A Comparative Study of LLMs for Gherkin Generation

Hiago Fernandes
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
hiagosousa@copin.ufcg.edu.br

Mirko Perkusich
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
mirko@virtus.ufcg.edu.br

Danyllo Albuquerque
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
danyllo.albuquerque@virtus.ufcg.edu.br

Izabella Silva
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
izabella.silva@virtus.ufcg.edu.br

Danilo F. S. Santos
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
danilo.santos@virtus.ufcg.edu.br

Kyller Gorgônio
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
kyller@virtus.ufcg.edu.br

Angelo Perkusich
ISE Group, VIRTUS/UFCG
Campina Grande, Brazil
perkusic@virtus.ufcg.edu.br

ABSTRACT

[Context] Behavior-Driven Development (BDD) is widely adopted, but the manual creation of Gherkin scenarios remains a significant bottleneck. While Large Language Models (LLMs) show promise for automation, there is a lack of empirical evidence on their accuracy and stability when converting free-form test descriptions into structured Gherkin, creating risks for industrial adoption. Manual scenario authoring is also time-consuming and prone to inconsistencies, leading to miscommunication between technical and non-technical stakeholders and impacting software quality assurance. **[Objective]** This study addresses this gap by investigating the use of LLMs to automate the generation of Gherkin-based BDD scenarios from real-world, free-form test case descriptions. The goal is to assess the robustness of current models when handling informal, ambiguous, and diverse inputs typically found in practice. **[Method]** We conducted a comparative evaluation involving seven LLMs — GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3, Gemini, and DeepSeek R1 — using zero-shot, one-shot, and few-shot prompting strategies. The models generated BDD scenarios from a stratified sample of ten test descriptions selected from a corpus of 1,286, ensuring diversity in structure and domain complexity. We assessed quality and consistency using quantitative metrics (METEOR, variability analysis) and Repeated Measures ANOVA to test statistical significance. **[Results]** The analysis revealed that simple zero-shot prompting was highly effective, achieving results comparable to more complex example-based prompting. For the top-performing model, Gemini, which balanced accuracy and stability, the difference between zero-shot and few-shot was not statistically significant. Performance differences across models were often small, suggesting that practical factors like integration and cost should also guide model choice. Some models showed higher output variability, raising concerns about consistency in test generation workflows. **[Conclusion]** This paper offers practical insights into prompt design and model selection for LLM-based BDD scenario generation. Results show that effective zero-shot prompts can enable scalable, high-quality generation comparable to more complex techniques, simplifying LLM adoption in industrial testing. These findings suggest that LLMs can be leveraged with minimal setup to streamline BDD, reduce costs, and accelerate validation cycles.

KEYWORDS

Behavior-Driven Development; Gherkin; Large Language Models; Test Automation; Prompt Engineering.

1 Introduction

Modern software systems require methods that integrate business requirements, development, and quality assurance. Behavior-Driven Development (BDD) [19, 22] addresses this need by translating behavior specifications into automated test scenarios using a shared, semi-structured language like Gherkin. Unlike the technical focus of Test-Driven Development (TDD), BDD fosters collaboration by involving non-technical stakeholders in scenario creation and validation [22, 24].

Despite these benefits, manually writing BDD scenarios poses challenges in large-scale projects. Scenario quality and consistency are difficult to maintain across distributed teams, especially under evolving requirements [17]. Manual authoring is also labor-intensive and error-prone. Automation can improve BDD's scalability by reducing effort, increasing consistency, and minimizing human error.

Large Language Models (LLMs) offer a promising path to such automation. With strong natural language capabilities, models like GPT-3.5 and GPT-4 can generate syntactically valid Gherkin scenarios, even with limited input [19]. Compared to rule-based methods, LLMs better capture implicit requirements, align outputs with business goals, and adapt to domain-specific language [16, 19].

Prior efforts have explored using LLMs to generate test cases or conceptual models [16, 19]. Still, these studies often focus on code quality, overlook structured BDD constraints, or lack rigorous quantitative evaluation. Moreover, existing work does not benchmark different LLMs on free-form test-case translation into Gherkin, nor does it examine output stability under varied prompting strategies.

This gap is critical: premature adoption of LLMs in CI/CD pipelines risks introducing silent failures and flaky tests. A rigorous benchmark is essential for guiding adoption and informing post-editing

and repair strategies [18, 29], especially given broader concerns around data leakage and reproducibility [32].

Translating informal descriptions into formal Gherkin syntax is a core BDD challenge [20]. For example, the sentence "The user must be able to log in with their credentials and access their account" can be rendered as:

```
Scenario: Successful Login
  Given the user is on the login page
  When the user enters valid credentials
  Then the user should be redirected to the dashboard
```

Figure 1: Gherkin-formatted BDD scenario.

Creating such scenarios manually requires a detailed understanding of syntax and domain context, limiting scalability [4]. LLMs can automate this process while respecting structural constraints and managing linguistic variability [14, 25], accelerating development and improving consistency. Studies show automation can reduce development time by 30% and test error rates by 25% [2].

Still, LLM performance depends on factors such as model architecture, tuning, and task design. While generalist models like GPT-4 offer adaptability, specialized models (e.g., Codex, Flan-PaLM) target code generation [26]. Understanding these trade-offs is vital for effective deployment. Recent work also points to the broader impact of LLMs on tasks like code generation and maintenance [38].

In this study, we evaluate several LLMs—GPT-3.5, GPT-4, Gemini, Llama 3, DeepSeek R1, and Phi-3—using three prompting strategies (zero-, one-, and few-shot) to generate Gherkin scenarios from ten free-form test descriptions drawn from a curated dataset of 1,286 cases. We compare their accuracy and stability, with Gemini showing the best average performance. While differences were not statistically significant, Gemini's results warrant closer analysis. We present both comparative findings and a detailed investigation of Gemini's behavior and prompt design. Two research questions guide this work:

- **RQ1** – How do model choice and prompting strategy (zero-, one-, few-shot) influence the accuracy of Gherkin BDD scenarios, measured by METEOR?
- **RQ2** – How do the same factors affect run-to-run stability, expressed as the coefficient of variation (CV) of METEOR scores?

The remainder of this paper is organized as follows. Section 2 reviews BDD and LLM-driven test generation. Section 3 details our prompt engineering methodology, followed by our experimental setup in Section 4. Sections 5 and 6 present comparative results and a focused analysis of Gemini. Section 7 discusses practical implications, and Section 8 addresses threats to validity. We conclude in Section 9 with recommendations and future directions.

2 Background and Related Work

Behavior-Driven Development (BDD). BDD is an agile methodology that extends TDD by promoting collaboration among developers, testers, and non-technical stakeholders [30, 37]. Its core goal is to align software behavior with business expectations by expressing requirements in a shared, human-readable format [8, 34]. This alignment improves communication and supports test automation.

BDD scenarios use the Gherkin language, which relies on keywords like `Given`, `When`, and `Then` to describe preconditions, actions, and outcomes [1]. These scenarios serve as both documentation and executable acceptance tests, supporting traceability between requirements and implementation. Tools such as Cucumber, JBehave, and SpecFlow automate Gherkin execution and help maintain quality assurance [5, 15].

Gherkin's declarative syntax promotes modular, reusable tests and strengthens communication across roles [15, 25]. BDD has shown particular value in agile settings, where iterative development and tight feedback loops are critical [27]. However, success depends on clear writing and adherence to best practices. Ambiguous steps or redundant phrasing can reduce BDD's effectiveness [7], while domain-specific tuning and tooling support—such as IDE integration and CI/CD compatibility—are often necessary [28, 30].

In summary, BDD improves software quality through executable, structured, and accessible test scenarios. Gherkin's syntax enables teams to align expectations and track system behavior. Recent progress in natural language processing, particularly LLMs, presents new opportunities to automate BDD scenario generation and scale agile practices with reduced manual effort.

LLMs in Automated Test Case Generation. Automated test generation aims to improve testing efficiency, reduce manual effort, and enhance fault detection [12]. Traditional methods like symbolic execution and search-based software testing (SBST) have been effective but often involve high computational cost and limited adaptability to natural language [12].

Transformer-based LLMs [35] have expanded the scope of automation by generating structured test cases from free-form inputs such as user stories and functional requirements [10]. Within BDD, LLMs help translate natural language into executable Gherkin scenarios, reinforcing the idea of "living documentation" that evolves with the software [19, 40].

Empirical studies have begun to benchmark LLMs—including GPT-3.5 Turbo, GPT-4 Turbo, GPT-4o Mini, LLaMA 3, Phi-3, Gemini 1.5 Pro, and DeepSeek R1—for BDD scenario generation [13, 33, 39]. These models vary in architecture, training data, and instruction-following ability. Gemini, in particular, has shown strong performance in structured generation due to its calibrated outputs and multimodal capabilities [33].

Beyond BDD, LLMs have been tested on unit test generation. The TESTEVAL benchmark [36] evaluates models like GPT-4o and Gemini 1.0 Pro on Python test generation tasks, measuring metrics such as branch coverage and path execution. While LLMs perform well in general coverage, they still struggle with complex control flow, where traditional methods like SBST remain more precise [12].

Complementary studies show that LLMs can match or exceed manual efforts in test coverage but remain weaker in fault detection [11]. This supports hybrid approaches where LLMs assist human testers. Reinforcement learning (RL) agents have also been integrated into BDD workflows, especially for UI testing [23]. These agents autonomously explore interfaces and generate readable Gherkin scenarios, enhancing adaptability.

Despite progress, challenges remain. Prompt sensitivity, hallucinations, syntactic errors, and domain misalignment persist across studies. Long-term maintainability and real-world adoption also

require further investigation. Nonetheless, LLMs offer flexible, scalable solutions that align well with agile and user-centered development. Ongoing work in prompt engineering, hybrid systems, and evaluation methods will be critical to realizing their full potential.

3 Prompt Engineering for BDD Scenario Generation

Prompt engineering is a critical component in harnessing the full potential of LLMs, especially in structured tasks such as generating BDD scenarios. The clarity, specificity, and structure of the prompt provided heavily influence the effectiveness of the output produced by an LLM. In this study, prompt design played a fundamental role in ensuring that the generated BDD scenarios adhered strictly to the Gherkin syntax and incorporated best practices recognized in the software engineering (SE) community.

The prompts were carefully constructed by integrating BDD quality guidelines identified by Oliveira et al. [25]. These guidelines were selected because they originate directly from interviews with experienced BDD practitioners, offering a valuable, field-tested perspective. Furthermore, given that BDD best practices often lack rigid standardization and can be subjective, the checklist derived from these interviews provided concrete, actionable criteria aligned with a professional viewpoint, making it particularly suitable for structuring the prompts. Key elements drawn from this checklist include articulating the business value clearly, focusing on a single action and its outcome, using declarative language, maintaining terminological consistency aligned with business language, and adopting a third-person point of view. These principles were embedded in all prompts to improve clarity, traceability, and alignment with stakeholder expectations.

By combining established BDD practices with systematic prompt engineering techniques, the study ensured that the input to the LLMs provided structure and context, enabling better generalization and reducing ambiguity in scenario generation. The prompts were adapted to three distinct prompting strategies—*zero-shot*, *one-shot*, and *few-shot*—each varying in the amount of contextual information and number of examples provided.

In the *zero-shot* setting, the prompt provided clear instructions without examples, depending entirely on the model's internal understanding of task structure and Gherkin syntax conventions. The *one-shot* approach extended this by incorporating a single illustrative example to explicitly demonstrate expected syntax and semantics, reinforcing structural alignment. Conversely, the *few-shot* strategy employed multiple diverse examples to enhance the model's generalization capability across various functional domains, though this increased computational overhead due to the longer prompts required. An illustrative example of the prompt used for the *zero-shot* configuration in this study is shown in Figure 2.

The prompts for the *one-shot* and *few-shot* settings followed similar guidelines, differing primarily by incorporating one or more illustrative examples preceding the task instructions. The final prompt designs presented here resulted from an iterative refinement process, where initial versions were tested and adjusted based on analyses of the generated outputs concerning Gherkin syntax and quality criteria. The complete set of prompts used in this study, including examples from all configurations, is available online[1].

```
Convert the following test case description into a single BDD scenario
using strict Gherkin syntax. Ensure the output contains only the Gherkin
syntax for the scenario, without comments, explanations, or the word "Feature."
Use Portuguese for the scenario details, but keep Gherkin keywords in English.

Now convert the following test case description into exactly one BDD
scenario using strict Gherkin syntax. The output must follow the format
of the provided example and include only the BDD scenario. Only Gherkin
keywords should appear in English; all other text must be in Portuguese.

Test Case Description:
test_case_description

Make sure to clearly declare the business value or expected outcome
and focus on a single action and result. Use only the essential steps
(Given, When, Then, And) in a clear and declarative way, avoiding
implementation details and unnecessary repetition. Scenarios should be
independent, use consistent business terminology without technical jargon,
and be written in the third person to avoid ambiguity.

Scenarios must be indented with two spaces under 'Scenario', without
blank lines between steps, and a blank line should separate different scenarios.

Follow this structure for the output:

Scenario: [Brief Scenario Description]
  Given [initial context]
  And [additional context, if any]
  When [an action is performed]
  Then [a specific outcome should occur]
  And [another outcome, if any]

Only the BDD scenario should be returned, without formatting or
additional text (e.g., no 'gherkin```'), and it must be a single valid
Gherkin scenario.
```

Figure 2: Prompt used for zero-shot BDD scenario generation.

Furthermore, the study acknowledges the limitations inherent in each prompting strategy. The *zero-shot* approach, while efficient, may underperform in tasks requiring domain-specific reasoning [31]. The *one-shot* strategy is sensitive to the quality of the single example, which can lead to overfitting or misgeneralization [**?**]. Although more robust, the *few-shot* technique faces scalability challenges and depends heavily on the representativeness of selected examples [6].

In summary, this study's prompt engineering process was guided by theoretical and empirical considerations. It leveraged best practices from BDD to shape instructions and examples, while applying principles of prompt design to tailor input for LLMs. This alignment between domain knowledge and prompt structure was crucial to achieving consistent, high-quality outputs in the automated generation of BDD scenarios.

4 Study Design

This study adopts an experimental and exploratory methodology to evaluate the ability of state-of-the-art LLMs to generate BDD test scenarios in Gherkin syntax based on free-form natural language descriptions. The methodological design includes dataset preparation, prompt strategies, controlled multi-model evaluations, and qualitative and quantitative analyses to assess model effectiveness and consistency. Each component of this methodology is elaborated upon in the subsequent subsections.

[1]Available at: https://github.com/hiagonfs/bdd-scenario-evaluation

The overall workflow of our study is depicted in Figure 3. It outlines the five main stages of our research, from dataset preparation to statistical analysis, which will be detailed in the following sections.

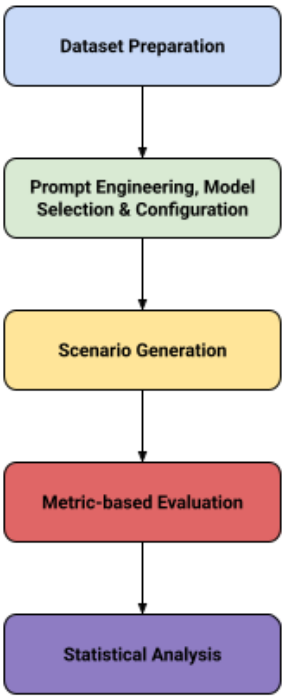

Figure 3: Overview of the experimental methodology pipeline.

Dataset Construction. The dataset comprises 1,286 real-world test cases written in natural language, extracted from a production software system [9]. After preprocessing, which involved removing very short cases, filtering out redundant or vague descriptions, and selecting functionally relevant scenarios, a sample of 10 cases was randomly drawn. These cases were manually converted into reference BDD scenarios using Gherkin syntax by a domain specialist, following best practices identified by Oliveira et al. [25]. This curated reference set served as the ground truth for all evaluations, providing a reliable benchmark to assess the accuracy and consistency of the generated BDD scenarios. Establishing such a ground truth is crucial in evaluating generative AI applications, as it ensures that model outputs can be objectively compared against predefined standards [3].

Evaluated Models and Prompting Strategies. Seven LLMs were selected to cover a wide range of architectures, sizes, and capabilities, including both commercial and open-source models. Table 1 lists the models and their respective versions used in this study.

Each model was evaluated using three prompting strategies: zero-shot, one-shot, and few-shot. The full description of the prompt engineering process, including the rationale, design, and examples for each strategy, is detailed in Section 3.

Table 1: Evaluated models and their respective versions.

Model	Version
GPT-3.5 Turbo	gpt-3.5-turbo-0125
GPT-4o Mini	gpt-4o-mini
GPT-4 Turbo	gpt-4-turbo
Gemini	gemini-1.5-pro
LLaMA 3	meta-llama-3-70b-instruct
Phi-3 Mini	phi-3-mini-128k-instruct
DeepSeek	deepseek-r1:free

Experimental Design and Evaluation Metrics. Each LLM was prompted with the 10 selected test cases using each of the three prompting techniques, producing 30 unique generations per model. To assess intra-model variability, each combination was repeated five times, totaling 150 generations per LLM. All outputs were stored with metadata (i.e., model name, technique, timestamp, and response text) for reproducibility.

For evaluating the similarity between the generated and reference scenarios, we initially considered three complementary evaluation metrics: Manhattan Distance, BERTScore, and METEOR. These metrics were selected based on their ability to capture distinct aspects of quality relevant to BDD scenarios: structural differences (Manhattan Distance), semantic similarity via contextual embeddings (BERTScore), and semantic and syntactic similarity incorporating synonym matching and word alignment penalties (METEOR). Each metric was applied to compare the generated BDD scenarios against the ground truth scenarios manually crafted by a domain expert. To select the most suitable metric for the main analysis, we conducted a correlation study between the quantitative scores provided by each metric and the qualitative evaluations performed by the expert, who assessed scenario adequacy and adherence to BDD best practices.

The results indicated that METEOR exhibited the strongest correlation with expert judgments, outperforming both Manhattan Distance and BERTScore in capturing relevant semantic and structural nuances inherent to BDD scenarios. Unlike purely token-based or embedding-based approaches, METEOR accounts for synonymy, stemming, and word alignment, which proved particularly beneficial for evaluating structured natural language tasks such as Gherkin scenarios [21, 41]. Consequently, METEOR was adopted as the primary evaluation metric for this study. Further details on metric selection, implementation specifics, correlation analyses, and complete measurement tables are provided in the supplementary material available online[2].

Analysis Procedure. The evaluation methodology included two key stages. First, the variability of each model under repeated executions was analyzed using descriptive statistics: mean, standard deviation (SD), and coefficient of variation (CV). This allowed the identification of models that are more deterministic or prone to inconsistency across executions. Second, the comparative performance of the models was assessed using METEOR scores for each generated scenario. The normality of distributions was tested using the Shapiro-Wilk test, and variance homogeneity was verified

[2]Available at: https://github.com/hiagonfs/bdd-scenario-evaluation

with Bartlett's test. Depending on the results, Repeated Measures ANOVA, followed by paired t-tests with Bonferroni correction, or Kruskal-Wallis with Dunn's test (with Bonferroni correction) was applied to evaluate significant differences among models and prompting strategies.

Although this paper primarily focuses on the Gemini model, the comparative analysis involving all models is presented and discussed in Section 5. Gemini was chosen for deeper analysis due to its superior average performance and consistency, as will be further detailed.

5 Comparative Evaluation of LLMs in BDD Scenario Generation

This section presents a comparative evaluation of the seven LLMs considered in this study regarding their ability to generate BDD scenarios from free-form test case descriptions. A detailed set of statistical results, including distribution plots and significance tests, is available as supplementary material.

Accuracy Comparison Across Models. Table 2 summarizes the mean METEOR scores achieved by each evaluated LLM across zero-shot, one-shot, and few-shot prompting strategies for BDD scenario generation. Analysis reveals that the zero-shot approach frequently yielded the highest performance. Notably, Gemini (0.84) and Phi-3 Mini (0.81) achieved the top METEOR scores using zero-shot prompting. Furthermore, GPT-3.5 Turbo (0.78) and GPT-4o Mini (0.78) demonstrated strong performance, while LLaMA 3 (0.75) and GPT-4 Turbo (0.74) exhibited competitive results, all also peaking under the zero-shot condition.

Table 2: Mean METEOR scores across LLMs and prompting strategies.

Model	Zero-shot	One-shot	Few-shot
GPT-3.5 Turbo	0.78	0.74	0.72
GPT-4o Mini	0.78	0.71	0.72
GPT-4 Turbo	0.74	0.71	0.73
Gemini	0.84	0.78	0.74
LLaMA 3	0.75	0.71	0.72
Phi-3 Mini	0.81	0.74	0.70
DeepSeek	0.69	0.67	0.71

Conversely, DeepSeek generally recorded lower scores and represented the sole exception, achieving its maximum performance (0.71) with the few-shot strategy. For most models, both one-shot and few-shot prompting typically resulted in lower METEOR scores than zero-shot, with the one-shot strategy often yielding the least favorable outcomes. Based on these findings, the zero-shot technique proved most effective overall for this task, with the Gemini model employing zero-shot attaining the highest METEOR score (0.84), distinguishing it as the best-performing model and strategy combination among those evaluated.

Variability Across Models and Techniques. In addition to accuracy, variability plays a crucial role when considering using LLMs for test scenario generation, as stability directly impacts the reliability of automated pipelines. Variability was assessed by analyzing the mean, standard deviation (SD), and coefficient of variation (CV), calculated from five independent executions for each model and prompting strategy. Table 3 presents the results for the *zero-shot* strategy.

Table 3: Mean, standard deviation, and coefficient of variation (CV) of METEOR scores for Zero-shot prompting.

Model	Mean	SD	CV (%)
GPT-4 Turbo	0.77	0.04	5.19
GPT-3.5 Turbo	0.76	0.05	6.58
GPT-4o Mini	0.78	0.03	3.85
Gemini	0.84	0.03	3.57
Llama 3	0.76	0.05	6.58
Phi-3 Mini	0.81	0.04	4.94
DeepSeek	0.69	0.03	4.35

In the **zero-shot** configuration, variability was more pronounced in LLaMA 3 and GPT-3.5 Turbo (CV = 6.58%), whereas Gemini stood out as the most consistent model, with the lowest CV (3.57%) and SD (0.03). Phi-3 Mini also presented relatively low variability (CV = 4.94%). GPT-4o Mini exhibited strong stability (CV = 3.85%), slightly above Gemini. DeepSeek exhibited moderate variability (CV = 4.35%), higher than Gemini but lower than GPT-3.5 Turbo and LLaMA 3. Table 4 shows the same analysis under the one-shot prompting configuration.

Table 4: Mean, standard deviation, and coefficient of variation (CV) of METEOR scores for One-shot prompting.

Model	Mean	SD	CV (%)
GPT-4 Turbo	0.73	0.04	5.48
GPT-3.5 Turbo	0.72	0.05	6.94
GPT-4o Mini	0.74	0.04	5.41
Gemini	0.78	0.05	6.41
Llama 3	0.70	0.04	5.71
Phi-3 Mini	0.74	0.05	6.76
DeepSeek	0.69	0.04	5.80

With the **one-shot** strategy, which introduces a single example to guide the model, variability tended to decrease for most models. GPT-4o Mini showed the best stability (CV = 5.41%, SD = 0.04), followed closely by GPT-4 Turbo (CV = 5.48%) and LLaMA 3 (CV = 5.71%). Gemini (CV = 6.41%) and Phi-3 Mini (CV = 6.76%) maintained acceptable consistency. DeepSeek showed comparable variability (CV = 5.80%), close to LLaMA 3 and GPT-4 Turbo.

Table 5 presents the results for the *few-shot* strategy.

Table 5: Mean, standard deviation, and coefficient of variation (CV) of METEOR scores for Few-shot prompting.

Model	Mean	SD	CV (%)
GPT-4 Turbo	0.75	0.06	8.00
GPT-3.5 Turbo	0.70	0.04	5.71
GPT-4o Mini	0.73	0.03	4.11
Gemini	0.74	0.06	8.11
Llama 3	0.72	0.03	4.17
Phi-3 Mini	0.70	0.06	8.57
DeepSeek	0.70	0.04	5.71

In the **few-shot** configuration, where the model receives multiple examples, GPT-4o Mini and LLaMA 3 again demonstrated excellent stability, with CVs of 4.11% and 4.17%, respectively. Phi-3 Mini (CV = 8.57%), Gemini (CV = 8.11%), and GPT-4 Turbo (CV = 8.00%) exhibited higher variability. DeepSeek showed moderate variability (CV = 5.71%), on par with GPT-3.5 Turbo.

The comparative evaluation of LLMs in generating BDD scenarios underscores two pivotal observations. Firstly, zero-shot prompting consistently yielded superior accuracy across most models, with Gemini achieving the highest METEOR score of 0.84. This suggests that, for structured tasks like BDD scenario generation, providing clear instructions without examples can effectively leverage the models' pre-trained knowledge. Secondly, the analysis revealed that output stability varies among models and prompting strategies. While Gemini demonstrated high accuracy and low variability under zero-shot prompting, other models exhibited differing degrees of consistency, indicating that model selection and prompting approach should be tailored to the specific requirements of the task.

6 Analysis of Gemini's Performance

Building on the comparative analysis from Section 5, we now delve deeper into the performance of the Gemini model. This section examines its output variability, accuracy, and qualitative behavior under zero-shot, one-shot, and few-shot prompting. Gemini was chosen for this focused analysis due to its strong average performance and promising stability observed earlier.

Variability Analysis: Consistency Across Executions. A key aspect of a reliable generative model is the consistency of its outputs. We assessed Gemini's variability by running each test case five times for every prompting strategy (zero-shot, one-shot, few-shot) and measuring the similarity between the generated BDD scenarios using the METEOR metric. Lower variability indicates more predictable and reproducible results.

To understand variability in practice, consider the test case "Registering a profile with an existing identifier". The goal is to generate a scenario like the ground truth shown in Figure 4.

```
Scenario: Registration with an existing identifier
  Given a user attempts to register with an identifier
    that already exists in the system
  When the user submits the registration form
  Then the system must display an error message indicating
    that the identifier is already in use
  And the system must not create a new profile
```

Figure 4: Ground truth scenario for "Registering a profile with an existing identifier" (translated).

When using the few-shot strategy, Gemini produced different, though semantically valid, scenarios across executions. For instance, Figure 5 shows the output from one run, while Figure 6 shows the output from another run for the same test case and prompt configuration.

Notice the differences in the scenario title, the specific identifier used "joaosilva" vs. "usuarioteste"), the phrasing of steps, and the exact wording of the error message. This lexical and structural variation, despite conveying the same core logic, exemplifies the output variability measured by the METEOR scores (0.52 and 0.82, respectively, compared to the ground truth) and contributes to a high coefficient of variation (CV) for this specific case under few-shot prompting (20.88%).

```
Scenario: Attempt to register profile with already used identifier
  Given the system has a profile already registered with the
    identifier "joaosilva"
  When the user enters the new profile's data
  And the user enters the identifier "joaosilva"
  And the user clicks on "Register"
  Then the system displays the error message
    "This identifier is already in use"
  And the system does not register the new profile
```

Figure 5: Scenario by Gemini (Few-shot) – Run 3 (translated, METEOR 0.52 vs GT).

```
Scenario: Attempt to register profile with duplicate identifier
  Given the system already has a profile with the
    identifier "usuarioteste"
  When the user tries to register a new profile with the
    identifier "usuarioteste"
  Then the system displays an error message
    "Identifier is already in use"
  And the profile is not registered
```

Figure 6: Scenario by Gemini (Few-shot) – Run 4 (translated, METEOR 0.82 vs GT).

Our quantitative analysis revealed distinct variability patterns across the prompting strategies:

Zero-shot: This strategy demonstrated exceptional stability. The average CV across all test cases was remarkably low at 0.23% (SD = 0.39%). In fact, for eight out of ten cases, the outputs were identical across all five runs (CV = 0.00%). Even the cases with minimal variation maintained high accuracy (METEOR > 0.83). This suggests that clear instructions allow Gemini to generate highly consistent and accurate BDD scenarios.

One-shot: Introducing a single example increased inconsistency. The average CV rose significantly to 7.24% (SD = 2.26%). While some cases remained relatively stable (CV $\approx$ 5%), others showed substantial variation (CV > 10%). This indicates that a single example might unduly influence the model, perhaps causing it to focus too much on the example's specific style or content (anchoring bias) rather than generalizing effectively.

Few-shot: Providing three examples led to even less predictability. Although the average CV was slightly lower than one-shot (6.34%), the spread was much wider (SD = 5.75%). Some cases were stable (CV < 4%), but others, like the example shown in Figures 5 and 6, exhibited very high variability (CV up to 20.88%). This suggests multiple examples might introduce conflicting patterns or overwhelm the model, reducing output consistency.

These trends are visually summarized in the violin plot in Figure 7. The plot clearly shows the distribution of METEOR scores for each strategy.

Observe the tall, narrow distribution for zero-shot, concentrated near the top with minimal spread, confirming its high stability and accuracy. The one-shot distribution is slightly wider, indicating more variability. The few-shot distribution is the broadest and most irregular, visually representing its lower consistency and a wider range of accuracy scores across runs.

In conclusion, the zero-shot approach provided the most stable and reproducible results for generating BDD scenarios with Gemini. It achieved high accuracy with minimal variation between runs,

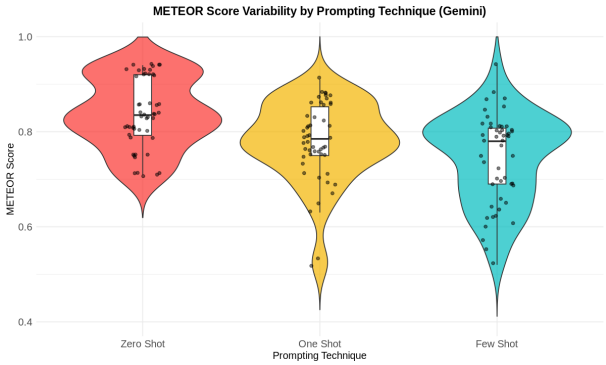

Figure 7: Violin plot comparing METEOR score variability for Gemini across prompting techniques. It illustrates score distributions, medians (white dots), interquartile ranges (black boxes), and individual data points.

suggesting that well-defined instructions are more effective than example-based demonstrations for ensuring consistent outputs. This makes zero-shot prompting a strong candidate for automated quality assurance tasks where predictable and reliable scenario generation is crucial.

Table 6: Takeaways from the Variability Analysis: Consistency Across Executions

No.	Finding description
1	The main objective of this section was to evaluate the consistency (or variability) of Gemini's outputs when generating BDD scenarios using different prompting techniques (zero-shot, one-shot, and few-shot) across multiple executions for the same test case. Lower variability indicates more predictable and reliable results.
2	Zero-shot was exceptionally consistent: The zero-shot technique demonstrated the highest stability and consistency. In many cases (8 out of 10), it produced identical results across all executions, resulting in an extremely low average variability (mean CV of 0.23%).
3	One-shot increases inconsistency: Introducing a single example (one-shot) significantly increased the variability of results (mean CV of 7.24%) compared to zero-shot. This suggests that a single example may overly influence the model (anchoring bias).
4	Few-shot presents high and wide variability: Using multiple examples (few-shot) resulted in lower overall predictability. Although the mean CV (6.34%) was slightly lower than one-shot, the spread of variability was much higher (SD of 5.75%), with some cases showing extremely high variation (CV up to 20.88%). This indicates that multiple examples may introduce conflicting patterns or overwhelm the model.
5	Visual confirmation: The violin plot (Figure 7) visually illustrates these differences, showing a narrow and high distribution for zero-shot (high consistency and accuracy), and progressively wider and more irregular distributions for one-shot and, especially, few-shot.
6	For generating reliable and reproducible BDD scenarios with Gemini, the zero-shot approach is the most recommended. Clear instructions (zero-shot) proved more effective than example-based demonstrations (one-shot, few-shot) in ensuring result consistency for this task.

Accuracy of Prompting Techniques. This section presents a comparative analysis of the accuracy of BDD scenario generation using the Gemini model across three prompting strategies: zero-shot, one-shot, and few-shot. Accuracy was measured using the METEOR metric, which evaluates semantic and lexical similarity between generated and reference scenarios. The analysis considers descriptive statistics (mean, median, SD, and CV) and is complemented by inferential statistical tests to assess significant differences between techniques.

Table 7 displays the measures of central tendency and dispersion for each prompting technique. The zero-shot configuration achieved the highest mean METEOR score (0.84), closely followed by one-shot (0.78), and then few-shot (0.74). The median scores further support this ordering, with zero-shot again leading at 0.83. These values indicate that zero-shot not only provided the most accurate outputs on average but also maintained a strong central tendency, reinforcing its consistency.

Table 7: Measures of central tendency and dispersion of METEOR scores by prompting technique (Gemini).

Technique	Mean	Median	SD	CV (%)
Zero-Shot	0.84	0.83	0.08	9.52%
One-Shot	0.78	0.79	0.07	8.97%
Few-Shot	0.74	0.79	0.10	13.51%

Table 8 complements this analysis by showing the minimum and maximum METEOR scores observed in each configuration. The zero-shot technique produced scores ranging from 0.71 to 0.94, indicating consistently high performance. By contrast, the few-shot approach exhibited the broadest range (0.57 to 0.87), revealing greater inconsistency in the quality of its generated scenarios. The one-shot configuration showed a moderate range, suggesting slightly more stable behavior than the few-shot but still less reliable than the zero-shot.

Table 8: Minimum and maximum METEOR scores by prompting technique (Gemini).

Technique	Min	Max
Zero Shot	0.71	0.94
One Shot	0.67	0.88
Few Shot	0.57	0.87

Figures 8 to 11 present representative examples comparing the outputs of Gemini using the zero-shot prompting strategy against the ground truth. These examples illustrate typical patterns observed in the quantitative analysis, where zero-shot tends to produce semantically faithful outputs with lexical and structural variation.

```
Scenario: Cancel common area registration successfully
  Given a user with permission to manage common areas
  And a common area is already registered in the system
  When the user cancels the common area registration
  Then the system must remove the registration
  And the system must display a success message
```

Figure 8: Ground truth scenario (translated) – Minimum METEOR case.

In Figures 8 and 9, the scenario retains the expected outcome but introduces structural elements not present in the original, such as authentication steps and interface navigation. These additions lead to a lower METEOR score (0.71) due to increased lexical and syntactic divergence.

```
Scenario: Cancel common area registration
  Given the user is authenticated in the system
  And is on the common area management page
  And there is a registered common area
  When the user selects the common area
  And confirms the cancellation
  Then the area is removed from the list
  And the system displays a success message
```

Figure 9: Scenario by Gemini (Zero-shot) – METEOR 0.71 vs Ground Truth.

Conversely, Figures 10 and 11 illustrate a case with high lexical and structural alignment, resulting in the highest observed similarity score under zero-shot prompting.

```
Scenario: Register sprint with start date after end date
  Given a user is registering a new sprint
  When the user enters a start date later than the end date
  Then the system must prevent the sprint from being registered
  And the system must display an error message indicating
    that the start date cannot be later than the end date
```

Figure 10: Ground truth scenario (translated) – Maximum METEOR case.

```
Scenario: Prevent sprint registration with an invalid start date
  Given the user is registering a new sprint
  When the user enters a start date later than the end date
  And attempts to save the registration
  Then the registration is not completed
  And the system displays an error message stating
    that the start date cannot be later than the end date
```

Figure 11: Scenario by Gemini (Zero-shot) – METEOR 0.94 vs Ground Truth.

The scenario in Figure 11 demonstrates a near-identical structure and vocabulary to the reference in Figure 10, differing only in minor details like "the registration is not completed" instead of "prevent the sprint from being registered." These slight variations minimally impact the METEOR score, which remains high (0.94).

Together, these examples highlight how differences in phrasing, added context, or reordering of steps influence similarity metrics, even when the generated scenarios are functionally equivalent.

Statistical testing further validated these findings. The Shapiro-Wilk test confirmed the normal distribution of METEOR scores for all techniques ($p > 0.05$), and Bartlett's test confirmed the homogeneity of variances. A Repeated Measures ANOVA was applied and revealed a statistically significant difference in the mean METEOR scores across the prompting techniques ($F(2, 18) = 5.49$, $p = 0.014$). However, post-hoc comparisons using paired t-tests with Bonferroni correction did not find statistically significant differences between any pair of techniques after the correction (Zero-Shot vs. One-Shot: $p_{adj} = 0.065$; Zero-Shot vs. Few-Shot: $p_{adj} = 0.063$; One-Shot vs. Few-Shot: $p_{adj} = 0.664$).

Figure 12 provides a visual summary of the METEOR score distributions. The boxplot reveals a more concentrated and elevated interquartile range for the zero-shot technique, with a higher median and fewer deviations than the other strategies. Few-shot, in contrast, displayed the widest spread and greater score volatility. No significant outliers were observed, supporting the reliability of the statistical observations.

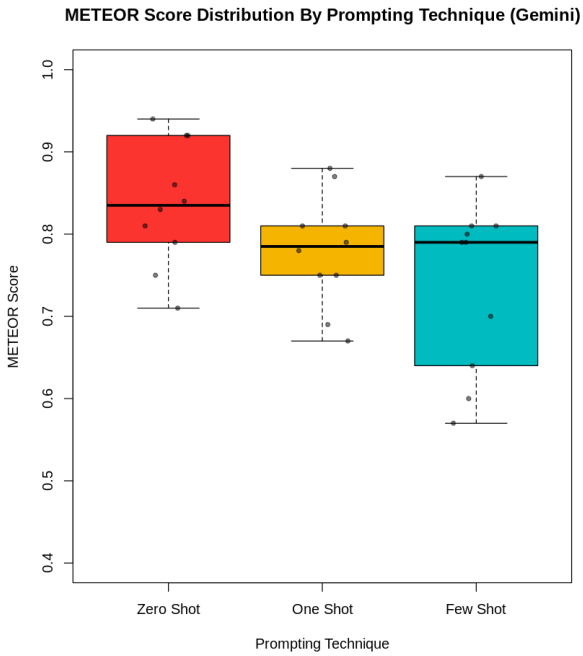

Figure 12: Boxplot of METEOR scores using Gemini for zero-shot, one-shot, and few-shot techniques.

In summary, using Gemini, the zero-shot technique emerged as the most promising and consistent prompting strategy for BDD scenario generation. Although the pairwise differences in accuracy were not statistically significant after post-hoc correction, its high mean and median scores, combined with a relatively low SD and narrow score range, demonstrate strong overall performance and reliability. The one-shot approach offers competitive results, while the few-shot strategy, despite its theoretical benefits in other contexts, resulted in lower accuracy and higher variability, likely due to the increased prompt complexity and potential introduction of conflicting patterns. These findings reinforce the suitability of zero-shot prompting for structured and formal text generation tasks, especially in scenarios requiring precision and reproducibility.

Qualitative Analysis. Beyond the quantitative evaluation of accuracy and stability, a qualitative analysis was conducted to understand the nature of errors present in the BDD scenarios generated by the LLMs. Although other issues were observed, the analysis below focuses on the most frequent categories of errors found. The goal is to demonstrate that the automated generation was not perfect and produced critical errors that could compromise the final execution of the test scenarios. This investigation focused on the outputs of the Gemini model, and the most common mistakes were categorized as follows:

- **Irrelevant Action Insertion:** This category encompasses errors where the model introduces steps or preconditions not implied by the original test case description. A frequent example is the addition of a step like "Given the user is authenticated" when the source material does not mention

any authentication requirement. This suggests the model may over-generalize from common software patterns, adding steps that are not pertinent to the specific behavior under test.

- **Business-Logic Flaws:** This is a more critical type of error involving semantic mistakes that misrepresent the core outcome of the scenario, either by reversing or omitting a key result. For instance, the model might generate a scenario that permits a duplicate user registration when the intended requirement was to prevent it. Such errors pose a significant risk, as they could lead to the creation of automated tests that validate incorrect system behavior.
- **Lexical Deviation:** This category refers to the use of phrasing that, while syntactically correct, subtly but significantly alters the intended meaning of a step. An example is the model generating a step such as "Then the system should display success" when the correct behavior was to "prevent submission and display an error". These deviations can be particularly deceptive, as they may seem plausible at first glance but actually describe the wrong outcome, potentially resulting in flawed test implementations.

This initial overview of the errors found during the generation of the cases highlights the challenges present in the automated generation of BDD scenarios. More importantly, it underscores that the generated outputs require careful human attention and validation before being used. Future work could catalog all error types and their frequencies more exhaustively.

Table 9: Takeaway points – Accuracy Analysis: Similarity to Reference Scenarios.

No.	Key Finding
1	The main objective of this analysis was to compare the accuracy of three prompting strategies (zero-shot, one-shot, and few-shot) when generating BDD scenarios using the Gemini model. Accuracy here refers to the semantic and lexical similarity between the generated scenario and the ground truth, as measured by the METEOR metric.
2	Zero-shot prompting achieved the highest average (0.84) and median (0.83) METEOR scores, indicating that it generally produced outputs most similar to the ideal reference scenarios.
3	One-shot prompting showed intermediate performance, with a mean score of 0.78. While its variability was lower than that of few-shot, it still performed worse than zero-shot in accuracy and consistency (e.g., higher minimum score in zero-shot).
4	Few-shot prompting resulted in the lowest average accuracy (0.74) and the highest variability in scores, with the largest standard deviation, coefficient of variation (CV), and score range (from 0.57 to 0.87). This suggests it was the least precise and most inconsistent strategy overall.
5	Statistical tests revealed a significant overall difference among the techniques (Repeated Measures ANOVA, $p < 0.05$). However, post-hoc tests with Bonferroni correction showed that the pairwise differences between zero-shot, one-shot, and few-shot were not statistically significant after correction.
6	The examples provided in Figures 8 to 11 illustrate how METEOR scores can vary under the same prompting strategy. Some generated scenarios were functionally correct but added steps or used different phrasing (yielding lower scores, e.g., 0.71), while others were nearly identical to the ground truth (yielding higher scores, e.g., 0.94).
7	Based on descriptive metrics, zero-shot prompting emerges as the strongest strategy for this task using Gemini, due to its superior average and median scores, as well as a consistently better score distribution, even if pairwise statistical significance was not achieved.

7 Research Implications

Our findings offer valuable insights for SE research and industry practice regarding using LLMs for BDD scenario generation.

Implications for Research. Prompt effectiveness varies significantly by model and task. Contrary to common assumptions, zero-shot prompting—using clear instructions without examples—often outperforms few-shot prompting for structured outputs like BDD scenarios, particularly with advanced models like Gemini. This result suggests that examples may introduce unnecessary complexity and ambiguity in tasks requiring strict syntax, such as Gherkin.

Researchers should evaluate both accuracy and variability when assessing LLMs and carefully control prompt features, such as the number of examples. Future work should explore whether this preference for zero-shot prompting extends to other structured SE artifacts, including user stories and API documentation.

Our methodological approach—manual ground truth construction, expert-guided prompt design, and mixed-method evaluation—offers a reproducible framework for studying LLMs in SE tasks. This framework can support future investigations into generating structured artifacts such as acceptance criteria, architectural descriptions, and regulatory documents.

Implications for Practitioners. Zero-shot prompting with capable models like Gemini offers high accuracy and low variability, simplifying LLM integration into development workflows and reducing the need for complex prompt engineering. However, model choice should also consider cost, API compatibility, and infrastructure constraints, which may outweigh marginal performance differences among GPT-3.5 Turbo, GPT-4, and Gemini.

LLMs can also support BDD adoption by serving as training aids or initial scenario generators. Teams unfamiliar with Gherkin syntax can use LLMs to automate early drafts and learn domain-specific writing patterns. This finding supports a broader vision of LLMs as collaborative agents that enhance productivity, promote consistency, and lower barriers to adopting structured practices like BDD.

Finally, integrating LLMs into SE pipelines requires a deliberate approach. Organizations should pair prompt optimization and output validation with awareness of model strengths and limitations. This cautiousness is especially critical in regulated or safety-sensitive domains, where transparency and human oversight are essential.

8 Threats to Validity

To ensure rigor and transparency, we systematically examined threats to validity following a standard framework. Each threat is linked to our study design and the mitigation strategies used.

Internal Validity. Selection bias is a key concern, as the sample size (ten scenarios) may not reflect the full complexity of real-world test cases. To reduce this risk, a domain expert reviewed and selected scenarios from a curated dataset of 1,286 cases to ensure diversity. Instrumentation posed another risk, since automated metrics may introduce inconsistencies. We addressed this by choosing METEOR, which correlates well with expert judgments and captures linguistic nuances. The stochastic nature of LLMs also threatens internal validity due to output variability. To account for this, each prompting technique was run five times, with consistency measured using

standard deviation and coefficient of variation. Finally, expert involvement in prompt and ground-truth creation could introduce bias. We mitigated this through adherence to industry best practices, multi-expert reviews, and validation procedures.

External Validity. Our dataset comes from a single real-world system and lacks domain categorization, limiting generalizability to other software systems or contexts. Controlled experiments with fixed prompt formats and evaluation criteria ensured reproducibility but may not reflect how practitioners use LLMs in practice. Future studies should test these findings across varied domains, languages, and deployment settings. Additionally, the focus on Gemini 1.5 Pro, while representing state-of-the-art performance, limits applicability to other LLMs. Extrapolation to different models should therefore be approached with caution.

Construct Validity. METEOR was chosen for its alignment with human judgment and handling of linguistic variation. A domain expert also conducted qualitative assessments, adding insight into semantic and structural aspects. This mixed-method approach strengthens construct validity. Still, current metrics may overlook factors like business rule completeness or stakeholder relevance, highlighting the need for more targeted evaluation methods.

Conclusion Validity. Small sample sizes and test assumptions can threaten statistical validity. We selected statistical tests based on data distribution, using the Shapiro-Wilk test for normality and Bartlett's test for homoscedasticity. Parametric tests (ANOVA with Tukey post-hoc) were used when assumptions held; otherwise, non-parametric alternatives were applied. While the sample size supported analysis of intra-technique variability, it limits broader generalization. Future studies should increase the number of scenarios and repetitions. Accuracy comparisons relied on a single execution per technique, but a separate variability analysis helped assess output stability.

9 Final Remarks

This study explored the use of LLMs to generate Gherkin test scenarios from free-form natural language. We compared seven state-of-the-art models and conducted an in-depth evaluation of Gemini 1.5 Pro. By testing zero-shot, one-shot, and few-shot prompting strategies, we examined how prompt design influences output accuracy and consistency. Gemini achieved the highest METEOR score (0.84) and lowest variability under the zero-shot strategy. However, GPT-3.5 Turbo and GPT-4 Turbo also performed well—particularly in stability—when using one-shot prompts. These findings show that the best prompting strategy depends on both the model and the intended application.

Our results contribute to intelligent SE by emphasizing the importance of prompt design in producing high-quality, consistent outputs. They also challenge the assumption that in-context learning always improves performance. For rule-bound tasks like BDD scenario generation, examples can introduce unnecessary complexity. The combined use of statistical testing and variability analysis offers a replicable framework for evaluating LLMs in structured generation tasks.

From a practical perspective, the findings inform model selection in software development settings. Since performance differences were often not statistically significant, factors like API cost, toolchain integration, and cloud infrastructure may guide model choice. The strong performance of zero-shot prompting also reduces reliance on prompt engineering, making LLMs more accessible to non-experts. Additionally, generated scenarios can support BDD adoption by serving as onboarding material for teams new to the practice.

This work has limitations. The scenario sample, while curated, may not represent the full diversity of real-world software systems. Our deep dive focused on a single model, and each strategy used a fixed prompt format, which may limit generalizability. METEOR, though effective, does not capture all quality dimensions relevant to BDD.

Future research should explore adaptive prompting methods that combine zero-shot and one-shot elements, adopt more nuanced evaluation metrics, and extend the analysis to other structured tasks such as API documentation or user stories. Evaluating LLM-generated scenarios within CI/CD pipelines and assessing their end-to-end impact on testing automation also offer promising directions.

ARTIFACT AVAILABILITY

All artifacts related to this study, including datasets, scripts, and additional resources, are publicly accessible online[3]. For further information or additional assistance regarding the artifacts, please contact the first author directly.

ACKNOWLEDGMENTS

This work has been partially funded by the project ISOP BASE supported by CENTRO DE COMPETÊNCIA EMBRAPII VIRTUS EM HARDWARE INTELIGENTE PARA INDÚSTRIA – VIRTUS-CC, with financial resources from the PPI HardwareBR of the MCTI grant number 055/2023, signed with EMBRAPII.

REFERENCES

[1] Mauricio Alferez, Fabrizio Pastore, Mehrdad Sabetzadeh, Lionel Briand, and Jean-Richard Riccardi. 2019. Bridging the Gap between Requirements Modeling and Behavior-Driven Development. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*. 239–249. https://doi.org/10.1109/MODELS.2019.00008

[2] Víctor Manuel Arredondo-Reyes, Saúl Domínguez-Isidro, Ángel J. Sánchez-García, and Jorge Octavio Ocharán-Hernández. 2023. Benefits and Challenges of the Behavior-Driven Development: A Systematic Literature Review. In *2023 11th International Conference in Software Engineering Research and Innovation (CONISOFT)*. 45–54. https://doi.org/10.1109/CONISOFT58849.2023.00016

[3] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. 2023. The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet* 15, 8 (2023), 260.

[4] Oleksandr Bezsmertnyi, Nataliia Golian, Vira Golian, and Iryna Afanasieva. 2020. Behavior Driven Development Approach in the Modern Quality Control Process. *IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)* (2020), 215–218. https://doi.org/10.1109/PICST51311.2020.9467891

[5] Maria Gerliane Cavalcante and José Iranildo Sales. 2019. The Behavior Driven Development Applied to the Software Quality Test:. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. 1–4. https://doi.org/10.23919/CISTI.2019.8760965

[6] Imran Chamieh, Torsten Zesch, and Klaus Giebermann. 2024. LLMs in Short Answer Scoring: Limitations and Promise of Zero-Shot and Few-Shot Approaches. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*. Association for Computational Linguistics. https://arxiv.org/abs/2406.16143

[7] Adwait Chandorkar, Nitish Patkar, Andrea Di Sorbo, and Oscar Nierstrasz. 2022. An Exploratory Study on the Usage of Gherkin Features in Open-Source Projects. In *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. 1159–1166. https://doi.org/10.1109/SANER53432.2022.00134

[3] Available at: https://github.com/hiagonfs/bdd-scenario-evaluation

[8] D. Chelimsky et al. 2010. *The RSpec Book: Behaviour-Driven Development with RSpec, Cucumber, and Friends.* Pragmatic Bookshelf.

[9] Ednaldo DiLorenzo de Souza Filho. 2021. *Uma Abordagem para Recomendação de Casos de Teste em Projetos Ágeis Baseados no Scrum.* Ph. D. Dissertation. Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Campina Grande, Campina Grande, Brasil. http://dspace.sti.ufcg.edu.br:8080/jspui/handle/riufcg/21343

[10] Adrian De Wynter, Xun Wang, Alex Sokolov, Qilong Gu, and Si-Qing Chen. 2023. An evaluation on large language model outputs: Discourse and memorization. *Natural Language Processing Journal* 4 (2023).

[11] Ronald Díaz-Arrieta, Byron Díaz-Monroy, and Luis Castillo. 2024. Comparative Analysis of the Efficiency of Generation of Unit Test Cases: Manual Methods Versus Automation with LLM. SSRN. https://doi.org/10.2139/ssrn.5185412

[12] Nicolas Erni, Al-Ameen Mohammed Ali Mohammed, Christian Birchler, Pouria Derakhshanfar, Stephan Lukasczyk, and Sebastiano Panichella. 2024. SBFT Tool Competition 2024 – Python Test Case Generation Track. arXiv:2401.15189 [cs.SE] https://arxiv.org/abs/2401.15189

[13] DeepSeek-AI et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

[14] Muhammad Shoaib Farooq, Uzma Omer, Amna Ramzan, Mansoor Ahmad Rasheed, and Zabihullah Atal. 2023. Behavior Driven Development: A Systematic Literature Review. *IEEE Access* 11 (2023), 88007–88019. https://doi.org/10.1109/ACCESS.2023.3302356

[15] Muhammad Shoaib Farooq, Uzma Omer, Amna Ramzan, Mansoor Ahmad Rasheed, and Zabihullah Atal. 2023. Behavior Driven Development: A Systematic Literature Review. *IEEE Access* 11 (2023), 88008–88024. https://doi.org/10.1109/ACCESS.2023.3302356

[16] Abhimanyu Gupta, Geert Poels, and Palash Bera. 2023. Generating multiple conceptual models from behavior-driven development scenarios. *Data & Knowledge Engineering* 145 (2023), 102141. https://doi.org/10.1016/j.datak.2023.102141

[17] Mohsin Irshad, Ricardo Britto, and Kai Petersen. 2021. Adapting Behavior Driven Development (BDD) for large-scale software systems. *Journal of Systems and Software* 177 (2021), 110944. https://doi.org/10.1016/j.jss.2021.110944

[18] Mohsin Irshad, Jürgen Börstler, and Kai Petersen. 2022. Supporting Refactoring of BDD Specifications—An Empirical Study. *Information and Software Technology* 141 (2022), 106717. https://doi.org/10.1016/j.infsof.2021.106717

[19] Shanthi Karpurapu, Sravanthy Myneni, Unnati Nettur, Likhit Sagar Gajja, Dave Burke, Tom Stiehm, and Jeffery Payne. 2024. Comprehensive Evaluation and Insights Into the Use of Large Language Models in the Automation of Behavior-Driven Development Acceptance Test Formulation. *IEEE Access* 12 (2024), 58715–58730. https://doi.org/10.1109/ACCESS.2024.3391815

[20] Mohammed Lafi, Thamer Alrawashed, and Ahmad Munir Hammad. 2021. Automated Test Cases Generation From Requirements Specification. *2021 International Conference on Information Technology (ICIT)* (2021), 851–857. https://doi.org/10.1109/ICIT52682.2021.9491761

[21] Alon Lavie and Michael J Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation* 23, 2-3 (2009), 105–115.

[22] Rakesh Kumar Lenka, Srikant Kumar, and Sunakshi Mamgain. 2018. Behavior Driven Development: Tools and Challenges. (2018), 1032–1036. https://doi.org/10.1109/ICACCCN.2018.8748756

[23] Ali Hassaan Mughal. 2025. An Autonomous RL Agent Methodology for Dynamic Web UI Testing in a BDD Framework. arXiv:2503.08464 [cs.SE] https://arxiv.org/abs/2503.08464

[24] Gabriel Oliveira and Sabrina Marczak. 2017. On the Empirical Evaluation of BDD Scenarios Quality: Preliminary Findings of an Empirical Study. In *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW).* 299–302. https://doi.org/10.1109/REW.2017.62

[25] Gabriel Oliveira, Sabrina Marczak, and Cassiano Moralles. 2019. How to Evaluate BDD Scenarios' Quality? (2019), 181–190. https://doi.org/10.1145/3350768.3351301

[26] Ciprian Paduraru, Miruna Zavelca, and Alin Stefanescu. 2025. Agentic AI for Behavior-Driven Development Testing Using Large Language Models. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence - Volume 2: ICAART.* INSTICC, SciTePress, 805–815. https://doi.org/10.5220/0013374400003890

[27] Ciprian Paduraru, Miruna Zavelca, and Alin Stefanescu. 2025. Agentic AI for Behavior-Driven Development Testing Using Large Language Models. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025), Volume 2.* SCITEPRESS, Rome, Italy, 805–815. https://doi.org/10.5220/0013374400003890

[28] Nitish Patkar, Andrei Chiș, Nataliia Stulova, and Oscar Nierstrasz. 2021. Interactive Behavior-driven Development: a Low-code Perspective. In *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C).* 128–137. https://doi.org/10.1109/MODELS-C53483.2021.00024

[29] Indra Kharisma Raharjana, Fadel Harris, and Army Justitia. 2020. Tool for Generating Behavior-Driven Development Test-Cases. *Journal of Information Systems Engineering and Business Intelligence* 6, 1 (2020), 27–36. https://doi.org/10.20473/jisebi.6.1.27-36

[30] Indra Kharisma Raharjana, Fadel Harris, and Army Justitia. 2020. Tool for Generating Behavior-Driven Development Test-Cases. *Journal of Information Systems Engineering and Business Intelligence* 6, 1 (2020), 27–36. https://doi.org/10.20473/jisebi.6.1.27-36

[31] Shafin Rahman, Salman Khan, and Fatih Porikli. 2018. A Unified Approach for Conventional Zero-Shot, Generalized Zero-Shot, and Few-Shot Learning. *IEEE Transactions on Image Processing* 27, 11 (Nov. 2018), 5652–5667. https://doi.org/10.1109/TIP.2018.2861573

[32] June Sallou, Thomas Durieux, and Annibale Panichella. 2024. Breaking the Silence: the Threats of Using LLMs in Software Engineering. In *New Ideas and Emerging Results (ICSE-NIER'24).* ACM, 1–5. https://doi.org/10.1145/3639476.3639764

[33] Neelabh Sinha, Vinija Jain, and Aman Chadha. 2025. Guiding Vision-Language Model Selection for Visual Question-Answering Across Tasks, Domains, and Knowledge Types. In *Proceedings of the First Workshop of Evaluation of Multi-Modal Generation*, Wei Emma Zhang, Xiang Dai, Desmond Elliot, Byron Fang, Mongyuan Sim, Haojie Zhuang, and Weitong Chen (Eds.). Association for Computational Linguistics, Abu Dhabi, UAE, 76–94. https://aclanthology.org/2025.evalmg-1.7/

[34] Carlos Solis and Xiaofeng Wang. 2011. A Study of the Characteristics of Behaviour Driven Development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications.* 383–387. https://doi.org/10.1109/SEAA.2011.76

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

[36] Wenhan Wang, Chenyuan Yang, Zhijie Wang, Yuheng Huang, Zhaoyang Chu, Da Song, Lingming Zhang, An Ran Chen, and Lei Ma. 2025. TESTEVAL: Benchmarking Large Language Models for Test Case Generation. arXiv:2406.04531 [cs.SE] https://arxiv.org/abs/2406.04531

[37] Matt Wynne, Aslak Hellesøy, and Steve Tooke. 2017. *The Cucumber Book, Second Edition: Behaviour-Driven Development for Testers and Developers* (2 ed.). Pragmatic Bookshelf, Raleigh.

[38] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. 2025. Agentless: Demystifying LLM-based Software Engineering Agents. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE 2025).* Association for Computing Machinery, Trondheim, Norway. https://doi.org/10.1145/3715754

[39] Zhifei Xie and Changqiao Wu. 2024. Mini-Omni2: Towards Open-source GPT-4o with Vision, Speech and Duplex Capabilities. arXiv:2410.11190 [eess.AS] https://arxiv.org/abs/2410.11190

[40] Yunxi Yan, Biao Li, Jinyuan Feng, Yang Du, Zhichen Lu, and Manling Huang. 2023. Research on the Impact of Trends Related to ChatGPT. In *Procedia Computer Science, Volume 221: Proceedings of the 10th International Conference on Information Technology and Quantitative Management (ITQM 2023).* Elsevier B.V., 1284–1291. https://doi.org/10.1016/j.procs.2023.08.117 Peer-review under responsibility of the scientific committee of ITQM 2023.

[41] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs.CL] https://arxiv.org/abs/1904.09675