifications and the human-annotated gold standard, aggregated by project. A match was recorded when the same (Module, Operation) pair appeared in both outputs. Table 1 shows the total number of user stories and tasks per project, and the total number of classifications indicated by manual and LLM annotations per project. Further, it presents the results, including false positives (FP), where the LLM predicted a classification not found in the gold standard, and false negatives (FN), where it failed to predict a valid manual label. Total Disagreements (TD) correspond to the sum of FP and FN, and the Agreement Rate (AR) represents the percentage of correctly matched labels relative to the total number of gold standard classifications.

Table 1: LLM vs. Manual Classification: Agreement by Project

Project	Total US	Total Tasks	Classif. Manual	Classif. LLM	FP	FN	TD	AR (%)
P01	2	14	7	9	2	4	6	45.45
P02	12	36	10	10	3	4	7	66.67
P03	38	216	12	16	24	33	57	40.63
P04	35	101	8	8	19	16	35	37.50
P05	21	136	6	7	8	11	19	44.12
P06	45	132	7	11	34	46	80	13.98
P07	8	53	8	12	5	8	13	40.91
P08	19	64	5	4	7	4	11	62.07
P09	12	91	5	6	4	6	10	76.19
P10	12	54	5	4	13	10	23	28.13
P11	12	75	10	8	3	2	5	84.38
P12	22	127	6	7	14	27	41	36.92

Across all projects, the average agreement rate was 48.1%, showing that the LLM successfully reproduced nearly half of the gold standard classifications without any domain-specific tuning. While this suggests promising potential for zero-shot classification, it also reveals important limitations. For instance, projects P09 and P11 achieved over 75% agreement, while others—such as P06 (13.9%) and P10 (28.1%)—showed much weaker alignment.

FPs and FNs offer useful insights into model behavior. Projects with lower agreement—like P03 and P06—presented high volumes of both FPs and FNs, suggesting a systematic mismatch between how the LLM interprets requirements and how the taxonomy is applied. Conversely, some projects with fewer classification pairs (e.g., P01 and P08) exhibited lower disagreement volume but still demonstrated patterns of overgeneralization or under-classification.

To better understand these gaps, we reviewed the three lowest-performing projects:

- **P06** (13.98%) stood out for its extreme ambiguity. It contained 36 classification pairs derived from vague user stories like "Improvements" or "UI Improvements". Moreover, the project involved advanced features—Bayesian network configuration and graph analytics—not fully covered by the WIS taxonomy. These two factors severely affected classification accuracy and alignment.
- P10 (28.13%) also involved Bayesian networks and complex analytics (e.g., US06: "metrics thresholds definition"; US09: "Update CPT"). Additionally, it lacked the standard user story format, making functional intent harder to infer. Generic items like "improvements" and "perform adjustments" offered no meaningful signal for classification.
- P12 (36.92%) had lower disagreement counts but still showed consistent confusion in user stories describing technical enablers rather than end-user functionality. For instance, US21 and US22 focused on creating endpoints for third-party integration—tasks better characterized as platform capabilities or infrastructure work, which fall outside the scope of the WIS taxonomy. Further, these items lacked user-facing goals and were not framed as user stories, contributing to ambiguous or forced classifications. Additionally, the project contained vague placeholders like US13 and US18, both described simply as "feature adjustments", which likely referred to undocumented hotfixes or minor tweaks made under time pressure. These lacked any contextual framing necessary for meaningful classification. Even among user stories that appeared aligned with the taxonomy, ambiguity persisted. For example, US09 ("As a user, I want to edit the *Curriculum Vitae* tags in the developer profile page") was classified manually as "Registration/Change data insertion", while the LLM returned "Registration/Update data". The story lacks the usual "...so that..." justification and contains no acceptance criteria, making it difficult to judge intent. The associated implementation tasks (e.g., enabling front-end editing and adding restrictions) offer limited insight into the underlying functionality. In this case, both classifications are arguably valid, but without further context, neither can be definitively confirmed—highlighting the limitations of both human and LLM-based interpretation when user stories are underspecified or poorly constructed.

These examples illustrate that disagreements often reflect three root causes: (i) poor user story expressiveness, (ii) limited domain coverage in the taxonomy, and (iii) missing functional context (e.g., absence of acceptance criteria or structured format). These observations motivated the in-depth qualitative analysis in the next section.

4.2 Qualitative Analysis of Disagreements

To better understand the classification mismatches between LLM predictions and manual annotations, one of the authors manually reviewed the 307 disagreement cases. Each case was examined to determine which party provided the most appropriate classification, based solely on the information available in the user story and its associated implementation tasks. No assumptions were made about intent or context beyond what was explicitly written.

Figure 2 shows the distribution of disagreement outcomes. The LLM was deemed correct in 142 cases (46%), while the manual label was favored in 78 cases (25%). In 59 cases (19%), both labels were

considered plausible or the user story was ambiguous. Finally, 27 cases (9%) were judged as incorrect by both parties.

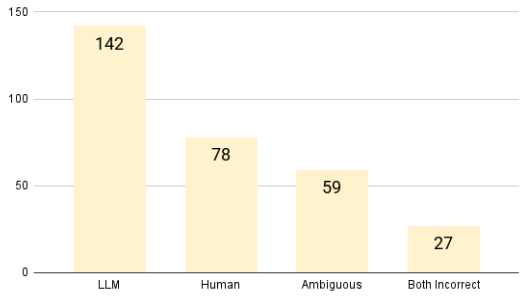

Figure 2: Disagreement between LLM and manual labels.

Importantly, among the 142 cases judged as "LLM-correct", 107 (75%) were attributed to human labeling errors. This is particularly striking given that the gold standard was derived from a previously validated dataset by Dilorenzo et al. [7], produced through a structured annotation and consensus process. These findings highlight a key insight: *even validated datasets created by experts can suffer from annotation drift and inconsistency when applied at scale.*

To further understand the nature of these disagreements and their implications, we analyzed each case and proposed specific actions. Figure 3 summarizes the recommended actions defined for each disagreement. The most frequent recommendation was to fix the manual classification (107 cases), followed by the need to refactor or split user stories (76 cases), often due to vague, overloaded, or incomplete descriptions. Other cases called for prompt refinement (39), taxonomy extension (30), fixing both labels (27), or accepting a justified divergence (28).

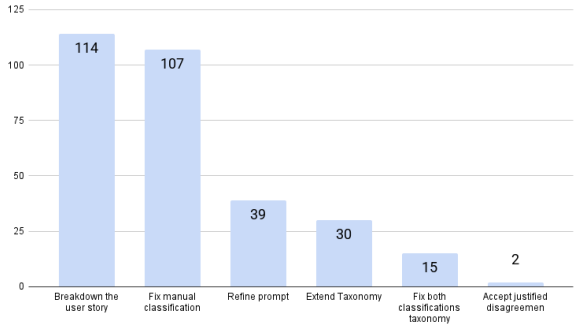

Figure 3: Actions for resolving disagreement cases.

These results underscore a fundamental challenge in large-scale annotation efforts: the inconsistent quality of user stories. Many lacked clear intent or defined acceptance criteria, rendering classification inherently subjective, even for human annotators. This subjectivity often forced reviewers to rely on weak signals to judge correctness, highlighting the importance of investing in user story quality for future reuse and automation scenarios. Efforts such as AQUSA [17] and LLM-based quality assistants [29] may offer promising directions for enforcing clarity and testability in requirement authoring practices.

5 Main Implications

The findings of this study provide actionable insights for both researchers and practitioners working at the intersection of agile requirements engineering and intelligent automation.

Implications for Research. This work contributes to the growing body of literature on the application of LLMs to requirements classification. A recent systematic review of 395 papers on LLMs in Software Engineering found that over 20% focused on classification tasks, mostly on distinguishing functional and non-functional requirements [14]. Yet, user stories, despite being a popular artifact in agile development [3], remain relatively underexplored.

Our results demonstrate that general-purpose LLMs, when guided by structured prompts and a domain-specific taxonomy, can perform zero-shot classification of user stories with moderate agreement (48.1%) with manual annotators. Moreover, our qualitative review revealed that in many cases where the model disagreed with the gold standard, the LLM provided the more appropriate label. This highlights both the potential of LLMs and the inherent limitations of manual annotation at scale—even when performed through a consensus-based process. Based on these findings, we outline the following research directions:

- *Robust Evaluation Frameworks:* Future studies should go beyond accuracy metrics and incorporate qualitative auditing, inter-annotator agreement, and confusion analysis. Mixed-methods approaches—such as the one adopted here—offer richer insights into the nature and causes of disagreement between LLMs and manual annotators.
- *Advancing Prompt and Model Strategies:* Our study employed a two-step zero-shot prompting protocol. More advanced strategies—such as few-shot prompting, instruction tuning, retrieval-augmented generation (RAG), or fine-tuning—should be explored to improve consistency and coverage in classification tasks.
- *Evolving and Inducing Taxonomies:* Our analysis surfaced 30+ disagreement cases that suggested missing categories in the WIS taxonomy [7]. This opens an opportunity for future work on using LLMs to support dynamic taxonomy refinement or to induce new classification schemes tailored to different domains or project types.
- *User Story Authoring Support:* Perhaps the most critical limitation uncovered was the poor quality of many user stories, which often lacked structure, intent, or sufficient detail. Tools like QUS [17] and LLM-based evaluators [29] could be integrated into the authoring process to improve clarity, testability, and downstream reusability, enabling better automation and smarter classification in the long run.
- *Toward Semantically Intelligent Environments:* Enriching user stories with structured labels allows them to serve as semantically typed nodes in knowledge graphs. These graphs can support intelligent reasoning, traceability, and artifact reuse—extending the vision of semantically intelligent software engineering environments. Similar approaches are already being used in domains like Industry 4.0 [26].

Implications for Practice. For agile teams and tool builders, this study provides evidence that LLMs can assist in labeling and organizing user stories by acting as supportive auditing tools that enhance consistency and reduce effort. Key takeaways include:

- *Structured Backlog Management:* Semantically labeled user stories help teams organize backlogs around functional categories, improving discoverability and reuse.

- *Lightweight Traceability:* Although not the focus of this study, prior work suggests that categorized user stories can support traceability to implementation tasks, tests, or design components [12, 16].
- *Tool Integration Potential:* Our prompting strategy could be embedded into requirements management tools to support semi-automated classification, real-time validation, and quality feedback—augmenting human analysts without fully replacing them.

In short, this study positions LLMs not just as classifiers, but as collaborative agents that can assist in both the enrichment and critical review of agile requirements data. It also emphasizes that high-quality user stories remain a foundational requirement for realizing the full potential of automation in agile environments.

6 Threats to Validity

This section discusses potential threats to the validity of this study using the commonly adopted classification [25].

Construct Validity. We treated the WIS taxonomy [7] as the ground truth for classification. While this provides a consistent reference, it may introduce risks if the taxonomy lacks sufficient coverage or expressiveness for some user stories. Further, our prompting strategy was developed based on expert intuition and aimed for clarity and coverage. Nonetheless, alternative prompt designs—such as incorporating few-shot examples, chain-of-thought reasoning, or external contextual cues—might yield different outcomes. Finally, to strengthen the interpretability of our findings and mitigate the risk of misclassifying borderline cases based solely on metric discrepancies, we employed an explanatory sequential mixed-methods design. This approach combines an initial quantitative evaluation with a follow-up qualitative analysis of disagreement cases, enhancing the depth and reliability of our conclusions.

Internal Validity. First, we did not explicitly measure the variability of the model's outputs across repeated runs. As LLMs are inherently probabilistic, different completions may occur even under seemingly deterministic conditions. Second, our evaluation focused exclusively on a single model configuration (GPT-4-turbo) with fixed parameters. Alternative architectures, prompt-tuning strategies, or parameter adjustments might produce different outcomes, and further studies are needed to assess the robustness of our results across configurations. Third, while the dataset encompasses user stories from various real-world projects—enhancing ecological validity—it also introduces variability in terminology, writing style, and detail level, which may act as latent confounding factors and increase classification complexity. Lastly, some degree of disagreement between model outputs and human annotations may arise from the multi-label nature of the task: user stories may relate to multiple (Module, Operation) pairs, and mismatches may reflect differences in interpretation or granularity rather than actual model errors.

External Validity. This study focuses on user stories from the WIS domain, using a single taxonomy [7]. The findings may not generalize to other domains (e.g., embedded or safety-critical systems), languages, or organizational cultures. Further, all user stories were written in Portuguese and sourced from Brazilian projects, which may introduce linguistic or contextual biases. Finally, we also evaluated only one model (GPT-4-turbo), accessed via ChatGPT between April and May 2025. Future versions or alternative models may behave differently. Nonetheless, the findings are consistent with our goal of assessing feasibility in a controlled setting.

Reliability. A single reviewer performed a qualitative analysis of disagreement cases between LLM outputs and gold standard labels. While this mixed-methods approach enhances interpretive depth, the absence of multiple reviewers may introduce subjectivity in how disagreements were evaluated. Additionally, the labeled dataset used originates from a previous manual annotation effort by Dilorenzo et al. [7]. Although their process followed established guidelines, we did not re-validate those annotations, and the original labeling decisions may carry biases that could influence our LLM evaluation. Finally, the use of the ChatGPT interface to access the GPT-4-turbo model introduces reproducibility challenges. The execution environment may be affected by undocumented updates, token window management policies, or other backend behaviors that are not fully transparent or controllable.

7 Conclusion

This study presented a preliminary investigation into the use of LLMs for the zero-shot classification of user stories, leveraging a reuse-oriented functional taxonomy. By applying a structured prompting protocol to a dataset derived from real-world projects, we observed that GPT-4-turbo was capable of reproducing a significant portion of human-provided classifications without prior fine-tuning.

The qualitative analysis of disagreement cases revealed that many mismatches were attributable to inconsistencies in manual annotations, ambiguity in user stories, or representational limitations of the taxonomy, rather than to inherent model deficiencies. In several instances, the LLM's output was considered more semantically appropriate than the corresponding human label. These findings suggest that LLMs, when properly guided, hold promise as supportive tools in agile requirements engineering, particularly for structuring informal artifacts as user stories. Although encouraging, the results must be interpreted as exploratory: the study was limited to a specific taxonomy, domain, and model configuration.

Future work should validate these findings through broader empirical studies involving diverse taxonomies, multiple reviewers, and alternative LLM architectures. Moreover, integration with downstream engineering processes—such as traceability link generation and knowledge graph population—offers a compelling direction for extending the practical utility of LLM-assisted classification frameworks.

ARTIFACT AVAILABILITY

All artifacts related to this study—including prompts, scripts, detailed methodology, results, and evaluation resources—are available to facilitate transparency, replication, and further research [1].

ACKNOWLEDGEMENTS

This work has been partially funded by the project 'iSOP Base: Investigação e desenvolvimento de base arquitetural e tecnológica da Intelligent Sensing Operating Platform (iSOP)' supported by CENTRO DE COMPETÊNCIA EMBRAPII VIRTUS EM HARDWARE INTELIGENTE PARA INDÚSTRIA - VIRTUS-CC, with financial resources from the PPI HardwareBR of the MCTI grant number 055/2023, signed with EMBRAPII.

[1] Available at: https://figshare.com/s/4357bc1e9bf22d63584a

REFERENCES

[1] Danyllo Albuquerque, Everton Guimarães, Graziela Tonin, Pilar Rodríguezs, Mirko Perkusich, Hyggo Almeida, Angelo Perkusich, and Ferdinandy Chagas. 2023. Managing Technical Debt Using Intelligent Techniques - A Systematic Mapping Study. *IEEE Transactions on Software Engineering* 49, 4 (2023), 2202–2220. https://doi.org/10.1109/TSE.2022.3214764

[2] Yonatha Almeida, Danyllo Albuquerque, Emanuel Dantas Filho, Felipe Muniz, Katyusco de Farias Santos, Mirko Perkusich, Hyggo Almeida, and Angelo Perkusich. 2024. AICodeReview: Advancing code quality with AI-enhanced reviews. *SoftwareX* 26 (2024), 101677.

[3] Anis R Amna and Geert Poels. 2022. Systematic literature mapping of user story research. *IEEE access* 10 (2022), 51723–51746.

[4] Bader Alshemaimri Batool Alawaji and, Mona Hakami and. 2024. Evaluating Generative Language Models with Prompt Engineering for Categorizing User Stories to its Sector Domains. In *Proceedings of the IEEE Conference*. https://ieeexplore.ieee.org/document/10544242/

[5] Edna Dias Canedo, Angelica Toffano S Calazans, Geovana Ramos Sousa Silva, Eloisa Toffano Seidel Masson, and Isabel Sofia Brito. 2024. On the Challenges to Documenting Requirements in Agile Software Development: A Practitioners' Perspective. In *Congresso Ibero-Americano em Engenharia de Software (CIbSE)*. SBC, 286–300.

[6] Shih-Chieh Dai, Aiping Xiong, and Lun-Wei Ku. 2023. LLM-in-the-loop: Leveraging Large Language Model for Thematic Analysis. *arXiv preprint arXiv:2310.15100* (2023). https://arxiv.org/abs/2310.15100

[7] Ednaldo Dilorenzo, Emanuel Dantas, Mirko Perkusich, Felipe Ramos, Alexandre Costa, Danyllo Albuquerque, Hyggo Almeida, and Angelo Perkusich. 2020. Enabling the Reuse of Software Development Assets Through a Taxonomy for User Stories. *IEEE Access* 8 (2020), 107285–107300. https://doi.org/10.1109/ACCESS.2020.2996951

[8] Henry Edison, Xiaofeng Wang, and Kieran Conboy. 2021. Comparing methods for large-scale agile software development: A systematic literature review. *IEEE Transactions on Software Engineering* 48, 8 (2021), 2709–2731.

[9] Ahmed Fawzy, Amjed Tahir, Matthias Galster, and Peng Liang. 2025. Exploring data management challenges and solutions in agile software development: a literature review and practitioner survey. *Empirical Software Engineering* 30, 3 (2025), 1–61.

[10] Katharina Großer, Volker Riediger, and Jan Jürjens. 2022. Requirements document relations: A reuse perspective on traceability through standards. *Software and Systems Modeling* 21, 6 (2022), 1–37.

[11] Samedi Heng, Monique Snoeck, and Konstantinos Tsilionis. 2022. Building a Software Architecture out of User Stories and BDD Scenarios: Research Agenda. In *CEUR Workshop Proceedings (CEUR-WS. org)*, Vol. 3134. 40–46.

[12] Tobias Hey, Jan Keim, and Sophie Corallo. 2024. Requirements classification for traceability link recovery. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. IEEE, 155–167.

[13] Tobias Hey, Jan Keim, Anne Koziolek, and Walter F Tichy. 2020. Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th international requirements engineering conference (RE)*. IEEE, 169–179.

[14] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. 2024. Large language models for software engineering: A systematic literature review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024), 1–79.

[15] Rashidah Kasauli, Eric Knauss, Jennifer Horkoff, Grischa Liebel, and Francisco Gomes de Oliveira Neto. 2021. Requirements engineering challenges and practices in large-scale agile system development. *Journal of Systems and Software* 172 (2021), 110851.

[16] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. 2021. Traceability transformed: Generating more accurate links with pre-trained bert models. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 324–335.

[17] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn EM van der Werf, and Sjaak Brinkkemper. 2016. Improving agile requirements: the quality user story framework and tool. *Requirements engineering* 21 (2016), 383–403.

[18] Xianchang Luo, Yinxing Xue, Zhenchang Xing, and Jiamou Sun. 2022. Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*. 1–13.

[19] Barak Or. 2025. Improving Requirements Classification with SMOTE-Tomek Preprocessing. *arXiv preprint arXiv:2501.06491* (2025).

[20] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599.

[21] Tingrui Qiao, Caroline Walker, Chris Cunningham, and Yun Sing Koh. 2025. Thematic-LM: A LLM-based Multi-agent System for Large-scale Thematic Analysis. In *Proceedings of the ACM on Web Conference 2025*. 649–658.

[22] Scarlet Rahy and Julian M Bass. 2022. Managing non-functional requirements in agile software development. *IET software* 16, 1 (2022), 60–72.

[23] Soumya Prakash Rath, Nikunj Kumar Jain, Gunjan Tomar, and Alok Kumar Singh. 2025. A systematic literature review of agile software development projects. *Information and Software Technology* (2025), 107727.

[24] Summra Saleem, Muhammad Nabeel Asim, Ludger Van Elst, and Andreas Dengel. 2023. FNReq-Net: A hybrid computational framework for functional and non-functional requirements classification. *Journal of King Saud University-Computer and Information Sciences* 35, 8 (2023), 101665.

[25] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, Anders Wesslén, et al. 2012. *Experimentation in software engineering*. Vol. 236. Springer.

[26] Yuchen Xia, Zhewen Xiao, Nasser Jazdi, and Michael Weyrich. 2024. Generation of asset administration shell with large language model agents: Towards semantic interoperability in digital twins in the context of industry 4.0. *IEEE Access* (2024).

[27] Asma Yamani, Malak Baslyman, and Moataz Ahmed. 2025. Leveraging LLMs for User Stories in AI Systems: UStAI Dataset. *arXiv preprint arXiv:2504.00513* (2025).

[28] He Zhang, Chuhao Wu, Jingyi Xie, Yao Lyu, Jie Cai, and John M Carroll. 2023. Redefining qualitative analysis in the AI era: Utilizing ChatGPT for efficient thematic analysis. *arXiv preprint arXiv:2309.10771* (2023).

[29] Zheying Zhang, Maruf Rayhan, Tomas Herda, Manuel Goisauf, and Pekka Abrahamsson. 2024. Llm-based agents for automating the enhancement of user story quality: An early report. In *International Conference on Agile Software Development*. Springer Nature Switzerland Cham, 117–126.