

# Can Large Language Models Decompose User Stories into Tasks? Exploring the Role of Prompting Strategies and Models

Delina Ly<sup>✉</sup>

VX Company, Netherlands  
Utrecht University, Netherlands  
dly@vxcompany.com

Fatma Başak Aydemir

Utrecht University  
Netherlands  
f.b.aydemir@uu.nl

Fabiano Dalpiaz

Utrecht University  
Netherlands  
f.dalpiaz@uu.nl

## Abstract

Refining user stories (USs) into tasks is an important step in agile software development, as it translates software requirements into project management artifacts that are assigned to team members. This story-to-task decomposition can be tedious. In this paper, we propose and evaluate DELTA, a Large Language Model (LLM)-based approach that automatically generates tasks from USs and assesses the quality of the resulting decompositions. We conduct a two-phase empirical study to evaluate DELTA. In the first phase, we identify the best-performing DELTA configuration by comparing different prompting strategies, LLM families, and sizes using text similarity metrics on human-created and LLM-generated decompositions. The comparison based on these metrics did not identify a single consistently best-performing configuration. Therefore, in the second phase, we conducted an expert evaluation to determine whether the observed differences in text similarity are reflected in expert judgments and whether DELTA's criteria-based evaluation module, which uses LLM judges, aligns with expert judgments. Our results show that DELTA's evaluation module produces rankings that align more closely with expert judgments than text similarity metrics. Moreover, experts generally prefer the top three LLM-generated decompositions, as ranked by LLM judges, over those created by humans, and they express a willingness to adopt DELTA.

## CCS Concepts

- Software and its engineering → Agile software development;
- Computing methodologies → Artificial intelligence.

## Keywords

User Story Decomposition, Agile Software Development, Requirements Engineering, Large Language Models, LLM-as-a-Judge

## ACM Reference Format:

Delina Ly, Fatma Başak Aydemir, and Fabiano Dalpiaz. 2026. Can Large Language Models Decompose User Stories into Tasks? Exploring the Role of Prompting Strategies and Models. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26)*, October 12–16, 2026, Munich, Germany. ACM, New York, NY, USA, 13 pages. <https://doi.org/10.1145/3832783.3837558>



This work is licensed under a Creative Commons Attribution 4.0 International License. ASE '26, Munich, Germany

© 2026 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-2882-2/2026/10  
<https://doi.org/10.1145/3832783.3837558>

## 1 Introduction

In agile software development, requirements are often captured as user stories (USs) [8, 26] and stored in issue tracking systems (ITS). USs are typically manually decomposed into concrete tasks that define the steps required for their implementation. We refer to this process as *user story decomposition*<sup>1</sup>, which involves identifying, refining, and rewriting tasks. This process bridges requirements engineering and software development by translating requirements into tasks. For example, a US in the Connextra format [8, 26], such as ‘As a student, I want to generate a PDF export of my academic transcripts,’ can be decomposed into three tasks: *i.* retrieve transcript data from the database, *ii.* implement the user interface for requesting and downloading the transcript, and *iii.* implement PDF export functionality. This story-to-task decomposition helps identify missing tasks and supports more reliable effort estimation [8].

The importance of US decomposition increases as software development practices evolve and LLM-based coding assistants and agents such as Copilot and Claude become more widely adopted [19]. USs are typically too abstract to serve as direct inputs for code generation, whereas the derived tasks provide a more suitable level of detail for LLM-assisted software development [42].

Practitioners have proposed many techniques for splitting USs [21], and empirical research shows that only a subset of these techniques recur frequently in practice [28, 30], suggesting that LLMs could learn and apply these patterns to support US decomposition.

Our research goal is to *propose and evaluate* an approach for LLM-based US decomposition and automated quality assessment, *for the purpose of* assessing the quality of US decompositions and evaluating the adequacy of reference-based and criteria-based assessment methods, *from the viewpoint of* software development practitioners, *in the context of* software projects managed in ITS.

To achieve this goal, we propose DELTA (**D**ecomposing USs into tasks via LLM-based **T**ask Assessment), an LLM-based approach that, given a US, generates multiple candidate decompositions and evaluates each using LLM judges against predefined criteria. DELTA then ranks the candidates and returns the top three to practitioners, who select the preferred decomposition. The selected decomposition is incorporated into the ITS to support software development.

We evaluate DELTA through a two-phase empirical study. In *phase one*, we investigate which combination of DELTA parameters (prompting strategies, LLM families and sizes) leads to the highest text similarity between human-created and LLM-generated decompositions, measured with BERTScore, ROUGE-L, and METEOR. For this evaluation, we sample USs and their tasks from two real-world datasets (Section 4.2): a proprietary dataset assembled

<sup>1</sup>Other authors have called this process splitting [10] or disaggregation [8]. We use the term decomposition as it aligns better with standard software engineering terminology.

from projects at a collaborating organization, and a public dataset extracted from an open-source project. Since this comparison did not identify a single configuration that consistently performed best, *phase two* employs a human evaluation with practitioners, who compare human-created tasks with those generated by DELTA for USs from real-world projects. This phase examines whether the differences observed in phase one align with expert judgments, and whether DELTA's criteria-based evaluation module, which relies on LLM judges, produces rankings that align with expert judgments.

This paper makes the following contributions:

- (1) We propose DELTA, an LLM-based approach for decomposing USs into development tasks. US decomposition is the primary contribution, and our evaluation assesses the quality of the resulting decompositions. We release DELTA publicly to support reuse and replication [29].
- (2) We show how prompting strategy (persona-based zero-shot vs. few-shot), LLM family (GPT, Llama, Mistral), and LLM size (small vs large) affect decomposition quality, finding that no single configuration dominates across all conditions.
- (3) We report an expert evaluation from a mid-sized software development organization, whose practitioners compare human-created tasks with those generated by DELTA for USs from real-world projects built with different technologies (.NET, Java, and Power Apps), and we analyze how well DELTA's evaluation module aligns with their judgments.

One striking result of our evaluation is that experts generally prefer the top three LLM-generated US decompositions, as ranked by LLM judges, over human-created ones. They express a willingness to adopt LLM-generated decompositions either as-is or with minor adjustments in their work. Our evaluation reveals that DELTA's evaluation module produces rankings that align more closely with expert judgments than text similarity metrics, making it a better proxy for US decomposition quality.

The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the DELTA approach. Section 4 presents the design of the first phase of our empirical study, and Section 5 reports its results. Section 6 presents the design of the second phase of the empirical study, and Section 7 reports its results. Section 8 discusses the findings and their implications and reviews threats to validity. Section 9 concludes the paper.

## 2 Related Work

*LLMs in Software Engineering (SE).* There are multiple architectures for LLMs, with the main distinction being between encoder-only (e.g., BERT [11]), decoder-only (e.g., GPT, Mistral, Llama), and hybrid [50], all representing types of pre-trained transformers. Surveys demonstrate how much research has been conducted in a short time on LLMs in SE [13, 17]. Recent studies highlight not only opportunities but also challenges, threats, and implications [14, 32, 39]. This paper focuses on decoder-only models that generate responses from natural language prompts for generating tasks from USs.

*Prompting strategies and LLM-as-a-Judge.* Researchers and LLM providers have proposed collections of prompts, with the catalog of prompts for ChatGPT by White *et al.* [46] being the most renowned. We adopt some of these patterns in DELTA's prompts. First, we employ the *persona* pattern, which instructs an LLM to assume a

specific role (e.g., senior software engineer) when responding to the prompt. Second, we explore the effectiveness of including a *few* input-output examples in the prompt. This is the distinction between zero-shot learning (no examples) and few-shot learning (a handful of examples are included) [6].

Zheng *et al.* [53] established the LLM-as-a-Judge paradigm and demonstrated that GPT-4 judgments correlate with human preferences at above 80% agreement across multi-turn dialogue. Building on this paradigm, G-Eval [25] introduced criterion-guided evaluation, where explicit quality criteria are embedded in the judge prompt to improve alignment with human evaluations of natural language generation over ROUGE and BLEU. Wang *et al.* [45] analyzed the ability of LLM-as-a-Judge to assess the outputs of three SE tasks: code translation, code generation, and code summarization. They found a strong correlation with humans in code translation and generation, outperforming conventional automated metrics.

*USs and their automated processing.* USs are the most popular notation for capturing requirements in agile development [8, 26]. They are often stored in ITS such as Jira or Azure DevOps. Researchers have investigated the use of Artificial Intelligence (AI) to support development teams working with USs. Existing surveys summarize the application of natural language processing tools to USs [35]. Another area of interest is model generation from USs. Researchers have investigated the automated generation of domain models [4, 27], business process models [31], goal models [16], and use case scenarios [15]. Generative AI (GenAI) has also been applied to US generation [12], with existing approaches generating USs from requirement documents [36] and interview transcripts [34].

*US splitting and decomposition.* Previous research has primarily focused on splitting large USs into smaller USs, whereas research on the decomposition of USs into tasks remains limited.

Practitioners proposed a catalog of techniques for splitting USs into smaller USs [21], including workflow steps, variations in data, business rule variations, and simple/complex. Dellsén *et al.* [10] found that vertical splitting is the most commonly used technique.

Ly *et al.* [28] investigated their application to US decomposition into implementation tasks and identified the occurrence of only two techniques, specifically horizontal splitting and workflow steps, at VX Company. Müter *et al.* [30] identified common linguistic structures in developer-written task labels resulting from US decomposition, but did not relate them to these techniques.

Recent work has used GenAI to automate US decomposition. Pavlič *et al.* [33] conducted a controlled experiment in which students applied either traditional techniques or GitLab Duo to decompose eight USs of a Jakarta EE system into tasks to implement the USs. While they found that AI-generated decompositions had low accuracy and could serve as a checklist, our evaluation combines automated criteria-based assessment with human expert evaluation, which shows a preference for LLM-generated US decompositions over human-created ones.

## 3 The DELTA Approach

DELTA is an LLM-based approach for decomposing USs into tasks and evaluating the quality of the resulting decompositions. Fig. 1

provides an overview; DELTA includes three modules: *i.* data preparation and context collection, *ii.* US decomposition, and *iii.* automated evaluation. The modules are executed sequentially, with each module's output serving as input to the subsequent module. DELTA can be configured using prompt templates, allowing decomposition instructions and evaluation criteria to be adapted to the context of a specific project or organization.

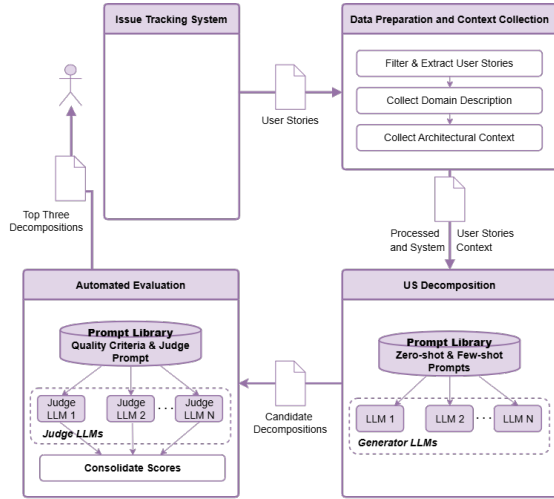


Figure 1: Overview of DELTA.

**Data preparation and context collection module.** DELTA retrieves USs from an ITS and filters them to remove items that do not represent functional USs suitable for decomposition, such as release-related or inactive items. Since labels on work items do not always reflect their actual content [43], each input must be verified to ensure it genuinely qualifies as a US by checking whether it specifies who requests the functionality, what is desired, and, optionally, why. DELTA then extracts the fields relevant to decomposition, limiting the input to reduce the risk of performance degradation from long LLM contexts [24]. Domain description and architectural context are also required inputs, since they guide decomposition toward tasks that reflect the system's design. DELTA incorporates these into its decomposition prompts rather than relying on Retrieval-Augmented Generation, favoring concise contextual descriptions to avoid the overhead of building and maintaining a retrieval pipeline.

**US decomposition module.** This module generates a set of tasks for a given US by prompting an LLM with a template. In the prompt provided to the LLM, we specify a senior software engineer persona, as software engineers are typically responsible for decomposing USs into tasks [8, 28]. DELTA supports an extensible library of prompting strategies, allowing new prompt templates to be added without modifying the core pipeline. In this paper, we provide one zero-shot and one few-shot prompt template, both of which are persona-based. These strategies differ in the amount of contextual guidance provided to the LLM. Each US is decomposed into a single prompt invocation to avoid context interference.

To develop the decomposition prompts, we conducted a preliminary study with real-project data from the collaborating organization, which was excluded from the final evaluation to avoid bias

introduced during prompt development. The process consisted of the following steps:

**1. Prompt initialization.** We started with a senior software engineer persona and a decomposition task instruction, adopting the definition of functional requirements from Jacobson *et al.* [18] as presented in Fig. 2. This definition informs the prompt by framing decomposition as the translation of functional requirements, expressed as input/output behavior, into implementation tasks.

**2. Iterative refinement.** We iteratively refined the prompt by adding, removing, and rephrasing instructions while preserving prompt conciseness. We also explored alternative structures and introduced explicit hard constraints and an internal reasoning step to reduce hallucinated outputs (as represented in Fig. 2). The hard constraints limit the generation of tasks that are not grounded in the US and reduce redundant tasks, while the internal reasoning step guides the model to systematically identify and cover all US requirements before generating the decomposition. We further incorporated domain description and architectural context to prevent the generation of tasks inconsistent with the system architecture.

**3. Evaluation.** After each prompt iteration, we reviewed the LLM-generated tasks by comparing them with the human-created tasks and used our observations to guide subsequent refinements.

**4. Documentation.** We documented each prompt iteration using DELTA's built-in logging feature.

The resulting persona-based zero-shot prompt serves as the baseline. The few-shot prompt augments this baseline with three US decomposition examples drawn from the real-project data used for prompt development. We used three examples as they yielded marginally better results than two in preliminary experiments, confirming the prior findings that an excessive number of few-shot examples can degrade performance [41]. We report the prompts following guidelines for LLM usage in SE research [2, 20]. The prompts and their iterations are available in the replication package [29]. Due to space constraints, Fig. 2 presents the final few-shot prompt template, which extends the zero-shot prompt with examples.

**Automated evaluation module.** This module assesses the quality of US decompositions using the LLM-as-a-Judge paradigm, in which an LLM evaluates candidate decompositions against predefined quality criteria, assigns scores, and ranks them accordingly. This module is useful for identifying the highest-quality decompositions when executing the US decomposition module multiple times, either with the same LLM or with different LLMs. We adopt the senior software engineer persona as used in the US decomposition module for consistency. Each prompt invocation evaluates a single US decomposition to avoid context interference.

We define five quality criteria for US decomposition from the literature [26, 44] and adapt them for task-level decomposition:

**Completeness.** The set of tasks fully covers all activities required to implement the functionality described by the US and its acceptance criteria.

**Relevance to US.** Each task directly contributes to achieving the intended outcome of the originating US.

**Estimable.** Each task is defined clearly enough for the development team to reliably estimate its effort or duration.

**Separation of Concerns.** Tasks represent distinct activities and minimize overlap and dependencies.

---

You are a senior software engineer.

Decompose the following user story into the smallest sufficient set  
 ↪ of tasks that satisfy all explicit functional requirements.  
 A functional requirement specifies an action that a system must be  
 ↪ able to perform, without considering physical constraints; it  
 ↪ specifies input/output behavior of a system.

HARD CONSTRAINTS:

- 1) Generate tasks only for requirements explicitly stated in the  
 ↪ user story or strictly necessary to achieve the described  
 ↪ input/output behavior.
- 2) Avoid redundancy; merge closely related work into a single task.
- 3) If the description adds no meaningful detail beyond the title,  
 ↪ set description to null.

INTERNAL STEP (do not output):

- 1) Identify the explicit functional requirements stated in the user  
 ↪ story (title, description, technical\_impact,  
 ↪ acceptance\_criteria, comment).
- 2) Ensure every requirement is addressed by at least one task.
- 3) Produce the smallest sufficient set of tasks covering all  
 ↪ requirements.

EXAMPLES:

```
{few_shot_examples}
```

ARCHITECTURE CONTEXT:

```
{architectural_context}
```

DOMAIN KNOWLEDGE:

```
{domain_knowledge}
```

USER STORY:

```
{user_story}
```

OUTPUT FORMAT:

Return ONLY a valid JSON object.

IMPORTANT: Escape all special characters in strings.

```
{
  "generated_tasks": [{"title": "string", "description":
    ↪ "string|null"}]
}
```

---

**Figure 2: Few-shot prompt template for US decomposition.**

**Uniformity.** Tasks are formulated using consistent linguistic structures within the set derived from a US.

Following the same iterative development process described for the US decomposition prompt, we refined the judge evaluation prompt until a stable formulation was reached. We explored alternative formulations of the evaluation criteria and different presentations of a 1–5 scoring scale. We incorporated an instruction for the judge to assign a confidence score between 0.0 and 1.0 reflecting its certainty in the evaluation. Fig. 3 presents the final judge prompt template.

To evaluate US decompositions, the automated evaluation module ranks candidates using LLM-based judges. DELTA is extensible and does not assume a fixed number of judges; the number can be adjusted based on evaluation needs. For each decomposition, it computes the aggregated criteria score across judges, excluding the generating LLM to mitigate self-evaluation bias [53]. The module outputs the top three decompositions ranked by this aggregated judgment score, with ties resolved using the average confidence score. These candidate decompositions are presented to the user.

---

You are a senior software engineer.

Evaluate the quality of a user story decomposition into tasks.

1. Review the user story decomposition against these quality  
 ↪ criteria:
  - A. Completeness: do the tasks cover all work needed to fulfill  
 ↪ the user story and its acceptance criteria?
  - B. Relevance to user story: do the tasks directly contribute to  
 ↪ achieving the goal of the user story?
  - C. Estimable: are the tasks defined clearly enough to estimate  
 ↪ effort or time?
  - D. Separation of concerns: are tasks divided into distinct  
 ↪ activities with minimal overlap?
  - E. Uniformity: are the tasks written in a consistent language and  
 ↪ style?
2. Score each criterion independently before assigning a score on a  
 ↪ scale from 1 to 5.
3. Assign a confidence\_score between 0.0 and 1.0 reflecting how  
 ↪ confident you are in the evaluation.

USER STORY:

```
{user_story}
```

TASKS:

```
{tasks}
```

OUTPUT FORMAT:

Return ONLY a valid JSON object.

```
{
  "completeness": 1-5,
  "relevance_to_user_story": 1-5,
  "estimable": 1-5,
  "separation_of_concerns": 1-5,
  "uniformity": 1-5,
  "confidence_score": 0.0-1.0,
}
```

---

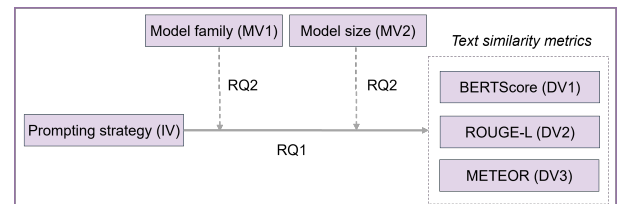
**Figure 3: LLM-as-a-Judge prompt template for evaluation.**

## 4 DELTA Configuration Study

We describe the first phase of our study, which aims to identify the best-performing DELTA configuration by comparing prompting strategies, LLM families, and sizes using text similarity metrics.

### 4.1 Research Goal, Questions and Metrics

Fig. 4 presents the conceptual model for RQ1 and RQ2. RQ1 investigates the relationship between the prompting strategy (independent variable, IV) and the textual similarity (dependent variable, DV) between LLM-generated and human-created US decompositions. Specifically, it compares two commonly used prompting strategies [20]: persona-based zero-shot and few-shot prompting, which differ in the amount of contextual guidance provided to the LLM. RQ2 enriches the analysis by examining the effects of two moderating variables: model family (MV1) and model size (MV2).



**Figure 4: Conceptual model of RQ1 and RQ2.**

**RQ1.** How do persona-based zero- and few-shot prompting strategies affect text similarity between LLM-generated and human-created user story decompositions?

**RQ2.** How do model family and model size affect text similarity between LLM-generated and human-created user story decompositions?

We operationalize text similarity via BERTScore (semantic similarity) [51], ROUGE-L (longest-subsequence lexical overlap) [23], and METEOR (lexical overlap with stemming and synonym matching) [3] as quality proxies to compare LLM-generated and human-created decompositions, consistent with prior SE studies on other generation tasks, such as code summarization [38, 52, 54].

RQ1 and RQ2 establish a reference-based baseline for DELTA’s outputs, indicating which configurations most closely resemble human-created decompositions. The reference decompositions were not generated for this study; software practitioners (primarily software engineers or project managers) produced them during sprint planning and development. They reflect authentic practitioner behavior, including variation in task granularity, naming, and completeness observed in real-world ITS data.

## 4.2 Datasets and Sampling Strategy

We conducted our study on two datasets: (1) a proprietary dataset and (2) an open-source dataset to evaluate DELTA across USs and their tasks that may reflect different decomposition practices.

*Proprietary dataset.* The dataset was taken from the ITS of VX Company, a Dutch organization with circa 200 employees that develops and maintains software solutions. We employed convenience sampling, as the first author is employed at the organization. The company’s development activities are organized around .NET, Java, and Power Apps. Each technology is managed by a core team, following an industry-adapted Agile development method. For each sprint, members are drawn from these teams based on availability and expertise required to implement USs, resulting in varying team compositions. USs and their tasks are managed in Azure DevOps. USs and tasks may be written in Dutch, English, or both. The organization selected nine systems for analysis. One system was reserved for the preliminary study to develop and tune the DELTA prompts and was excluded from the main evaluation to avoid bias. For the remaining eight systems, we extracted completed USs and their tasks from past sprints stored in the organization’s ITS.

*Open-source dataset.* The dataset was collected from an open-source GitHub repository. We selected Fleet<sup>2</sup> because its team members have consistently documented USs with their tasks using [User Story] and [Task] tags, and the repository contains a large set of completed USs, enabling stratified sampling. Fleet provides software to secure and maintain computing devices and is developed in Go and TypeScript. The USs and tasks are written in English.

For both datasets, we evaluated a stratified sample to limit the computational costs and energy consumption of LLM inference [1]. We categorized USs by field count (as shown in Table 1) into partial (1–2 fields) or detailed (3–5 fields), then grouped each category by task count (low, medium, or high, based on the 33rd and 67th percentiles), yielding six strata. We sampled one US per stratum per system for the proprietary dataset (8 systems  $\times$  6 strata = 48

USs) and eight USs per stratum for the single-system open-source dataset (48 USs), using a fixed random seed (42) for reproducibility.

**Table 1: Extracted US and task fields from ITS.**

| Field               | Description                                                                                                                        |
|---------------------|------------------------------------------------------------------------------------------------------------------------------------|
| Title               | Short textual summary of the work item.                                                                                            |
| Description         | Detailed explanation of the work item (optional).                                                                                  |
| Acceptance Criteria | Conditions that must be satisfied for US completion (optional; US only).                                                           |
| Technical Impact    | Expected effects of implementing the US, such as affected components, dependencies, and architectural changes (optional; US only). |
| Comments            | Discussion thread on the work item (optional).                                                                                     |

*Input data.* For the samples, we extracted fields listed in Table 1. The US-verification step was performed manually in the current implementation, with automation planned for future versions.

*Domain description and architectural context.* Domain description was limited to three sentences for conciseness. Architectural context was captured using a template covering frontend, backend, infrastructure, and external integrations. Both were manually validated against existing documentation and repositories.

For the proprietary dataset, we collected domain descriptions and architectural contexts from software engineers with 20, 26, 31, and 38 years of experience (YoE) involved in the development of the evaluated systems. For the open-source dataset, we derived the domain description and architectural context from project documentation and source code.

*Few-shot example selection.* For the proprietary dataset, we selected three representative US decompositions for each production system that reflect the system, technology, and development practices of the teams. For the open-source dataset, we randomly selected three examples from the held-out subset, as project-specific context was not sufficient to identify representative examples.

## 4.3 LLM Selection and Configuration

Due to data sensitivity and compliance requirements, we restricted our experiments to organization-approved models deployable in controlled environments (locally or via Azure AI). We selected representative models from three LLM families: proprietary GPT models, open-weight Llama models, and Mistral models with mixed availability. All selected models are widely adopted, offered by established providers, and have demonstrated promising performance in the preliminary study. For each model family, we consider two variants: *i.* the most capable model accessible within our resource constraints (hereafter referred to as *large*) and *ii.* a less resource-intensive counterpart (hereafter referred to as *small*). Table 2 summarizes the selected models and their size configurations.

For locally hosted models (Llama 3.1 8B and Mistral Small), the decoding temperature is set to zero to maximize determinism [37]. For each dataset, USs are decomposed using 12 LLM configurations (3 families  $\times$  2 sizes  $\times$  2 prompting strategies), resulting in 576 experiments (48 USs  $\times$  12 configurations). Each decomposition, including human-created ones, is evaluated by six models (624 decompositions  $\times$  6 models), yielding 3,744 experiments.

<sup>2</sup><https://github.com/fleetdm/fleet>

**Table 2: Selected LLMs and model sizes.**

| Model           | Family  | Provider | Availability | Size  |
|-----------------|---------|----------|--------------|-------|
| GPT-5.2         | GPT     | OpenAI   | Proprietary  | Small |
| GPT-5 Pro       |         |          |              | Large |
| Llama 3.1 8B    | Llama   | Meta     | Open-weight  | Small |
| Llama 3.3 70B   |         |          |              | Large |
| Mistral Small   | Mistral | Mistral  | Mixed        | Small |
| Mistral Large 3 |         |          |              | Large |

#### 4.4 Statistical Analysis

We apply statistical procedures tailored to each RQ, based on the data’s distributional properties and the nature of the comparisons. For each dataset, we compare LLM-generated and human-created tasks using text similarity metrics. For each generated task, we compute its similarity to all human-created tasks and retain the highest-scoring pairs. We first assess the normality of BERTScore, ROUGE-L, and METEOR score distributions using the Shapiro–Wilk test. For BERTScore and ROUGE-L, we report the F1 variant, as it balances precision and recall into a single similarity measure. All three metrics deviate significantly from normality ( $p < 0.05$ ). The complete results are available in the replication package [29].

**RQ1.** Given that all three metrics deviate from normality, we use the Wilcoxon signed-rank test [47] to compare zero-shot and few-shot prompting strategies. This non-parametric test is suitable for paired, non-normally distributed data. As three tests are conducted for each dataset, the probability of obtaining false positives increases due to multiple comparisons. We therefore apply a Bonferroni correction to control the family-wise Type I error rate, yielding an adjusted significance threshold of  $\alpha = 0.017$  (0.05/3).

**RQ2.** We apply the Aligned Rank Transform ANOVA (ART ANOVA) [48] to examine the effects of model family, model size, and their interaction on US decomposition quality across the three text similarity metrics. For each prompting strategy (zero-shot and few-shot, analyzed separately), we use a  $3 \times 2$  repeated-measures factorial design with model family (GPT, Llama, Mistral) and model size (small, large) as within-subject factors. The same 48 USs are evaluated under all conditions (288 outputs per strategy), with US modeled as a repeated factor to account for within-story dependence. To control the family-wise Type I error rate across the three metrics, we apply a Bonferroni correction ( $\alpha = 0.017$  (0.05/3)).

### 5 DELTA Configuration Study Results

We report the mean ( $\bar{x}$ ), standard deviation ( $\sigma$ ), and the proportion of tasks with descriptions ( $p_{\text{desc}}$ ) for the LLM-generated and human-created US decompositions across the sampled  $N = 48$  USs for both the proprietary and open-source datasets in Table 3. We provide statistics for US fields and task titles in the replication package [29].

Table 3 shows that zero-shot prompting yields a higher proportion of tasks with descriptions than few-shot prompting for GPT-5.2, GPT-5 Pro, Llama 3.3 70B across both datasets, and additionally for Mistral Large 3 on the proprietary dataset. Few-shot prompting reduces the mean number of tasks per US across all LLMs and the standard deviation in five of the six models, with the exception of Mistral Small for the proprietary dataset and Llama 3.1 8B for

the open-source dataset. This suggests that the few-shot examples guide LLMs toward the level of decomposition demonstrated by those examples, whereas zero-shot prompting provides no such guidance, resulting in decompositions with more generated tasks.

Llama and Mistral models generally produce more tasks per US than GPT models, with Mistral Large 3 producing the most tasks across both datasets. Across all LLMs, the open-source dataset generally yields a larger number of generated tasks per US than the proprietary dataset. In contrast, humans produce more tasks for the proprietary dataset than for the open-source dataset and show the highest standard deviation in the number of tasks.

**Table 3: Task statistics by prompting strategy and dataset.**

| (a) Zero-shot prompting |             |          |                   |             |          |                   |
|-------------------------|-------------|----------|-------------------|-------------|----------|-------------------|
| Generator               | Proprietary |          |                   | Open-source |          |                   |
|                         | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ |
| GPT-5.2                 | 3.23        | 2.22     | 83.23             | 6.65        | 3.40     | 97.81             |
| GPT-5 Pro               | 1.96        | 0.99     | 85.11             | 3.00        | 1.58     | 99.31             |
| Llama 3.1 8B            | 5.83        | 3.75     | 14.29             | 8.88        | 3.72     | 18.31             |
| Llama 3.3 70B           | 4.46        | 2.64     | 75.23             | 8.02        | 2.81     | 99.48             |
| Mistral Small           | 4.81        | 3.44     | 28.14             | 6.75        | 3.62     | 33.95             |
| Mistral Large 3         | 6.06        | 3.19     | 100.00            | 10.19       | 3.30     | 99.39             |

| (b) Few-shot prompting |             |          |                   |             |          |                   |
|------------------------|-------------|----------|-------------------|-------------|----------|-------------------|
| Generator              | Proprietary |          |                   | Open-source |          |                   |
|                        | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ |
| GPT-5.2                | 2.81        | 2.09     | 61.48             | 5.44        | 2.58     | 84.29             |
| GPT-5 Pro              | 1.69        | 0.85     | 51.85             | 2.31        | 0.75     | 93.69             |
| Llama 3.1 8B           | 3.73        | 2.24     | 18.44             | 6.83        | 4.25     | 82.01             |
| Llama 3.3 70B          | 3.00        | 1.68     | 21.53             | 5.56        | 1.80     | 51.31             |
| Mistral Small          | 3.46        | 3.74     | 37.95             | 6.56        | 3.43     | 90.79             |
| Mistral Large 3        | 3.77        | 1.56     | 86.74             | 6.96        | 1.98     | 99.70             |

| (c) Human |             |          |                   |             |          |                   |
|-----------|-------------|----------|-------------------|-------------|----------|-------------------|
| Generator | Proprietary |          |                   | Open-source |          |                   |
|           | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ | $\bar{x}$   | $\sigma$ | $p_{\text{desc}}$ |
| Human     | 5.40        | 5.55     | 11.97             | 4.71        | 5.04     | 44.69             |

In the following subsections, we highlight key findings with ★.

#### 5.1 Effect of Prompting Strategies on Human–LLM Text Similarity (RQ1)

Tables 4 and 5 compare the text similarity metrics (DV: BERTScore F1, ROUGE-L F1, and METEOR) for zero-shot and few-shot prompting (IV) based on the Wilcoxon signed-rank test. The complete results are available in the replication package [29]. Significance is assessed using a Bonferroni-corrected threshold ( $p < 0.017$ ). Significant results are highlighted in blue or orange, where blue indicates that few-shot outperforms zero-shot prompting, and orange indicates the opposite. Darker shades indicate larger effect sizes. The highest score for each metric is marked with an asterisk (\*).

**Table 4: Proprietary dataset: Text similarity metrics for US decompositions across prompting strategies and models.**

| Prompt         | Model           | BERTScore (DV1) |          | ROUGE-L (DV2) |          | METEOR (DV3) |          |
|----------------|-----------------|-----------------|----------|---------------|----------|--------------|----------|
|                |                 | $\bar{x}$       | $\sigma$ | $\bar{x}$     | $\sigma$ | $\bar{x}$    | $\sigma$ |
| Zero-shot (IV) | GPT-5.2         | 0.676           | 0.045    | 0.106         | 0.142    | 0.104        | 0.161    |
|                | GPT-5 Pro       | 0.666           | 0.039    | 0.091         | 0.124    | 0.081        | 0.109    |
|                | Llama 3.1 8B    | 0.684           | 0.043    | 0.100         | 0.147    | 0.086        | 0.132    |
|                | Llama 3.3 70B   | 0.692           | 0.047    | 0.111         | 0.141    | 0.101        | 0.141    |
|                | Mistral Small   | 0.688           | 0.032    | 0.074         | 0.112    | 0.063        | 0.090    |
|                | Mistral Large 3 | 0.681           | 0.041    | 0.093         | 0.120    | 0.083        | 0.127    |
| Few-shot (IV)  | GPT-5.2         | 0.680           | 0.045    | 0.113         | 0.136    | 0.129        | 0.172    |
|                | GPT-5 Pro       | 0.701           | 0.093    | 0.197         | 0.250    | 0.175*       | 0.189    |
|                | Llama 3.1 8B    | 0.725*          | 0.096    | 0.218*        | 0.282    | 0.158        | 0.175    |
|                | Llama 3.3 70B   | 0.708           | 0.063    | 0.171         | 0.196    | 0.146        | 0.186    |
|                | Mistral Small   | 0.718           | 0.068    | 0.176         | 0.205    | 0.128        | 0.136    |
|                | Mistral Large 3 | 0.701           | 0.055    | 0.158         | 0.168    | 0.137        | 0.153    |

**Table 5: Open-source dataset: Text similarity metrics for US decompositions across prompting strategies and models.**

| Prompt         | Model           | BERTScore (DV1) |          | ROUGE-L (DV2) |          | METEOR (DV3) |          |
|----------------|-----------------|-----------------|----------|---------------|----------|--------------|----------|
|                |                 | $\bar{x}$       | $\sigma$ | $\bar{x}$     | $\sigma$ | $\bar{x}$    | $\sigma$ |
| Zero-shot (IV) | GPT-5.2         | 0.721           | 0.037    | 0.220         | 0.094    | 0.202        | 0.117    |
|                | GPT-5 Pro       | 0.725           | 0.034    | 0.229         | 0.103    | 0.236        | 0.128    |
|                | Llama 3.1 8B    | 0.715           | 0.036    | 0.195         | 0.087    | 0.151        | 0.083    |
|                | Llama 3.3 70B   | 0.733           | 0.035    | 0.255         | 0.102    | 0.189        | 0.105    |
|                | Mistral Small   | 0.719           | 0.037    | 0.230         | 0.109    | 0.172        | 0.121    |
|                | Mistral Large 3 | 0.729           | 0.031    | 0.235         | 0.089    | 0.168        | 0.089    |
| Few-shot (IV)  | GPT-5.2         | 0.735           | 0.038    | 0.250         | 0.104    | 0.253*       | 0.129    |
|                | GPT-5 Pro       | 0.742*          | 0.043    | 0.283         | 0.127    | 0.267        | 0.149    |
|                | Llama 3.1 8B    | 0.710           | 0.035    | 0.171         | 0.091    | 0.122        | 0.092    |
|                | Llama 3.3 70B   | 0.741           | 0.039    | 0.298*        | 0.136    | 0.226        | 0.129    |
|                | Mistral Small   | 0.707           | 0.029    | 0.190         | 0.108    | 0.127        | 0.097    |
|                | Mistral Large 3 | 0.742*          | 0.037    | 0.259         | 0.117    | 0.230        | 0.144    |

**Finding 1 ★:** Few-shot prompting generally improves text similarity on the proprietary dataset, whereas improvements vary across LLMs on the open-source dataset.

On the proprietary dataset, GPT-5 Pro, Mistral Small, and Mistral Large 3 show significant improvements across all metrics under few-shot prompting. Llama 3.1 8B improves on BERTScore and ROUGE-L, while Llama 3.3 70B improves only on ROUGE-L. GPT-5.2 shows no statistically significant differences across metrics. Effect sizes ranged from medium to large ( $r = 0.498$ – $0.807$ ).

On the open-source dataset, GPT-5.2 shows significant improvements across all metrics under few-shot prompting. GPT-5 Pro improves on BERTScore and ROUGE-L, while Llama 3.3 70B improves on ROUGE-L and Mistral Large 3 improves on BERTScore and METEOR. Llama 3.1 8B shows a significant decrease in METEOR, indicating that zero-shot outperforms few-shot prompting, whereas Mistral Small shows no statistically significant differences across metrics. Effect sizes were medium to large ( $|r| = 0.451$ – $0.757$ ).

The open-source dataset contains longer US titles (mean 8 words vs. 6 words), descriptions (mean 678 vs. 120 words), more comments per US (mean 10.8 vs. 1.5), and longer comments (mean 68 words vs. 41 words) than the proprietary dataset, producing generally larger decompositions. Moreover, text similarity scores under zero-shot prompting were consistently higher on the open-source dataset, indicating greater similarity between generated and human-created decompositions. The richer information available

in the open-source dataset may explain why few-shot prompting led to more variable improvements, as DELTA could make use of this information even without decomposition examples.

Finding 1 indicates that few-shot prompting generally improves text similarity compared to zero-shot prompting, but the effects differ across the datasets.

## 5.2 Effect of Model Family and Size on Human-LLM Text Similarity (RQ2)

We present the results of the ART ANOVA analysis separately for zero-shot and few-shot outputs to avoid confounding the prompting strategy with the moderating variables: model family (MV1) and size (MV2). Each prompting strategy comprises 288 US decomposition outputs per dataset: the same 48 USs  $\times$  3 model families (GPT, Llama, and Mistral)  $\times$  2 model sizes (small and large). We present strength ( $F$ ), statistical significance ( $p$ ), effect size ( $\eta_p^2$ ), and ( $r_{rb}$ ). Significance is assessed using a Bonferroni-corrected threshold ( $p < 0.017$ ). Table 6 summarizes the main effects and interaction of model family and size across text similarity metrics. The statistically significant effects are indicated by "Yes", which is highlighted in blue, whereas non-significant effects are indicated with "No".

**Table 6: Summary of significant main and interaction effects of model family and model size on text similarity metrics.**

| Dataset     | Prompting strategy | Effect of                        | BERTScore (DV1) | ROUGE-L (DV2) | METEOR (DV3) |
|-------------|--------------------|----------------------------------|-----------------|---------------|--------------|
| Proprietary | Zero-shot (IV)     | Model family (MV1)               | Yes             | No            | No           |
|             |                    | Model size (MV2)                 | No              | No            | No           |
|             |                    | Family (MV1) $\times$ Size (MV2) | Yes             | No            | No           |
|             | Few-shot (IV)      | Model family (MV1)               | Yes             | No            | No           |
|             |                    | Model size (MV2)                 | No              | No            | No           |
|             |                    | Family (MV1) $\times$ Size (MV2) | No              | No            | No           |
| Open-source | Zero-shot (IV)     | Model family (MV1)               | No              | No            | Yes          |
|             |                    | Model size (MV2)                 | Yes             | Yes           | Yes          |
|             |                    | Family (MV1) $\times$ Size (MV2) | No              | Yes           | No           |
|             | Few-shot (IV)      | Model family (MV1)               | Yes             | Yes           | Yes          |
|             |                    | Model size (MV2)                 | Yes             | Yes           | Yes          |
|             |                    | Family (MV1) $\times$ Size (MV2) | Yes             | Yes           | Yes          |

**Finding 2 ★:** Model family significantly affects text similarity metrics in 2 of the 6 evaluated cases for the proprietary dataset and in 4 of the 6 evaluated cases for the open-source dataset.

On the proprietary dataset, model family significantly affects BERTScore in zero-shot ( $F = 19.658$ ,  $p = 1.28 \times 10^{-8}$ ,  $\eta_p^2 = 0.143$ ) and few-shot prompting ( $F = 17.345$ ,  $p = 9.41 \times 10^{-8}$ ,  $\eta_p^2 = 0.129$ ). Post hoc comparisons show that Llama and Mistral achieve significantly higher BERTScore than GPT ( $p < .001$ ,  $|r_{rb}| = 0.549$ – $0.762$ ), with no significant difference between Llama and Mistral ( $p = 0.511$ – $0.560$ ). These results suggest that Llama and Mistral produce outputs that are more semantically aligned with human-created tasks than GPT.

On the open-source dataset, model family significantly affects METEOR under zero-shot prompting ( $F = 12.546$ ,  $p = 6.66 \times 10^{-6}$ ,  $\eta_p^2 = 0.096$ ). Post hoc comparisons show that GPT outperforms Llama and Mistral ( $p < .001$ ,  $r_{rb} = 0.648$ – $0.672$ ). Under few-shot

prompting, model family significantly affects all metrics: BERTScore ( $F = 7.760$ ,  $p = 5.45 \times 10^{-4}$ ,  $\eta_p^2 = 0.062$ ), ROUGE-L ( $F = 8.786$ ,  $p = 2.09 \times 10^{-4}$ ,  $\eta_p^2 = 0.070$ ), and METEOR ( $F = 32.411$ ,  $p = 3.71 \times 10^{-13}$ ,  $\eta_p^2 = 0.216$ ). Post hoc comparisons show that GPT significantly outperforms Llama and Mistral across all three metrics ( $p \leq .001$ ,  $r_{rb} = 0.374$ – $0.594$ ), with no significant difference between Llama and Mistral ( $p = 0.505$ ). These results suggest that GPT produces outputs that are more closely aligned with human-created tasks than Llama and Mistral, particularly with few-shot prompting.

**Finding 3 ★:** Model size does not significantly affect text similarity metrics for the proprietary dataset, whereas it significantly affects all 6 evaluated cases for the open-source dataset.

On the proprietary dataset, the main effect of model size is not statistically significant (e.g., BERTScore zero-shot:  $F = 1.956$ ,  $p = 0.163$ ,  $\eta_p^2 = 0.008$ ; few-shot:  $F = 1.767$ ,  $p = 0.185$ ,  $\eta_p^2 = 0.007$ ). Similarly, ROUGE-L and METEOR show no significant size-related differences. The very small effect sizes indicate that model size explains negligible variance in performance, suggesting that increasing model size within the same model family does not meaningfully improve similarity between generated and human-created US decompositions.

On the open-source dataset, model size significantly affects all metrics under both zero-shot prompting (BERTScore:  $F = 24.002$ ,  $p = 1.79 \times 10^{-6}$ ,  $\eta_p^2 = 0.093$ ; ROUGE-L:  $F = 13.163$ ,  $p = 3.50 \times 10^{-4}$ ,  $\eta_p^2 = 0.053$ ; METEOR:  $F = 7.771$ ,  $p = 5.74 \times 10^{-3}$ ,  $\eta_p^2 = 0.032$ ) and few-shot prompting (BERTScore:  $F = 61.011$ ,  $p = 1.88 \times 10^{-13}$ ,  $\eta_p^2 = 0.206$ ; ROUGE-L:  $F = 69.544$ ,  $p = 6.34 \times 10^{-15}$ ,  $\eta_p^2 = 0.228$ ; METEOR:  $F = 55.677$ ,  $p = 1.65 \times 10^{-12}$ ,  $\eta_p^2 = 0.192$ ). These results suggest that larger models generally produce decompositions that are more closely aligned with the human-created decompositions.

**Finding 4 :** The effects of model family and model size are largely independent for the proprietary dataset (1/6 cases), whereas they interact for the open-source dataset: GPT’s advantage over Llama and Mistral is primarily observed among small models (4/6 cases).

For the proprietary dataset, the interaction between model family and model size is significant for BERTScore in the zero-shot setting ( $F = 5.374$ ,  $p = 0.005$ ,  $\eta_p^2 = 0.044$ ). Post hoc comparisons within each model size reveal that, among small models, Mistral outperforms GPT ( $p < 0.001$ ,  $|r_{rb}| = 0.434$ ). Among large models, Llama and Mistral outperform GPT (both  $p < 0.01$ ), with large effect sizes (Llama vs GPT:  $|r_{rb}| = 0.759$ ; Mistral vs GPT:  $|r_{rb}| = 0.629$ ). Llama significantly outperforms Mistral ( $p = 0.012$ ,  $|r_{rb}| = 0.449$ ).

For the open-source dataset, the interaction between model family and size is significant for ROUGE-L in the zero-shot setting ( $F = 6.090$ ,  $p = 2.64 \times 10^{-3}$ ,  $\eta_p^2 = 0.049$ ) and for BERTScore ( $F = 9.161$ ,  $p = 1.48 \times 10^{-4}$ ,  $\eta_p^2 = 0.072$ ), ROUGE-L ( $F = 7.994$ ,  $p = 4.38 \times 10^{-4}$ ,  $\eta_p^2 = 0.064$ ), and METEOR ( $F = 8.216$ ,  $p = 3.56 \times 10^{-4}$ ,  $\eta_p^2 = 0.065$ ) under few-shot prompting. Post hoc comparisons revealed that small Mistral models achieved higher ROUGE-L scores than small Llama models ( $p = 5.60 \times 10^{-3}$ ,  $|r_{rb}| = 0.366$ ). Under few-shot prompting, GPT significantly outperforms Llama and Mistral among the small models across BERTScore ( $p \leq 3.88 \times 10^{-6}$ ,  $r_{rb} = 0.808$ – $0.821$ ), ROUGE-L ( $p \leq 3.59 \times 10^{-5}$ ,  $r_{rb} = 0.539$ – $0.748$ ),

and METEOR ( $p \leq 9.31 \times 10^{-15}$ ,  $r_{rb} = 0.881$ – $0.947$ ), whereas no significant differences are observed for the large models.

Findings 1–4 show that the effects of prompting strategy, model family, and model size on text similarity vary across the proprietary and open-source datasets, suggesting that no single configuration consistently performed best based on text similarity metrics.

Beyond the absence of a consistently best-performing configuration, Findings 1–4 raise a more fundamental question: whether text similarity to human-created decompositions is an appropriate proxy for decomposition quality. These metrics assume that human-created tasks constitute a reliable gold standard, yet a US can be decomposed in multiple valid ways that differ in granularity, naming, and structure while remaining equally correct. Consequently, the best DELTA configuration cannot be determined through text similarity alone, motivating the second phase of our study.

## 6 Expert Evaluation and Criteria-Based Assessment Study

The second phase of our study consists of a human evaluation to determine whether the observed differences in text similarity metrics are reflected in human expert judgments (RQ3) and whether DELTA’s criteria-based evaluation module aligns with those judgments (RQ4). As expert evaluation (which requires team members to participate) was feasible only for the proprietary dataset, the following sections focus exclusively on the proprietary dataset.

### 6.1 Research Goal, Questions, and Metrics

Fig. 5 presents the conceptual model for RQ3 and RQ4. RQ3 examines the validity of the reference-based measurements obtained in RQ1 and RQ2 by assessing the extent to which text similarity metrics (Proxy 1) align with expert judgments of decomposition quality. RQ4 assesses the alignment between the scores produced by DELTA’s evaluation module (Proxy 2) and the expert judgments.

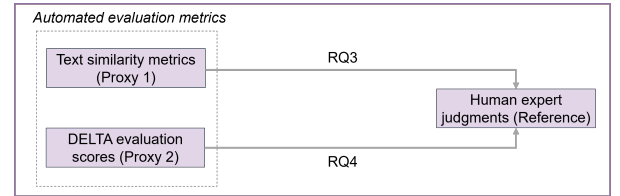


Figure 5: Conceptual model of RQ3 and RQ4.

**RQ3** To what extent do text similarity metrics align with human expert judgments of user story decomposition quality?

**RQ4.** To what extent does the criteria-based automated evaluation module of DELTA align with human expert judgments of decomposition quality?

### 6.2 Expert Evaluation Protocol

Human evaluation serves two complementary purposes: (i) to gather expert preferences on US decompositions to examine whether reference-based text similarity metrics reliably reflect the quality of US decompositions (RQ3), and (ii) to obtain expert quality judgments to assess alignment between human evaluations and the criteria-based automated evaluation module of DELTA (RQ4).

We recruited nine experts responsible for developing or managing the evaluated projects, including seven software engineers (3.5, 3.5, 23, 25, 31, 38, and 38 YoE) and two project managers (14 and 34 YoE). Participants were invited via email with a brief description of the study. Each session began with an informed consent procedure and proceeded only after consent was obtained. During the session, participants were asked to think aloud to articulate their reasoning.

Each participant evaluated six USs using an Azure DevOps sprint board reflecting their familiar working environment. To reduce cognitive load, participants were not shown all candidate decompositions. Instead, each US was presented with four task sets, introducing a selection bias which we discuss in Section 8.2. These consisted of the three highest-ranked LLM-generated decompositions, determined based on their average score across six LLM judges (excluding the generating model to mitigate potential self-preference bias [53]), and the human-created decomposition. In the event of tied scores, the average confidence score across the LLM judges was used as a secondary ranking criterion. The order of USs and task sets was randomized to mitigate ordering effects [22].

To further reduce cognitive load and avoid introducing bias, participants were not asked to rank or rate all US decompositions against quality criteria. Instead, for each US, they selected the best decomposition based on their own judgment and provided a rationale. At the end of the sessions, participants prioritized the most important quality criteria for evaluating US decomposition. They also indicated their willingness to use LLM-generated tasks on a five-point Likert scale: (1) would not use them; (2) for inspiration only; (3) starting point with substantial modifications; (4) starting point with minor modifications; (5) would use them as-is. The sessions lasted between 30 and 60 minutes. Sessions were recorded, transcribed, and analyzed using qualitative content analysis (RQ3 and RQ4) to enrich the quantitative findings (RQ1 and RQ2).

## 7 Expert Evaluation and Criteria-Based Assessment Study Results

We report the results of RQ3 and RQ4 for the proprietary dataset.

### 7.1 Alignment Between Text Similarity Metrics and Human Expert Judgments (RQ3)

As explained in Section 4.2, USs were stratified by field count (partial: 1–2 fields; detailed: 3–5 fields) and task count (low, medium, high based on the 33<sup>rd</sup> and 67<sup>th</sup> percentiles), yielding six strata. One US per stratum per system resulted in 48 USs across eight systems. Across four systems (.NET, Java, and 2 × Power Apps), 21 USs were evaluated by nine experts, with partial overlap for validation.

DELTA ranks LLM-generated decompositions using six LLM judges that assign scores based on the criteria described in Section 3. These scores are averaged (excluding the LLM that generated the decomposition) into an aggregated score, with ties resolved using the average confidence score reported by the LLM judges themselves. Each expert evaluated six USs. For each US, four decompositions were shown to the experts: the top three decompositions as ranked by the LLM judges and the human-created decomposition.

Of the 21 USs evaluated by the nine experts, a majority agreement was reached for 11 USs. The remaining USs are not reported,

as additional expert evaluations are required to draw reliable conclusions. For each US, the experts selected the best out of the four candidate decompositions. These decompositions were also ranked by the text similarity scores (BERTScore, ROUGE-L, and METEOR).

Table 7 presents the rank positions assigned by text similarity metrics to the expert-preferred decompositions across the 11 USs. For example, if the experts select the decomposition ranked second by BERTScore, one count is added to the 2<sup>nd</sup> column for BERTScore.

**Table 7: Rank position of the expert-preferred decomposition across text similarity metrics.**

| Text similarity metric | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> |
|------------------------|-----------------|-----------------|-----------------|-----------------|
| BERTScore              | 0               | 7               | 1               | 3               |
| ROUGE-L                | 0               | 1               | 8               | 2               |
| METEOR                 | 0               | 2               | 8               | 1               |

**Finding 5 ★:** While BERTScore shows a degree of alignment with human expert preferences, none of the three text similarity metrics constitute a reliable standalone evaluation measure.

Because the human-created decomposition serves as the reference for the text similarity metrics, it receives the maximum similarity score and is therefore ranked first by every metric. The 1<sup>st</sup> column represents cases where the experts selected the human-created decomposition as the best decomposition. This did not occur in any of the 11 USs, hence the zero counts in the 1<sup>st</sup> column.

The 2<sup>nd</sup> column represents cases where the decomposition selected by the experts is the highest-ranked LLM-generated decomposition. BERTScore achieves this in 7 of 11 cases (63.6%), compared to 2 cases for METEOR (18.2%) and 1 case for ROUGE-L (9.1%). In contrast, counts in the 3<sup>rd</sup> and 4<sup>th</sup> columns indicate misalignment, where the metric failed to identify the decomposition preferred by the experts. These results indicate that, while BERTScore shows a degree of alignment with expert preferences, none of the three metrics constitute a reliable standalone evaluation measure.

Motivated by this finding, we explore criteria-based evaluation, using LLM judges as a potential proxy for human expert preference. In contrast to text similarity metrics, criteria-based evaluation aims to capture qualitative aspects of decomposition quality.

### 7.2 Alignment Between Evaluation Module and Human Expert Judgments (RQ4)

We follow the same US selection and ranking procedure as in RQ3. We evaluate the alignment between the LLM-generated rankings of US decompositions and the expert judgments in two ways: *i.* for each individual LLM judge, and *ii.* for the ensemble judge.

The analysis is restricted to the 11 USs for which a majority expert agreement was reached. Table 8 presents the rank positions assigned by the LLM judges to the expert-preferred decompositions. For example, if the experts select the decomposition ranked first by GPT-5.2, one count is added to the 1<sup>st</sup> column for GPT-5.2.

**Finding 6 ★:** Mistral Large 3 shows the highest alignment with expert judgments, followed by GPT-5 Pro and GPT-5.2.

All LLM judges consistently rank the human-created US decomposition last, and human experts likewise never prefer it; hence the

**Table 8: Rank position of the expert-preferred decomposition across LLM judges.**

| Judge           | 1 <sup>st</sup> | 2 <sup>nd</sup> | 3 <sup>rd</sup> | 4 <sup>th</sup> |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| GPT-5.2         | 5               | 4               | 2               | 0               |
| GPT-5 Pro       | 7               | 1               | 3               | 0               |
| Llama 3.1 8B    | 4               | 4               | 3               | 0               |
| Llama 3.3 70B   | 3               | 3               | 5               | 0               |
| Mistral Small   | 4               | 4               | 3               | 0               |
| Mistral Large 3 | 8               | 1               | 2               | 0               |
| Ensemble judge  | 3               | 4               | 4               | 0               |

zero counts in the 4<sup>th</sup> column. In contrast, Mistral Large 3 shows the highest alignment with the experts, ranking their preferred US decomposition first in 8 of the 11 cases (72.7%), followed by GPT-5 Pro in 7 cases (63.6%) and GPT-5.2 in 5 cases (45.5%).

The ensemble judge shows the lowest alignment with the experts, ranking their preferred decomposition first in 3 of the 11 cases (27.3%), on par with Llama 3.3 70B, indicating that score aggregation does not improve alignment with human expert preferences in this setting. This finding may reflect the limitations of the smaller models included in the ensemble judge, consistent with the observations of Crupi *et al.*[9] that smaller models do not perform well on judging tasks for code generation and code summarization.

Based on the majority expert vote across the 11 USs, Mistral Large 3 zero-shot prompting was the most frequently selected configuration, followed by GPT-5 Pro zero-shot prompting and GPT-5 Pro few-shot prompting, which were tied for second place.

### 7.3 Additional Insights

We report additional insights from the expert evaluation sessions that are not directly tied to specific research questions.

**Finding 7 ★:** Among software engineers, completeness was most frequently prioritized for evaluating US decomposition quality (5/9); relevance and separation of concerns were less frequent (each 1/9). Estimability was favored by project managers (2/9).

Software engineers emphasized completeness as necessary for implementing USs, noting that *“it is essential to have all the information included, so that you can build and test it”* and that if elements such as interfaces or parameters are missing, *“you cannot build it.”* They also stressed the need for clear and well-specified tasks, emphasizing that *“you know exactly what to do so that you can produce the correct end product,”* and that LLM-generated US decompositions should *“take all aspects of the request into account.”* Separation of concerns was mentioned in relation to avoiding simultaneous work on multiple tasks and reducing the overhead of managing multiple branches and pull requests, while relevance to the US was noted once. In contrast, project managers chose estimability to determine implementation time and communicate effort and cost to clients.

One software engineer identified accuracy as a missing criterion, emphasizing that *“what is written should be correct and applicable to the story,”* distinguishing it from completeness. Another one noted the absence of a definition of done (DoD) [40], i.e., *“when is it finished?”* and reflected on its appropriate level of specification.

While DoD is typically associated with USs and their acceptance criteria, they suggested it may also apply at the task level, depending on whether completing specific elements is considered sufficient.

**Finding 8 :** Most human experts would use the DELTA’s outputs as a starting point with minor adjustments (6/9), while fewer would use them directly (2/9) or with major changes (1/9).

DELTA’s outputs were perceived as a starting point, providing structure and a plan that support progress tracking and estimation. As one software engineer noted, *“as long as it is complete, you can copy and paste it in one go and execute it [with Claude].”* They also highlighted that LLM-generated decompositions are beneficial in larger teams by improving coordination and task distribution.

Experts emphasized that the quality of USs is critical, as they form the basis for decomposition. One expert further noted that using DELTA may improve US quality, as *“you start writing better user stories and then generating tasks.”* Some experts expressed concerns that LLMs may miss nuance or context from USs, particularly when information is incomplete or not explicitly stated, potentially leading to missing or incorrect tasks. One expert also highlighted that manually decomposing tasks contributes to a deeper understanding of the problem, suggesting that fully automating this step may reduce opportunities for reflection. Finally, experts pointed out that in low-code contexts, tasks are defined at a higher level, enabling more flexible implementation than in high-code development.

## 8 Discussion

Text similarity metrics are widely used to evaluate generation tasks [7]; however, our findings suggest that, in the context of this study, these metrics do not always align with human expert judgments for decomposition quality (Finding 5). This discrepancy may stem from the fact that there are multiple valid ways to decompose a US. Moreover, expert evaluations are inherently subjective, as practitioners may apply different assessment criteria depending on their experience and project context. Accordingly, majority agreement among experts was reached for only 11 of the 21 evaluated USs. These findings reinforce the need for evaluation strategies that go beyond similarity to a single human-created reference.

Unlike text similarity metrics, DELTA’s evaluation module, which uses LLM judges, operationalizes decomposition quality through criteria such as completeness and separation of concerns. This enables LLM judges to assess aspects of decomposition quality that go beyond textual similarity. Consistent with this interpretation, the evaluation module produced rankings that align more closely with expert judgment than text similarity metrics (Findings 5 & 6).

Most experts stated they would use DELTA’s decompositions as a starting point with minor adjustments (Finding 8). These findings suggest that LLMs can assist both the generation and evaluation of US decompositions within a human-in-the-loop workflow, where top-ranked LLM-generated decompositions serve as draft inputs for practitioners, who refine them using their contextual knowledge.

### 8.1 Implications

*For practitioners.* We provide DELTA, which includes reusable prompt templates for automated US decomposition and evaluation that can be adapted to different contexts. We recommend that organizations

use LLMs for the generation and evaluation of US decomposition, with human-in-the-loop. The prompts and quality criteria should be tailored to role-specific needs to support effective adoption.

*For researchers.* Our study highlights several directions for future research on automated US decomposition. First, the inconsistent alignment of text similarity metrics with human expert judgments indicates researchers should not rely solely on these metrics for assessing US decomposition quality. Second, future work could investigate whether the generated tasks can serve as intermediate artifacts for automated code generation, and for verifying that the generated code satisfies the US. Third, future work should quantify the costs and human effort required to run DELTA and compare them with manual US decomposition to evaluate savings.

## 8.2 Threats to Validity

We discuss threats to validity and study limitations according to the categories proposed by Wohlin *et al.* [49].

*Construct validity.* We chose text similarity metrics to capture quantitative aspects of US decomposition quality. These metrics may be influenced by the number and granularity of generated tasks. Furthermore, a US can have multiple valid decompositions that differ from the reference set, potentially affecting the scores.

We use five metrics (completeness, relevance to US, estimable, separation of concerns, and uniformity) to assess qualitative aspects of US decomposition quality. These metrics are grounded in the literature [26, 44] and have been adapted to the context of task-level decomposition, but have not been validated. LLM judges applying these metrics may prefer more detailed decompositions, which do not necessarily correspond to higher quality decompositions. They may also fail to fully capture project-specific context, such as team practices or sprint constraints. Future work should validate the criteria and investigate additional quality criteria, such as accuracy.

To mitigate these limitations, we complement LLM-based assessments with human expert evaluations. As expert evaluations are inherently subjective, each US was independently evaluated by multiple experts, and our analysis focused on cases with majority agreement. Additionally, experts evaluated only the top three LLM-generated decompositions ranked by LLM judges, together with the human-created decomposition, introducing a selection bias. Consequently, our findings apply only to the top-ranked LLM-generated decompositions rather than to all generated decompositions.

Team members may rely on implicit knowledge, such as team- and project-specific conventions, that is not captured in the US, which may influence the US decomposition. To partially mitigate this threat, we incorporate domain description and architectural context into the prompts. We exclude augmentation through RAG to isolate the effects of our DELTA approach. In addition, real-world US data may contain inconsistencies (e.g., missing task descriptions). To account for this, we adapt prompts to handle varying input structures and design our sampling strategy to cover different combinations of available fields and contextual information.

*Internal validity.* Observed differences in US decomposition quality attributed to model size may stem from architectural or provider-specific factors. Prompt phrasing may also introduce confounding effects due to model-specific sensitivity to instruction wording. To mitigate these threats, we systematically refined and documented

the prompts and applied consistent prompt templates across models. We further evaluated US decomposition quality across multiple LLMs from different model families to reduce model-specific bias.

*Conclusion validity.* LLMs are non-deterministic, which may affect the reproducibility of experimental outcomes. To reduce output variability, we set the decoding temperature to 0 for locally deployed LLMs [37], logged all prompts, recorded generated US decompositions and LLM-based evaluations, and documented the model versions. Furthermore, the proprietary dataset cannot be publicly released due to confidentiality constraints. Consequently, the study can be replicated but only partially reproduced. To partially mitigate this limitation, we also evaluate DELTA on an open-source dataset, enabling reproduction of the analyses for RQ1 and RQ2.

*External validity.* The study was largely conducted within a single organization, which limits the generalizability of the results to other organizations [5]. Different ITSs and organizational practices may influence how information is documented in USs and tasks, further affecting generalizability. As a partial mitigation, we evaluated DELTA on a sample across eight proprietary projects with varying technologies, domains, and team compositions, and an open-source project. We applied a stratified sampling strategy to ensure that the evaluation covered US decompositions with different numbers of available fields and task counts. We also considered a limited set of LLM families, model sizes, and prompting strategies, which may introduce sampling bias. We mitigate this threat by using LLMs from established providers, including OpenAI, Meta, and Mistral.

## 9 Conclusions

In this paper, we presented DELTA, an LLM-based approach for US decomposition and quality assessment. We evaluated DELTA via an empirical study involving 96 USs from real-world projects. Within our context, persona-based few-shot prompting generally increases text similarity to human-created decompositions (RQ1). The effects of prompting strategy, model family, and size on text similarity vary across the datasets (RQ2), suggesting that no single configuration consistently performed best. We therefore complemented our quantitative analysis with human expert evaluations to determine whether differences in text similarity reflect expert judgments and whether DELTA's criteria-based evaluation module aligns with those judgments. BERTScore has the highest alignment with human expert judgments, followed by METEOR and ROUGE-L, but none of these metrics seem to be a reliable standalone evaluation measure (RQ3). In contrast, DELTA's criteria-based evaluation module achieves higher alignment with expert judgments (RQ4). Consistent with this, experts generally prefer the top three LLM-generated decompositions, as ranked by LLM judges, over human-created decompositions, and they report a willingness to adopt DELTA.

## Acknowledgments

We thank VX Company for providing the data used to construct the proprietary dataset. We are grateful to the colleagues at VX Company for providing the domain descriptions, architectural contexts, and evaluations. Finally, we thank the anonymous reviewers for their feedback, which has contributed to the quality of the paper.

## Data Availability Statement

The replication package includes the open-source dataset, the DELTA implementation, and the prompt templates with their iterations. For the open-source dataset, it also provides the domain description, architectural context, few-shot examples, and generated US decompositions and evaluations. For the proprietary dataset, the package provides the human evaluation materials (informed consent, session protocol, expert characteristics, and evaluation session durations). The proprietary dataset itself cannot be publicly released due to confidentiality constraints. For both datasets, it includes descriptive statistics of US fields and task titles and statistical results [29].

## References

- [1] Negar Alizadeh, Boris Belchev, Nishant Saurabh, Patricia Kelbert, and Fernando Castor. 2025. Language Models in Software Development Tasks: An Experimental Analysis of Energy and Accuracy. In *2025 IEEE/ACM 22nd International Conference on Mining Software Repositories (MSR)*. IEEE Press, Ottawa, Canada, 725–736. doi:10.1109/MSR66628.2025.00109.
- [2] Sebastian Baltes et al. 2026. Guidelines for Empirical Studies in Software Engineering involving Large Language Models. (2026). <https://arxiv.org/abs/2508.15503> [cs. SE].
- [3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Association for Computational Linguistics, Ann Arbor, Michigan, United States, (June 2005), 65–72. <https://aclanthology.org/W05-0909/>.
- [4] Maxim Bragilovski, Ashley T. Van Can, Fabiano Dalpiaz, and Arnon Sturm. 2024. Deriving Domain Models From User Stories: Human vs. Machines. In *2024 IEEE 32nd International Requirements Engineering Conference (RE)*. Reykjavik, Iceland, 31–42. doi:10.1109/RE59067.2024.00014.
- [5] Lionel Briand, Domenico Bianculli, Shiva Nejati, Fabrizio Pastore, and Mehrdad Sabetzadeh. 2017. The Case for Context-Driven Software Engineering Research: Generalizability Is Overrated. *IEEE Software*, 34, 5, (Jan. 2017), 72–75. doi:10.1109/MS.2017.3571562.
- [6] Tom B. Brown et al. 2020. Language Models are Few-Shot Learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS)* Article 159. Vol. 33. Curran Associates Inc., Vancouver, Canada, 1877–1901. [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf).
- [7] Shweta Chauhan and Philemon Daniel. 2023. A Comprehensive Survey on Various Fully Automatic Machine Translation Evaluation Metrics. *Neural Processing Letters*, 55, 9, (May 2023), 12663–12717. doi:10.1007/s11063-022-10835-4.
- [8] Mike Cohn. 2004. *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, (Mar. 2004). ISBN: 978-0-321-20568-1.
- [9] Giuseppe Crupi, Rosalia Tufano, Alejandro Velasco, Antonio Mastropaolo, Denys Poshyvanyk, and Gabriele Bavota. 2025. On the Effectiveness of LLM-as-a-Judge for Code Generation and Summarization. *IEEE Transactions on Software Engineering*, 51, 8, (Aug. 2025), 2329–2345. doi:10.1109/TSE.2025.3586082.
- [10] Emanuel Dellsén, Karl Westgärth, and Jennifer Horkoff. 2022. Invest in Splitting: User Story Splitting Within the Software Industry. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*. Vol. 13216. Springer International Publishing, Birmingham, United Kingdom, 115–130. doi:10.1007/978-3-030-98464-9\_10.
- [11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, United States, (June 2019), 4171–4186. doi:10.18653/v1/N19-1423.
- [12] Carlos Alberto dos Santos, Kevin Bouchard, and Bianca Minetto Napoleão. 2025. Automatic user story generation: A comprehensive systematic literature review. *International Journal of Data Science and Analytics*, 20, 1, 1–24. doi:10.1007/s41060-024-00567-0.
- [13] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M. Zhang. 2023. Large Language Models for Software Engineering: Survey and Open Problems. In *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. Melbourne, Australia, 31–53. doi:10.1109/ICSE-FoSE59343.2023.00008.
- [14] Cuiyun Gao, Xing Hu, Shan Gao, Xin Xia, and Zhi Jin. 2025. The Current Challenges of Software Engineering in the Era of Large Language Models. *ACM Transactions on Software Engineering and Methodology*, 34, 5, Article 127, (May 2025), 1–30. doi:10.1145/3712005.
- [15] Fabian Gilson, Matthias Galster, and François Georis. 2020. Generating Use Case Scenarios from User Stories. In *Proceedings of the International Conference on Software and System Processes (ICSSP)*. Association for Computing Machinery, Seoul, Republic of Korea, 31–40. doi:10.1145/3379177.3388895.
- [16] Tuğçe Güneş and Fatma Başak Aydemir. 2020. Automated Goal Model Extraction from User Stories Using NLP. In *2020 IEEE 28th International Requirements Engineering Conference (RE)*. Zurich, Switzerland, 382–387. doi:10.1109/RE4852.1.2020.00052.
- [17] Xinyi Hou et al. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology*, 33, 8, Article 220, (Dec. 2024), 1–79. doi:10.1145/3695988.
- [18] Ivar Jacobson, Grady Booch, and James Rumbaugh. 1999. *The Unified Software Development Process*. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, United States, (May 1999), 463. ISBN: 978-0-201-57169-1.
- [19] Kai-Kristian Kemell, Matti Saarikallio, Anh Nguyen-Duc, and Pekka Abrahamson. 2025. Still just personal assistants? A multiple case study of generative AI adoption in software organizations. *Information and Software Technology*, 186, (Oct. 2025), 107805. doi:10.1016/j.infsof.2025.107805.
- [20] Alexander Korn, Lea Zaruchas, Chetan Arora, Andreas Metzger, Sven Smolka, Fanyu Wang, and Andreas Vogelsang. 2026. Reporting LLM Prompting in Automated Software Engineering: A Guideline Based on Current Practices and Expectations. (2026). <https://arxiv.org/abs/2601.01954> arXiv: 2601.01954 [cs. SE].
- [21] Richard Lawrence and Peter Green. 2025. The Humanizing Work Guide to Splitting User Stories. <https://www.humanizingwork.com/the-humanizing-work-guide-to-splitting-user-stories/>. Accessed: 2025-11-19. (2025).
- [22] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. 2017. *Research Methods in Human-Computer Interaction*. (2nd ed.). Morgan Kaufmann, Cambridge, Massachusetts, United States. ISBN: 978-0-12-805390-4.
- [23] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, (July 2004), 74–81. <https://aclanthology.org/W04-1013/>.
- [24] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the Middle: How Language Models Use Long Contexts. *Transactions of the Association for Computational Linguistics*, 12, (Feb. 2024), 157–173. doi:10.1162/tacl\_a\_00638.
- [25] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Singapore, (Dec. 2023), 2511–2522. doi:10.18653/v1/2023.emnlp-main.153.
- [26] Garm Lucassen, Fabiano Dalpiaz, Jan Martijn Van Der Werf, and Sjaak Brinkkemper. 2016. Improving agile requirements: the Quality User Story framework and tool. *Requirements Engineering*, 21, 3, (Sept. 2016), 383–403. doi:10.1007/s00766-016-0250-x.
- [27] Garm Lucassen, Marcel Robeer, Fabiano Dalpiaz, Jan Martijn Werf, and Sjaak Brinkkemper. 2017. Extracting conceptual models from user stories with Visual Narrator. *Requirements Engineering*, 22, 3, (Sept. 2017), 339–358. doi:10.1007/s00766-017-0270-1.
- [28] Delina Ly, Fatma Başak Aydemir, and Fabiano Dalpiaz. 2026. From Concept to Action: How Do Agile Software Teams Decompose User Stories into Tasks? In *38th International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer Nature Switzerland, Verona, Italy, 267–283. doi:10.1007/978-3-032-28110-4\_15.
- [29] Delina Ly, Fatma Başak Aydemir, and Fabiano Dalpiaz. Replication Package for the Paper “Can Large Language Models Decompose User Stories into Tasks? Exploring the Role of Prompting Strategies and Models”. Zenodo. doi:10.5281/zenodo.21788048.
- [30] Laurens Müter, Tejaswini Deoskar, Max Mathijssen, Sjaak Brinkkemper, and Fabiano Dalpiaz. 2019. Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer International Publishing, Essen, Germany, 109–116. doi:10.1007/978-3-030-15538-4\_7.
- [31] Samia Nasiri, Amina Adadi, and Mohammed Lahmer. 2023. Automatic generation of business process models from user stories. *International Journal of Electrical and Computer Engineering*, 13, 1, (Feb. 2023), 809–822. doi:10.11591/ijece.v13i1.pp809-822.
- [32] Ipek Ozkaya. 2023. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. *IEEE Software*, 40, 3, (May 2023), 4–8. doi:10.1109/MS.2023.3248401.
- [33] Luka Pavlič, Reinhard Bernsteiner, Stephan Schlögl, and Christian Ploder. 2025. Splitting User Stories Into Tasks with AI - A Foe or an Ally? In *International Conference on Knowledge Management in Organisations (KMO)*. Springer Nature Switzerland, Kota Kinabalu, Malaysia, 150–160. doi:10.1007/978-3-031-95901-1\_11.

- [34] Giovanni Quattrocchi, Liliana Pasquale, Paola Spoletini, and Luciano Baresi. 2026. Can LLMs Generate User Stories and Assess Their Quality? *IEEE Transactions on Software Engineering*, 52, 5, (May 2026), 1773–1790. doi:[10.1109/TSE.2026.3670612](https://doi.org/10.1109/TSE.2026.3670612).
- [35] Indra Kharisma Raharjana, Daniel Siahaan, and Chastine Faticah. 2021. User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access*, 9, 53811–53826. doi:[10.1109/ACCESS.2021.3070606](https://doi.org/10.1109/ACCESS.2021.3070606).
- [36] Tajmilur Rahman, Yuecai Zhu, Lamyeh Maha, Chanchal Roy, Banani Roy, and Kevin Schneider. 2024. Take Loads Off Your Developers: Automated User Story Generation using Large Language Model. In *2024 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Flagstaff, Arizona, United States, 791–801. doi:[10.1109/ICSME58944.2024.00082](https://doi.org/10.1109/ICSME58944.2024.00082).
- [37] Matthew Renze. 2024. The Effect of Sampling Temperature on Problem Solving in Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*. Association for Computational Linguistics, Miami, Florida, United States, (Nov. 2024), 7346–7356. doi:[10.18653/v1/2024.findings-emnlp.432](https://doi.org/10.18653/v1/2024.findings-emnlp.432).
- [38] Devjeet Roy, Sarah Fakhoury, and Venera Arnaoudova. 2021. Reassessing automatic evaluation metrics for code summarization tasks. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*. Association for Computing Machinery, Athens, Greece, 1105–1116. doi:[10.1145/3468264.3468588](https://doi.org/10.1145/3468264.3468588).
- [39] June Sallou, Thomas Durieux, and Annibale Panichella. 2024. Breaking the Silence: the Threats of Using LLMs in Software Engineering. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. Association for Computing Machinery, Lisbon, Portugal, 102–106. doi:[10.1145/3639476.3639764](https://doi.org/10.1145/3639476.3639764).
- [40] Ana Silva, Thalles Araújo, João Nunes, Mirko Perkusich, Ednaldo Dilenzo, Hyggo Almeida, and Angelo Perkusich. 2017. A systematic review on the use of Definition of Done on agile software development projects. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (EASE)*. Association for Computing Machinery, Karlskrona, Sweden, 364–373. doi:[10.1145/3084226.3084262](https://doi.org/10.1145/3084226.3084262).
- [41] Yongjian Tang, Doruk Tuncel, Christian Koerner, and Thomas Runkler. 2025. The Few-shot Dilemma: Over-prompting Large Language Models. In *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*. IEEE, Vienna, Austria, (Nov. 2025), 134–141. doi:[10.1109/fllm67465.2025.11391015](https://doi.org/10.1109/fllm67465.2025.11391015).
- [42] Jonathan Ullrich, Matthias Koch, and Andreas Vogelsang. 2025. From Requirements to Code: Understanding Developer Practices in LLM-Assisted Software Engineering. In *2025 IEEE 33rd International Requirements Engineering Conference (RE)*. Valencia, Spain, 257–266. doi:[10.1109/RE63999.2025.00032](https://doi.org/10.1109/RE63999.2025.00032).
- [43] Ashley T. van Can and Fabiano Dalpiaz. 2024. Requirements Information in Backlog Items: Content Analysis. In *Requirements Engineering: Foundation for Software Quality (REFSQ)*. Springer Nature Switzerland, Winterthur, Switzerland, 305–321. doi:[10.1007/978-3-031-57327-9\\_19](https://doi.org/10.1007/978-3-031-57327-9_19).
- [44] Bill Wake. 2003. INVEST in Good Stories, and SMART Tasks. Accessed: 2026-03-14. (2003). <https://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>.
- [45] Ruiqi Wang, Jiyu Guo, Cuiyun Gao, Guodong Fan, Chun Yong Chong, and Xin Xia. 2025. Can LLMs Replace Human Evaluators? An Empirical Study of LLM-as-a-Judge in Software Engineering. *Proceedings of the ACM on Software Engineering*, 2, ISSTA, (June 2025), 23 pages. doi:[10.1145/3728963](https://doi.org/10.1145/3728963).
- [46] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. In *Proceedings of the 30th Conference on Pattern Languages of Programs (PLoP)* Article 5. The Hillside Group, Monticello, Illinois, United States, 31 pages. doi:[10.64346/PLoP2023p05](https://doi.org/10.64346/PLoP2023p05).
- [47] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1, 6, 80–83. doi:[10.2307/3001968](https://doi.org/10.2307/3001968).
- [48] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the 2011 CHI Conference on Human Factors in Computing Systems (CHI)*. Association for Computing Machinery, Vancouver, Canada, 143–146. doi:[10.1145/1978942.1978963](https://doi.org/10.1145/1978942.1978963).
- [49] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated, 259. ISBN: 978-3-642-29043-5.
- [50] Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond. *ACM Transactions on Knowledge Discovery from Data*, 18, 6, Article 160, (Apr. 2024), 1–32. doi:[10.1145/3649506](https://doi.org/10.1145/3649506).
- [51] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. (2020). <https://arxiv.org/abs/1904.09675> arXiv: 1904.09675 [cs.CL].
- [52] Xuejun Zhang, Xia Hou, Xiuming Qiao, and Wenfeng Song. 2024. A review of automatic source code summarization. *Empirical Software Engineering*, 29, 6, (Oct. 2024), 162. doi:[10.1007/s10664-024-10553-6](https://doi.org/10.1007/s10664-024-10553-6).
- [53] Lianmin Zheng et al. 2023. Judging LLM-as-a-Judge with MT-bench and Chatbot Arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems (NeurIPS)* Article 2020. Vol. 36. Curran Associates Inc., New Orleans, Louisiana, United States, 46595–46623.
- [54] Tingwei Zhu, Zhong Li, Minxue Pan, Chaoxuan Shi, Tian Zhang, Yu Pei, and Xuandong Li. 2024. Deep Is Better? An Empirical Comparison of Information Retrieval and Deep Learning Approaches to Code Summarization. *ACM Transactions on Software Engineering and Methodology*, 33, 3, Article 67, (Mar. 2024), 37 pages. doi:[10.1145/3631975](https://doi.org/10.1145/3631975).

Received 2026-03-26; accepted 2026-06-18