BDD4ML: A Framework for Applying Behaviour-Driven Development to Test the Performance of Supervised Machine Learning Models

Eduardo Motta
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
egmotta@inf.puc-rio.br

Marcos Kalinowski
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
kalinowski@inf.puc-rio.br

Abstract

[Context] Machine Learning (ML) systems present unique testing challenges due to their non-deterministic nature and lack of formal specifications. [Goal] In this paper, we introduce BDD4ML, a framework that adapts Behavior-Driven Development (BDD) principles for testing of ML models. [Method] We designed BDD4ML to support both classification and regression model testing through natural language clauses in Gherkin syntax, and implemented it using the Python Behave framework, allowing model specifications to be explicit and executable. The framework was evaluated through an academic observational study and an industrial focus group discussion, both assessing technology acceptance. In the observational study, graduate students used the framework to automate tests for an industrial ML model. Subsequently, in the focus group, we presented the BDD4ML framework to the ML experts responsible for developing the ML model and discussed its usage within real projects. [Results] In the classroom setting, participants were able to specify BDD4ML scenarios for successfully testing the industrial model with minimal mistakes. A majority of the students agreed on the framework's usefulness (84.62%), ease of use (76.92%), and expressed intention to adopt it (76.92%). In the industrial focus group, participants also considered BDD4ML useful and easy to use, and showed interest in adopting it. In the discussions, they highlighted BDD4ML's value for transparency, decision support, model monitoring, and collaborative requirement specification. [Conclusion] The results indicate BDD4ML as a promising approach for bringing BDD practices to ML model testing. The framework is openly available for use and extension.

Keywords

BDD, Behaviour Driven Development, ML, Machine Learning, framework

1 Introduction

Machine learning (ML) systems are now embedded in a growing number of software products across domains such as healthcare, transportation, and finance. These systems introduce new complexities to software engineering, particularly concerning testability and maintainability. Unlike conventional software, ML models often lack formal specifications and behave in probabilistic or data-dependent ways, making traditional quality assurance (QA) strategies insufficient.

Nahar *et. al* [11] conducted a meta-summary of the state-of-the-art in ML-enabled systems, highlighting key QA challenges such as managers struggling to understand the project's development, lack of specifications, and difficulties in testing. These challenges impact the ability of development teams to effectively test ML components, leading to fragile deployments and opaque behaviors.

In this context, traditional software engineering, offers Behavior-Driven Development (BDD), a widely adopted technique to improve communication between stakeholders and developers. It bridges the requirements-implementation gap using natural language specifications that double as executable tests. Despite its success in conventional systems, BDD remains largely unexplored in the context of ML-enabled system development.

In this research, we investigate how BDD can support the executable specification of supervised ML model performance. Therefore, we present BDD4ML, a novel framework that adapts BDD principles for the testing of supervised ML models. BDD4ML allows technical members of ML-enabled system teams to describe expected model behaviors using natural language clauses in Gherkin syntax. These are parsed to automatically test the ML model, providing traceable, readable, and maintainable test artifacts. Our approach enables the specification of executable tests for *Scikit-learn* classification and regression models, with support for asserting metric threshold conditions.

To evaluate the BDD4ML framework within academic and industrial contexts, we formulated two overarching research questions. *RQ1 (Academic).* How is BDD4ML perceived in an academic classroom setting in terms of participants' ability to use it effectively, challenges, and acceptance? *RQ2 (Industry).* How is BDD4ML perceived by industry professionals in terms of general impressions, coverage of relevant concerns, and acceptance as a practical tool for ML-enabled systems?

To address RQ1, we conducted an observational study in a classroom setting. In this study, participants with a background in software engineering used the framework to specify executable BDD scenarios for ML model testing, allowing us to collect objective metrics such as coverage and task success rate. We observed that participants were able to specify executable BDD4ML scenarios for successfully testing the ML model with minimal mistakes.

Subsequently, to address RQ2, we conducted a focus group in an industry context with professionals, where the framework was presented and participants discussed their perceptions, which were captured through a transcript and analyzed to identify key observations. The results indicated that participants perceived BDD4ML's value for transparency, decision support, model monitoring, and collaborative requirement specification.

In both settings, participants completed a questionnaire based on the Technology Acceptance Model (TAM) [4] to provide structured

feedback on the framework's acceptance considering the following TAM constructs: perceived usefulness (PU), perceived ease of use (PEU), and behavioral intention to use (BI). In general, participants showed high agreement rates for all three TAM constructs.

Therefore, this paper makes the following contributions:

- BDD4ML, a framework that adapts BDD to ML model testing. It introduces a readable and executable scenario-based specification approach, where model expectations are expressed in Gherkin syntax and automatically evaluated through executable tests.
- An empirical evaluation consisting of (i) an academic observational study and (ii) an industrial focus group, assessing effectiveness, usability, and acceptance. The framework enabled participants to specify executable test scenarios with high task completion rates (92.31% in a task to test a real industrial ML regression model). Participants highlighted improved clarity, transparency, and collaboration, with 84.62% agreement on perceived usefulness and 76.92% indicating intention to use the framework in practice. Practitioners provided positive qualitative feedback, emphasizing stakeholder communication benefits and straightforward ML model test automation.

Hence, the findings of our study strengthen our confidence in the value of making BDD4ML available to the community.

2 Background and Related Work

This section presents the theoretical background supporting our investigation into the application of BDD for ML systems. We begin by providing an overview of BDD, then we discuss key challenges in testing ML-enabled systems, followed by an exploration of how BDD principles can be adapted to address these challenges. Finally, we summarize related work that seeks to bridge BDD and ML development practices.

2.1 Behavior-Driven Development

BDD, introduced by North [12], emphasizes collaboration and communication across technical and non-technical teams. BDD proposes the use of a *ubiquitous language* [6], that is, a vocabulary shared by project participants for describing relevant system behavior. BDD uses a shared language to construct BDD scenarios, which act as concrete examples of expected behavior and, when connected to automation code, become executable specifications that can be repeatedly checked against the implementation.

BDD scenarios typically follow the "Given-When-Then" format, structured as follows:

- **Given**: The initial context or system state.
- **When**: The event or action that triggers the scenario.
- **Then**: The expected outcome or system response.
- **And**: An optional clause to chain multiple preconditions, actions, or outcomes.

Through its scenario-driven approach, BDD supports development practices aligned with end-user goals.

2.2 Challenges in Machine Learning System Testing

Testing ML systems introduces unique challenges not fully addressed by traditional software engineering techniques. Nahar *et al.* [11] highlight persistent difficulties in specifying and testing ML models, validating interactions across pipelines, and ensuring effective monitoring in production. Moreover, they observed a lack of standardized processes or guidelines for evaluating non-functional qualities such as fairness, security, and safety in real-world practice.

These results were reinforced by Kalinowski *et al.* [8], who also identified that practitioners often struggle with evaluating ML model performance issues, having difficulties in selecting appropriate metrics and interpreting the results. Hence, given the success of BDD in the context of traditional software development, its application to ML system development seems like a natural and promising extension.

2.3 BDD for Machine Learning

In ML projects, non-technical stakeholders often lack a deep understanding of underlying model behaviors and may hold unrealistic expectations, such as demanding near-perfect accuracy without appreciating domain-specific limitations [1].

By using BDD to express model expectations—such as acceptable accuracy thresholds criteria—in a natural language format early in the development process, one could assume that teams would be able to better align technical deliverables with stakeholder goals. Moreover, describing ML requirements as scenarios might help to mitigate miscommunications, enable earlier validation of expectations, and foster a more transparent development process.

2.4 Related Work

In regards to the development of other ML testing frameworks. Maffey et al. [10] propose the Machine Learning Test and Evaluation (MLTE) framework, a comprehensive process that spans the ML development lifecycle. MLTE formalizes requirement negotiation, metric definition, evidence-based validation, and automated reporting, offering domain-specific tools to enhance the robustness, fairness, and efficiency of ML systems. However, their research focus differs from ours, as they do not directly explore the potential benefits of using BDD in the ML context.

Regarding BDD, although it has been extensively explored in traditional software engineering, its application to ML-enabled systems remains relatively nascent. A mapping study by Binamungu and Maro [3] consolidates existing research on BDD, identifying only two contributions related to ML contexts. Deng *et al.* [5] introduced BMT, a BDD-based metamorphic testing approach that generates augmented test scenarios for autonomous driving models. Nevertheless, their approach does not concern automating model performance testing itself.

Fung *et al.* [13] proposed the Accountability-Driven Development (ADD) framework. Their work targets the integration of accountability requirements—such as transparency, explainability, and fairness—into ML system development. Using BDD-style natural language scenarios, the ADD framework motivates discussions among stakeholders to define accountability criteria early, guiding

the creation of testable requirements. However, while their framework provides a valuable starting point for discussions on BDD applied for accountability, it is only conceptual in nature and not an executable tool.

Despite these previous efforts, the intersection of BDD and ML development remains largely underexplored, presenting an opportunity for advancing the application of BDD's systematic testing practices for ML-enabled systems.

3 Overview of the BDD4ML Framework

To develop a framework tailored to user-centered concerns in ML-enabled systems, we drew inspiration from the Perspective-Based ML Task and Concern diagram [16]. This diagram identifies a comprehensive set of concerns that arise when specifying ML systems, structured across five perspectives: system objectives, user experience, infrastructure, model, and data.

This paper's scope concentrates on the *Performance Metrics* concern, due to its central role in delivering value to end users in both classification and regression problems. Therefore, BDD4ML focuses on model performance as a reasonable starting point for research on applying BDD to ML.

3.1 Framework Overview

The BDD4ML framework introduces a structured and human-readable approach to testing machine learning models by leveraging the *Behave*[1] tool and Behavior-Driven Development (BDD) principles. Figure 1 shows how the framework takes a *.feature* file as input, which specifies the model, test data, optional preprocessing and postprocessing modules, and test assertions. These components are orchestrated by the BDD4ML engine to execute the behavior-driven tests and produce test results. All of these assets should also be provided by the user as per the framework's instructions.

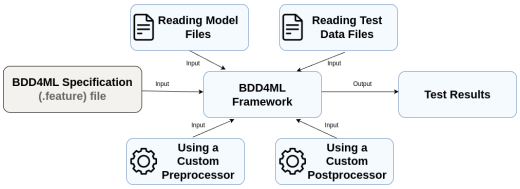

Figure 1: Overview of the BDD4ML framework.

After the specification is completed and executed, the BDD4ML framework interprets the "Given-When-Then" clauses from the *.feature* files, translating them into executable Python steps that configure the model, load the test data, apply optional transformations, and verify performance metrics. The framework then produces pass/fail outputs based on the defined assertions.

3.2 Execution Instructions

The framework is publicly available in our Open Science repository [7]. It includes comprehensive documentation with detailed instructions and a variety of example cases organized in separate folders, each containing corresponding test datasets, models, and *.feature* files. To run the framework, users should follow these steps:

[1]https://github.com/behave/behave

(1) Write a *.feature* file using "Given-When-Then" syntax. Each clause is written according to the semantics described in Section 4.
(2) Specify whether the task is a classification or a regression problem and define the desired performance metrics.
(3) Place the model file, test dataset, and optional preprocessing/postprocessing code into their respective folders in the framework directory as per the framework instructions.
(4) Move the completed *.feature* file to the *features* folder.
(5) Run the tests using Behave, which then executes the framework's code to test the specified scenarios.

After following the previous steps, the tool will execute the feature file and output test results, indicating whether the specified assertions were met.

In the following section, we detail how each clause of the BDD4ML framework functions and how it can be used to test model performance metrics.

4 BDD4ML Notation

The BDD4ML framework uses a Behave ".feature" file as the basis for specifying tests to run on the machine learning model. Hereafter, we detail the specific BBD4ML notation to be used for the 'Given', 'When', and 'Then' clauses. Figure 2 presents the 'Given' and 'When' clauses, while Figure 3 depicts the 'Then' clauses.

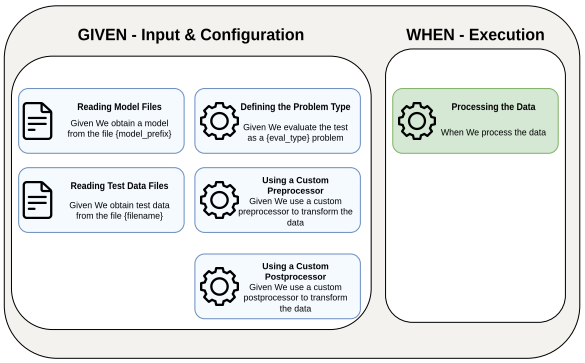

Figure 2: Overview of BDD4ML's 'Given' and 'When' clauses.

4.1 Given-When-Then Clauses

In the "Given" clause, the framework specifies the artifacts required for testing, including loading trained machine learning models from *.pkl* files and test datasets from *.csv* files, defining whether the task is classification or regression, and optionally providing custom preprocessing and postprocessing scripts to transform input data and model outputs before evaluation.

Once configurations from the "Given" step are completed, the framework proceeds to execute the model tests using the "When" clause.

Finally, the framework assesses model performance using the 'Then' clauses, which evaluate machine learning models against classification and regression metrics in a behavior-driven format. The supported evaluation metrics, illustrated in Figure 3, fall into

two main categories: *Descriptive Metrics* and *Explicit Metrics*, which we detail hereafter.

Descriptive metrics clauses enable users to evaluate model performance without explicitly referencing specific metric names. Instead, they describe the model's correctness in terms of real-world outcomes, enhancing interpretability. These clauses are available exclusively for classification problems within the framework. The descriptive classification metrics are organized into three categories—'Precision', 'Recall', and 'Accuracy'—and support both 'general' (macro/weighted average) and 'class-specific' evaluations.

Conversely, *Explicit Metrics* clauses directly reference standard performance metrics, suitable for advanced evaluations where the metric explanation is too abstract for a single BDD clause. This time, both classification and regression metrics can be displayed. Explicit classification metrics may target class-specific evaluations or be used to assess the model's overall performance across all classes.

4.2 Example Application of BDD4ML

Listing 1 presents an example *.feature* file used in BDD4ML for testing a model centered on ranking carbon emissions. It illustrates how the 'Given' clauses are used to define all necessary artifacts for testing, the 'When' clause triggers the execution, and the 'Then' clauses specify the verification of metric thresholds.

```
Feature: testing a classification model performance
    Scenario: run a performance test on a classification model
        Given We obtain a model from the file aframax_2023
        And We evaluate the test as a classification problem
        And We obtain test data from the file carbon_aframax_2023.csv
        And We use a custom preprocessor to transform the data
        And We use a custom postprocessor to transform the data

        When We process the data

        # Recall
        Then the model correctly classifies real positives of class 'D' 40
               percent of the time

        # Precision
        And the model correctly classifies predicted positives of class 'C' 40
               percent of the time

        # Accuracy
        And the model correctly classifies all classes 50 percent of the time
```

Listing 1: Feature file for testing a classification model's performance

Building on this example, the use of class-specific precision and recall metrics is particularly advantageous because the emissions problem is formulated as a classification task involving discrete ranking categories (from A to E). Certain classes, such as C and D, are especially sensitive to misclassifications, as labeling errors can result in significant consequences, including regulatory fees and legal penalties.

5 Academic Observational Study

Following the construction of the solution, we conducted an academic validation of the BDD4ML framework to address RQ1. The objective was to evaluate the framework's effectiveness, ease of use, and acceptance when applied to the specification and testing of ML models.

5.1 Goal and Research Questions

Following the Goal-Question-Metric (GQM) template [2], the evaluation goal to answer RQ1 was defined as follows.

> *Analyze* the BDD4ML framework *for the purpose of* characterization *with respect to* the challenges encountered during its usage, perceived usefulness, perceived ease of use, and behavioral intention to use *from the point of view of* software engineering graduate students *in the context of* specifying test scenarios for an industrial ML model.

Based on this evaluation goal, we decomposed RQ1 into subquestions related to the challenges and TAM-based [15] acceptance constructs:

- **RQ1.1 (Challenges—Specification):** What challenges do participants face when specifying actionable ML model test requirements using BDD4ML?
- **RQ1.2 (Challenges—Analysis):** What challenges do participants face when analyzing test results and adjusting ML model test requirements accordingly?
- **RQ1.3 (Acceptance—PU):** How do participants perceive BDD4ML in terms of usefulness?
- **RQ1.4 (Acceptance—PEU):** How do participants perceive BDD4ML in terms of ease of use?
- **RQ1.5 (Acceptance—BI):** Do participants intend to continue using BDD4ML after the experiment?

5.2 Study Design

The study followed a structured sequence. First, participants completed a characterization questionnaire to self-assess their background in software engineering, machine learning, and BDD using a proficiency scale inspired by the NIH Proficiency Scale[2]. Next, during "framework familiarization", they attended a presentation on BDD4ML and reviewed example scripts. Then, in the "task execution" phase, participants had 40 minutes to carry out the following two tasks: (1) designing BDD4ML feature files to test a regression model, and (2) adapting the model with a post-processing step to perform classification. Finally, they completed a follow-up questionnaire, including the TAM questions assessing PU, PEU, and BI, along with open-ended questions for each construct and for the framework in general. These assets target later qualitative analysis.

5.2.1 Task Design. The study was performed using a real model deployed in an industry setting, designed to estimate the carbon emissions of cargo ships. As the predicted emissions are continuous values, the model is suited for evaluation with regression metrics. However, these continuous outputs can also be translated into discrete emission rankings using post-processing modules, turning the problem suitable for classification metrics as well. This made the model appropriate for testing both types of metrics implemented in BDD4ML.

5.2.2 Participant Selection. In terms of the participant selection process, 13 graduate students participated in the study. They were selected out of convenience, as they were enrolled in a course related to Software Engineering for Machine Learning, making them suitable candidates for evaluating the BDD4ML framework. Prior to the main experiment, two pilot sessions were conducted with students from separate courses to refine the study design.

[2]https://hr.nih.gov/about/faq/working-nih/competencies/what-nih-proficiency-scale

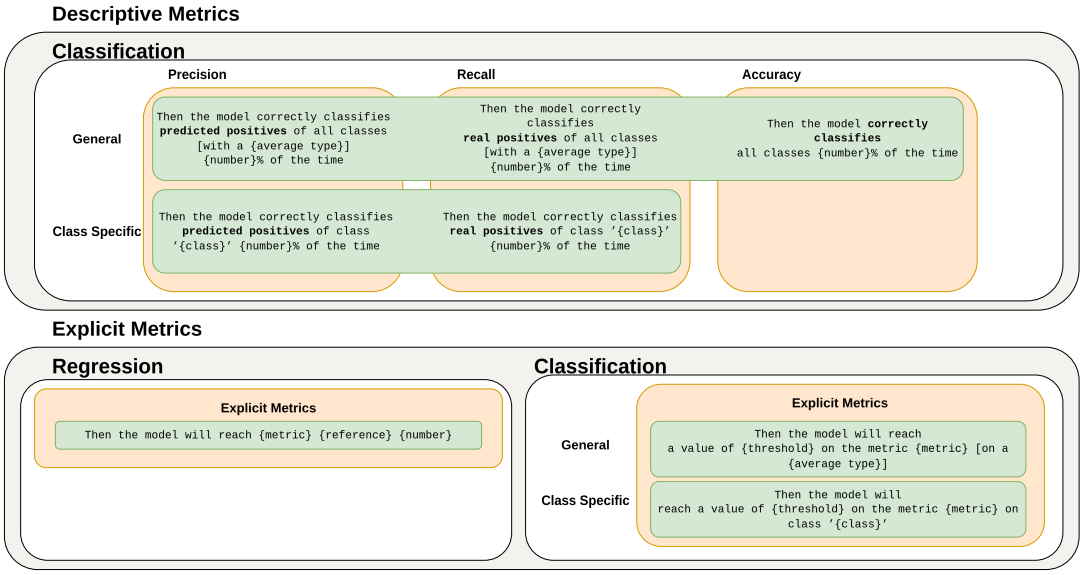

Figure 3: Overview of the 'Then' clauses in the BDD4ML framework.

5.2.3 Study Variables. Independent and dependent variables were recorded, as described in Tables 1 and 2 respectively.

Table 1: Independent Variables

Variable	Description
L-SE	Level of knowledge of software engineering
L-BDD	Level of knowledge of Behavior-Driven Development
L-ML	Level of knowledge of machine learning
L-MLS	Level of knowledge of ML specification
L-MLD	Experience developing ML systems
A-LEVEL	Academic level

All independent variables in Table 1 were measured on a 5-point Likert scale from 1 (Beginner) to 5 (Expert). Participants' academic backgrounds (A-LEVEL) were also recorded.

Table 2: Dependent Variables

Variable	Description
Q1 to Q9	TAM question agreement levels
SUCCESS-T1, SUCCESS-T2	Task success (0 or 1)
TOTAL-TIME	Task completion time in minutes

5.2.4 TAM Variables. We used the TAM questionnaire presented in [15] as follows.

- **Usefulness (PU)**
  - Q1: Using BDD4ML would improve my performance in specifying ML requirements and testing.
  - Q2: Using BDD4ML would improve my productivity when specifying ML requirements and testing.
  - Q3: Using BDD4ML would enhance my effectiveness when specifying ML requirements and testing.
  - Q4: I would find the BDD4ML useful for specifying ML requirements and testing.
- **Ease of Use (PEU)**
  - Q5: Learning to operate BDD4ML was easy for me.
  - Q6: I found it easy to use BDD4ML for specifying ML requirements and testing.
  - Q7: I found it easy to become skillful in using BDD4ML.
  - Q8: I found BDD4ML to be easy to use.
- **Intention to Use (BI)**
  - Q9: I intend to use BDD4ML when specifying ML requirements.

5.3 Results

The results presented in this subsection, along with the corresponding datasets and Jupyter notebooks used for analysis, are available in our open science repository [7].

5.3.1 Participants' Background. Figure 4 illustrates the distribution of participants' proficiency levels across relevant fields, including ML, BDD, and software engineering.

5.3.2 Task Completion Analysis. Table 3 presents the percentage of participants who successfully completed both tasks. Regarding completion time, participants took an average of 35 minutes in total to finish both of them.

Table 3: Task Completion Rates

Task	Completion Rate (%)
Task 1 (Regression Test)	92.31
Task 2 (Classification Test)	61.54

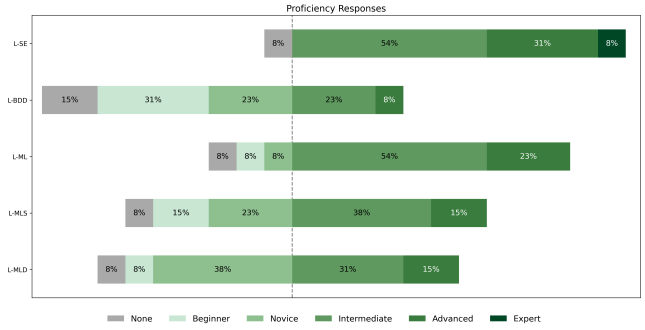

Figure 4: Proficiency distribution of participants.

It is possible to observe a noticeable decline in participant performance on Task 2 compared to Task 1. To investigate the factors contributing to this outcome, we applied the Wilcoxon-Mann-Whitney test. The results of the statistical tests are presented in Table 4.

Table 4: Wilcoxon-Mann-Whitney Test Results

Variable	U Statistic	p-value
L-SE	9.5	0.9624
L-BDD	28.5	0.1145
L-ML	31.0	0.0463
L-MLS	32.5	0.0342
E-MLD	32.5	0.0331
A-LEVEL	20.0	0.5328

The table suggests that participants with stronger knowledge and experience in machine learning (L-ML, L-MLS, E-MLD) achieved higher task completion rates, as suggested by p-values below 5%. However, these results should be interpreted as exploratory rather than confirmatory because the study involved 13 participants, a binary task-success outcome, and no adjustment for multiple comparisons. To contextualize these comparisons, we also examined each error that prevented participants from completing the tasks.

Errors made during the tasks were classified into two types: “Syntax Errors" and “Definition Errors". Table 5 summarizes all the mistakes that were made by all participants, separated by type. All but one of the ‘Definition’ type errors were committed by a single participant. In contrast, the remaining errors were isolated cases, each made by a different participant, and primarily consisted of minor syntax issues. Overall, the table indicates that most of the observed errors were relatively minor and could potentially be mitigated through tooling enhancements, such as syntax highlighting or the use of linters. These errors do not suggest that participants were unable to formulate Gherkin clauses. Nevertheless, the study design does not allow us to fully separate difficulties caused by BDD4ML from those associated with participants’ previous BDD, testing, or ML experience.

5.3.3 Answering the Research Questions. Hereafter, we analyze RQ1 based on the study results, including the open-ended questions of the follow-up questionnaire.

Table 5: Observed Error Types

Error Type	Description	Participants That Made The Error
Definition Error	Used an incorrect model name that does not exist in the test configuration.	P10
Definition Error	Incorrect class name used in classification clause.	P10 and P3
Definition Error	Classification metric value was not updated or configured properly.	P10
Definition Error	Missing required postprocessing configuration before this line.	P10
Syntax Error	Redundant use of the keyword “of” in the clause caused a syntax error.	P10
Syntax Error	Used single quotation marks instead of double quotation marks, resulting in a syntax error.	P12
Syntax Error	Improper use of spacebar introduced an extra space within the clause, resulting in a syntax error.	P3 and P9

RQ1.1 (Challenges—Specification): Participants generally found the specification of actionable ML test requirements intuitive once familiar with the BDD4ML structure. The “Given–When–Then" syntax helped them articulate model expectations in a clear and structured manner. However, some encountered challenges in composing valid clauses, particularly regarding clause syntax and the manual creation of feature files. These issues indicate that while the BDD approach helps technically oriented participants specify executable performance specifications, additional tool support, such as improved syntax validation tools and clearer feedback mechanisms, could further streamline the specification process.

RQ1.2 (Challenges—Analysis): When analyzing test outcomes and adjusting requirements, participants valued BDD4ML’s transparency in linking test failures to specific model behaviors. Nonetheless, a few users struggled to interpret framework-generated errors or to trace misconfigurations back to their source. This suggests that while the framework effectively supports iterative refinement of test scenarios, improvements such as enhanced error messages and documentation could facilitate a smoother analysis and adjustment cycle.

RQ1.3-1.5 (TAM Analysis): The TAM results, shown in Figure 5, indicated overall positive perceptions of BDD4ML. Agreement rates for each TAM category were calculated by averaging the percentage of participants who responded with “Agree" or “Slightly Agree" to each question within that category. For example, the category “Perceived Usefulness" included four questions; we first computed the agreement rate for each individual question and then averaged these values to obtain the overall agreement rate.

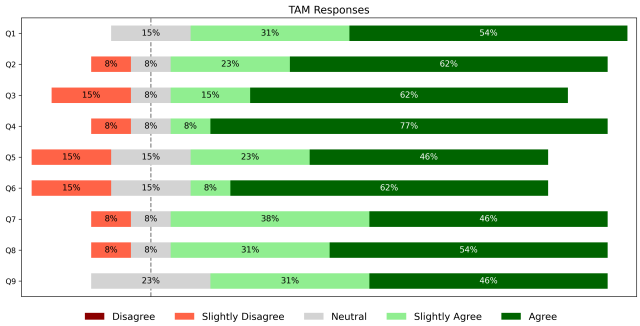

Figure 5: TAM Responses.

- **Perceived Usefulness (PU):** Agreement rates for PU questions averaged up to **84.62%** overall.
- **Perceived Ease of Use (PEU): 76.92%** of participants found BDD4ML easy to learn and use.
- **Behavioral Intention to Use (BI): 76.92%** of participants expressed intention to use the framework.

6 Industrial Focus Group

Following the observational study presented in the previous section, we conducted a focus group with data science practitioners to evaluate the acceptability of BDD4ML in a professional environment. Therefore, we presented the BDD4ML framework to the development team responsible for the carbon emission estimation model described in the observational study. The purpose was to validate whether the framework is usable and adds value in practice.

6.1 Goal and Research Questions

Using the GQM template [2], the evaluation goal for the focus group (addressing RQ2) was defined as follows:

> *Analyze* the BDD4ML framework *for the purpose of* characterization *with respect to* perceptions, coverage, perceived usefulness, perceived ease of use, and behavioral intention to use *from the point of view of* software engineers in the industry *in the context of* specifying test scenarios for an industrial ML model.

Based on this evaluation goal, we decomposed the industry-focused inquiry (RQ2) into sub-questions related to perception, coverage, and TAM-based acceptance facets:

- **RQ2.1 (General Perception—Initial Thoughts):** What are practitioners' initial thoughts on the framework?
- **RQ2.2 (General Perception—Immediate Benefit):** Do practitioners see an immediate benefit to their current process?
- **RQ2.3 (Coverage & Completeness—Right Things):** Does the framework test the right things?
- **RQ2.4 (Coverage & Completeness—Gaps):** Are there any gaps in what it evaluates?
- **RQ2.5 (Acceptance—PU):** How do participants perceive BDD4ML in terms of usefulness?
- **RQ2.6 (Acceptance—PEU):** How do participants perceive BDD4ML in terms of ease of use?
- **RQ2.7 (Acceptance—BI):** Would participants use this framework in future ML-enabled system projects?

6.2 Participant Selection

Participants were three professionals selected from the team responsible for the carbon emissions model discussed in the observational study. We selected these participants for having at least three years of experience in developing ML systems. With respect to their characterization, they self-assessed their proficiency in all ML concepts (L-ML, L-MLS, and L-MLD) as intermediate to expert.

6.3 Focus Group Design

The focus group methodology followed guidelines by Kontio *et. al* [9] and the best practices set by Ribeiro *et al.* [14]. First, we selected and characterized the participants, then we conducted the session, and finally, we analyzed the results.

We prepared a board using the MIRO collaborative platform to facilitate discussions. In this board, using virtual post-its, participants registered and openly discussed their initial thoughts (RQ2.1), benefits (RQ2.2), and coverage (RQ2.3 and RQ2.4). Thereafter, participants expressed their agreement level on the TAM constructs (RQ2.5-2.7) across five columns (Strongly agree, Partially agree, Partially disagree, Strongly disagree, and No opinion). The session was recorded and transcribed using Whisper[3].

6.4 Results

The results are organized by research question. We analyzed the open questions and summarized our findings, supporting them with illustrative quotations. For auditability purposes, the full transcript is publicly available in the Open Science repository [7].

RQ2.1 – What are practitioners' initial thoughts on the framework?

Participants expressed a predominantly positive initial impression of the BDD4ML framework, textitasizing its potential to enhance model validation, facilitate collaboration, and streamline automation.

Participant P1 highlighted three key benefits. First, the framework serves as a safeguard to prevent the deployment of underperforming models. *"The first is that we can block a bad model"*. Second, they appreciated its monitoring capability, especially for detecting data drift. *"Also for monitoring, right? Then data drift [...] becomes easier to detect"*. Third, they noted the possibility of delegating the writing of BDD clauses to less technical team members, such as Product Owners. *"The Product Owner could write the BDD clauses while the developers do the integration."*.

Participant P2 praised the framework's ability to express different performance thresholds within tests, distinguishing between base and optimal goals. They also praised that clauses return visual outputs that signal success or failure. *"If we define both a baseline and an ideal target for a metric, the framework can evaluate both and reflect this in the results — red for failures, green for successes."*

Participant P3 was particularly interested in automation and transparency. They textitasized the clarity the framework offers

[3]https://github.com/openai/whisper

through English-like clauses and raised the potential for a Boolean output that could automate decision-making. Additionally, they proposed maintaining a version history of evaluation reports. *"It helps to see what models have been used and how their metrics evolved — it's easier to track this than having to rerun everything manually."*

Future enhancements were also suggested. These included introducing a built-in model comparison feature to support deployment decisions (P1), and implementing version control for performance history retrieval (P3). The idea of issuing non-blocking warnings, similar to CI/CD pipelines that flag but do not prevent deployment, was also discussed: *"I wouldn't block it, but deploy with a warning if precision is dropping"* (P1).

RQ2.2 – Do practitioners see an immediate benefit to their current process?

Participants identified several immediate benefits of adopting the BDD4ML framework in their current workflows, particularly in areas such as model validation, deployment decision-making, and client collaboration.

Participant P1 noted the framework's potential as a safeguard that adds value during delivery. *"...It could be a kind of alert for them [customer] [...] They run it on their own infrastructure."* Participant P2 highlighted the improved transparency the framework brings to metric monitoring, noting that it clarifies expectations for both internal teams and customers and that non-technical stakeholders could use it. *"The [customer] company's staff can use it"*. Participant P3 discussed how the framework supports comparison tasks during model optimization, such as hyperparameter tuning: *"When we want to optimize a model, like test hyperparameters, with or without queue, it might be easier to compare"*.

Further enhancement suggestions included adding support for warning-level thresholds—beyond binary pass/fail logic—and integrating features such as automatic report generation, custom metric functions, and model metadata for reproducibility.

RQ2.3 and RQ2.4 – Does the framework test the right things? Are there any gaps in what it evaluates?

Participants identified several areas for improvement to increase production-readiness. For instance, they advocated for a more expressive output status system. Participant P1 textitasized the importance of distinguishing between different result categories such as success, warning, and error: *"Not just the 'true' flag, but also something like: success, warning, error. Because sometimes someone might want to proceed even with a warning"*. This would allow teams to set thresholds that do not block deployment but still alert stakeholders.

Another mentioned enhancement was the need for a richer output mechanism. Participant P3 proposed the generation of a persistent report file instead of just console outputs. They also recommended displaying all metrics, even when within expectations, to facilitate trend monitoring: *"Maybe it's better to not just show when it's outside the expected—but always show the metric in some way... a 5% drop might draw someone's attention"*.

Flexibility and extensibility were also mentioned. Participants stressed the need to support custom metrics and loaders for models not strictly aligned with the *scikit-learn* API. Participant P1 explained, *"In the same way that someone can pass a file that loads a model, they can pass a function with a custom metric"*, adding that such flexibility would broaden the framework's applicability: *"Do I natively support all metrics? No. But if you pass me yours, it expands the support"*. Additionally, the idea of adding a loader interface was revisited to allow broader data source compatibility (P1, P3). Furthermore, cloud integration and reproducibility were also cited as enhancement areas. Participant 1 suggested native integration with platforms like AWS, GCP, and Azure. Meanwhile, Participant P3 recommended including model metadata such as hyperparameters in the report to support reproducibility and comparative analyses.

RQ2.5 – How do participants perceive BDD4ML in terms of usefulness? (PU)

Table 6 presents the findings related to the TAM constructs PU, PEU, and BI. The table ommited lines With regard to PU, participants were asked to reflect on the perceived usefulness of the BDD4ML framework for real-world ML projects. All participants recognized its potential usefulness, particularly in contexts where model performance is tied to operational decision-making.

Participant P2 envisioned the framework being integrated into industrial processes: *"If it were used in a process... like a soft sensor... it could already trigger something, like send a signal saying, 'OK, we've hit the result we were aiming for'..."*. They textitasized that such a utility would reduce the need for technical personnel to interpret model results, making the system more accessible to technicians and operators. Participants P1 and P3 agreed.

Hence, participants found the framework promising for use in ML projects, particularly where decisions depend on model validation outcomes. Its greatest potential lies in its ability to communicate validation results clearly to non-developer stakeholders, thus aiding real-time decision-making in operational workflows.

Table 6: TAM focus group responses

	PU			PEU			BI		
	P1	P2	P3	P1	P2	P3	P1	P2	P3
Strongly Agree		X					X		X
Partially Agree	X		X	X	X	X	X	X	
Partially Disagree									
Strongly Disagree									
No Opinion									

RQ2.6 – How do participants perceive BDD4ML in terms of ease of use? (PEU)

To assess perceived ease of use (PEU), participants responded whether the BDD4ML framework would be easy to use for testing machine learning models. All participants selected *Partially Agree*, reflecting a shared recognition that current usability depends heavily on the user's technical proficiency.

Participant P1 textitasized that the framework, in its current form, requires programming in Python when custom pre or post-processors are needed and is therefore still best suited to individuals with development experience: *"you have to write the [pre or post processing] function in Python— meaning it already needs to be someone*

minimally specialized to write that code.". They further noted that in such cases, the benefit over standard scripting may be marginal unless low-code interfaces are introduced.

Participant P3 expanded on similar concerns, particularly regarding the ease of debugging errors in preprocessing and post-processing steps. They pointed out that the burden of handling exceptions falls on the user, as the scripts are integrated directly into the Python-based BDD execution flow.

Although participants acknowledged the structural guidance the framework provides, they unanimously agreed that its current implementation is more accessible to technically skilled users. Participant P1 summarized this view by stating: *"I think there's a lot of potential, but the way it is today, I partially agree"*.

Overall, while participants found the framework promising and structurally helpful, they stressed that its ease of use is still limited by the reliance of some features on Python scripting. They recommended future improvements—such as low-code integration and more user-friendly debugging features—to enhance accessibility for broader user profiles beyond the development team.

RQ2.7 – Would participants use this framework in future ML-enabled system projects? (BI)

This final question evaluated participants' willingness to adopt the BDD4ML framework in their future machine learning system projects. All participants expressed positive behavioral intent, with responses ranging from *Partially Agree* to *Strongly Agree*, conditional upon integration enhancements and usability improvements.

Participant P1 positioned themselves between *Agree* and *Strongly Agree*, praising the framework's support for specification and decision-making: *"It's really good for decision-making... It's just missing some details to be more useful in daily life"*. They textitasized the absence of a consolidated status output as a key limitation: *"If it returned a single result at the end, I could do so much with it... you could wrap it in a Docker container, deploy it anywhere, and it would automate things"*. Participant P1 also proposed packaging the framework as a Python library to improve reusability across different phases of the ML lifecycle.

Participant P2 selected *Partially Agree*, citing interest in using the framework in monitoring-focused contexts such as model deployments for the Carbon project. However, they conditioned their agreement on having the necessary infrastructure in place for smooth deployment and use.

Participant P3 responded with *Agree*, noting that they would use the framework even in its current form, particularly for evaluation tasks in development workflows: *"I think I'd use it to test... because sometimes, in practice, we find ways to use it we didn't think of just from the presentation"*. They added that incorporating the suggestions discussed during the session would further increase its usefulness.

In conclusion, all participants expressed interest in applying the framework in future projects. Their responses highlight its utility in supporting metric validation and stakeholder alignment. However, they also textitasized that broader adoption would be facilitated by enhancements such as a consolidated status output (e.g., `True`/`False`), library distribution, low-code support, and simplified integration into development and deployment pipelines.

7 Discussion

This section focuses on analyzing the cross-cutting insights and implications for the BDD4ML framework. The discussion here builds upon the findings for RQ1 and RQ2 to provide an integrated perspective.

7.1 Cross-cutting insights

Both groups perceived BDD4ML as potentially useful for clarifying performance expectations and communicating ML test requirements and outcomes. Although the observational study results suggest a possible relationship between ML experience and success in the more complex task, the small sample and concentration of some errors prevent us from identifying ML literacy as the main determinant of success. Nonetheless, the main source of friction arises from developer ergonomics, such as syntax and feedback mechanisms. These findings imply parallel workstreams—pedagogical materials and examples for ML-related aspects, and tooling enhancements for usability. Finally, both studies implicitly frame BDD4ML as a decision-support mechanism, highlighting that transforming scenario results into artifacts (such as reports, status codes, dashboards, or pipeline signals) is central to realizing its value.

These results demonstrate that BDD4ML is well-received by users in terms of its ability to aid in specifying and automating ML model testing. Furthermore, Turner *et al.* have highlighted in their work that the best predictor of actual usage is BI [15], which reached an agreement of 76.92% in RQ1.5. We consider this a commendable result, considering all the challenges reported in both studies. By incorporating user suggestions and improvements, such as a syntax highlighter or a low-code interface, BI for BDD4ML could increase even further.

7.2 Implications for the framework

The next stages of BDD4ML development should include a consolidated status output with configurable thresholds to support non-blocking warnings. Persistent reporting mechanisms should also be introduced to generate both machine-readable and human-readable outputs. Developer usability could be further improved through schema-aware linting, code snippets, clearer error messages, and low-code authoring interfaces. Extensibility will also be essential, enabling the use of custom metrics and data loaders while capturing metadata, such as hyperparameters and data provenance, to support reproducibility.

Additionally, further research may investigate whether the executable specification approach can be extended beyond standard performance metrics. For example, new clauses and step definitions could compare reference and current data distributions to assess data drift or encode fairness and robustness checks.

Finally, distribution and integration efforts should focus on publishing BDD4ML as a Python package with a command-line interface and API, supported by continuous integration templates and cloud deployment examples across different platforms, such as AWS, GCP, and Azure.

8 Threats to Validity

To properly consider the results of our studies and their implications for the BDD4ML framework, we identify and discuss the main

threats to the validity of our findings, according to the categories presented by Wohlin *et al.* [17].

Internal Validity: Observed effects may partially reflect confounding factors rather than BDD4ML alone. In the observational study, participants were drawn from the same course, which can introduce shared-context bias despite pilot sessions and a structured procedure to mitigate variation. Although participants received introductory training on BDD and BDD4ML before performing the task, individual differences in ML proficiency likely influenced task performance. In the focus group, heterogeneity in prior ML/BDD knowledge similarly shaped the depth and direction of feedback, even though the session design aimed at minimizing bias.

Construct Validity: Our instruments for evaluating the framework target constructs, such as usability, perceived usefulness, and task success, are measured primarily through structured task performance metrics and the TAM model. Task success was operationalized through completion rates and time taken, which provide an objective but still limited perspective on the participants' understanding and engagement with the framework. While the TAM is a well-established model, its subjective nature makes responses sensitive to individual interpretation despite standardized briefings and consistent instructions. To minimize this threat, we also collected feedback through open questions in the observational study. Regarding the creation of the framework itself, construct validity is tied to the design of the self-descriptive BDD clauses used to express ML testing concerns. The mapping between these clauses and underlying ML metrics (e.g., accuracy, precision, recall) may not perfectly align with participants' or stakeholders' interpretations of the intended concepts. The chosen formulations were inspired by formal metric definitions, which may not represent the most intuitive or accessible phrasing for non-experts. Alternative clause formulations—validated through workshops or comparative studies—could further strengthen alignment between the theoretical intent of the constructs and their practical expression in BDD4ML.

External Validity: Generalizability is limited by participant profiles and context. The observational study relied on graduate students, and the focus group included a limited sample of participants with industrial experience. These samples may underrepresent diverse organizational settings. Moreover, the application domain (*e.g.*, carbon-emissions estimation) may constrain transferability to other ML problem areas. Broader validation should include additional sessions across multiple domains.

Conclusion Validity: Inference strength is constrained by sample size and data characteristics across both evaluations. The focus group included only three participants, amplifying the potential influence of individual perspectives on qualitative interpretation. In the observational study ($n = 13$), we used the Wilcoxon–Mann–Whitney test to explore non-parametric performance differences. Although the test is non-parametric, the small sample, binary task-success measure, multiple comparisons, and concentration of several errors in a small number of participants limit the stability and interpretability of the results. We therefore treat the statistical findings as exploratory and descriptive rather than confirmatory. Replications with larger and more diverse groups are necessary to assess their reliability.

9 Conclusion

This work presented BDD4ML, a novel framework that brings BDD principles to the testing of ML models. The approach helps to bridge communication gaps among machine learning stakeholders by using readable Gherkin-based clauses to collaboratively express model requirements, while also providing a comprehensive workflow for loading models and test data, configuring evaluation pipelines, and defining performance thresholds as executable tests. It supports both descriptive specifications for human understanding and explicit metric definitions for technical rigor across classification and regression tasks.

We formulated research questions to evaluate the BDD4ML framework within academic and industrial contexts. To address them, we conducted an observational study in a classroom setting and a focus group with industry professionals.

The results indicate that BDD4ML was generally perceived as useful and easy to use, aligning with RQ1 by showing that students were capable of applying the framework in an academic setting with positive attitudes toward future use, despite some reported challenges. Likewise, the findings support RQ2 by revealing that industry professionals considered BDD4ML to adequately address relevant concerns and expressed practical acceptance and intention to adopt, highlighting BDD4ML's value for transparency, decision support, model monitoring, and collaborative requirement specification.

These results reinforce our belief that BDD Principles can contribute to ML component development, and that the framework should be shared with the community to allow its further evaluation, use, and evolution. The framework is openly available to the community in our open science repository [7].

We also discussed the implications of the received feedback for the evolution of the framework, opening avenues for future work. Besides addressing the suggested enhancements (*e.g.*, richer reporting capabilities, low-code interface), possibilities for future work include replications with larger and more diverse groups to enhance the generalization of our findings and conducting longitudinal industrial case studies on its adoption and sustained use.

Artifact Availability

The framework and the resources from our observational study and focus group are publicly available in our Zenodo repository [7].

Acknowledgments

We gratefully acknowledge the support of CAPES (Finance Code 001 and PROEX), CNPq (Grants 312275/2023-4 and 420191/2025-9), FAPERJ (Grant E-26/204.256/2024), and Instituto Kunumi for their generous support.

References

[1] Antonio Pedro Santos Alves, Marcos Kalinowski, Görkem Giray, Daniel Mendez, Niklas Lavesson, Kelly Azevedo, Hugo Villamizar, Tatiana Escovedo, Helio Lopes, Stefan Biffl, Jürgen Musil, Michael Felderer, Stefan Wagner, Teresa Baldassarre, and Tony Gorschek. 2023. Status quo and problems of requirements engineering for machine learning: Results from an international survey. In International Conference on Product-Focused Software Process Improvement. Springer, 159–174.

[2] V.R. Basili and H.D. Rombach. 1988. The TAME project: towards improvement-oriented software environments. IEEE Transactions on Software Engineering

14, 6 (1988), 758–773. doi:10.1109/32.6156

[3] Leonard Peter Binamungu and Salome Maro. 2023. Behaviour driven development: A systematic mapping study. Journal of Systems and Software 203 (2023), 111749. doi:10.1016/j.jss.2023.111749

[4] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS quarterly (1989), 319–340.

[5] Yao Deng, Guannan Lou, Xi Zheng, Tianyi Zhang, Miryung Kim, Huai Liu, Chen Wang, and Tsong Yueh Chen. 2021. BMT: Behavior driven development-based metamorphic testing for autonomous driving models. In 2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET). IEEE, 32–36.

[6] Eric Evans. 2003. Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley, Boston, MA. See Chapter 2: The Ubiquitous Language.

[7] Anonymous for Blind Review. 2025. Artifacts - BDD4ML: A Framework for Applying Behaviour-Driven Development to Test the Performance of Supervised Machine Learning Models. doi:10.5281/zenodo.17446055 Accessed: 2026-05-01.

[8] Marcos Kalinowski, Daniel Mendez, Görkem Giray, Antonio Pedro Santos Alves, Kelly Azevedo, Tatiana Escovedo, Hugo Villamizar, Helio Lopes, Teresa Baldassarre, Stefan Wagner, Stefan Biffl, Jürgen Musil, Michael Felderer, Niklas Lavesson, and Tony Gorschek. 2025. Naming the Pain in machine learning-enabled systems engineering. Information and Software Technology 187 (2025), 107866. doi:10.1016/j.infsof.2025.107866

[9] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. 2008. The Focus Group Method as an Empirical Tool in Software Engineering. In Guide to Advanced Empirical Software Engineering, Forrest Shull, Janice Singer, and Dag I. K. Sjøberg (Eds.). Springer London, 93–116. doi:10.1007/978-1-84800-044-5_4

[10] Katherine R. Maffey, Kyle Dotterrer, Jennifer Niemann, Iain Cruickshank, Grace A. Lewis, and Christian Kästner. 2023. MLTEing Models: Negotiating, Evaluating, and Documenting Model and System Qualities. In Proceedings of the 45th International Conference on Software Engineering: New Ideas and Emerging Results (Melbourne, Australia) (ICSE-NIER '23). IEEE Press, 31–36. doi:10.1109/ICSE-NIER58687.2023.00012

[11] Fariha Nahar and Muhammad Ali Babar. 2023. Quality assurance for machine learning systems: A systematic literature review. Journal of Systems and Software 200 (2023), 111563. doi:10.1016/j.jss.2023.111563

[12] Dan North. 2006. Introducing BDD. https://dannorth.net/introducing-bdd/ Accessed: 2024-08-26.

[13] C. Pang Fung, W. Pang, I. Naja, M. Markovic, and P. Edwards. 2021. Towards Accountability Driven Development for Machine Learning Systems. In CEUR Workshop Proceedings: Proceedings of the SICSA eXplainable Artificial Intelligence Workshop 2021 (SICSA Workshop on eXplainable Artificial Intelligence (XAI) 2021, Vol. 2894), K. Martin, N. Wiratunga, and A. Wijekoon (Eds.). 25–32.

[14] Talita Vieira Ribeiro, Breno Bernard Nicolau de França, Paulo Sérgio Medeiros dos Santos, Rafael Maiani de Mello, Cecília Apa, Diego Vallespir, and Guilherme Horta Travassos. 2026. Design, execution, and contextual factors shaping the benefits and drawbacks of focus groups in software engineering. Information and Software Technology 195 (2026), 108136. doi:10.1016/j.infsof.2026.108136

[15] Mark Turner, Barbara Kitchenham, Pearl Brereton, Stuart Charters, and David Budgen. 2010. Does the technology acceptance model predict actual use? A systematic literature review. Information and Software Technology 52, 5 (2010), 463–479. doi:10.1016/j.infsof.2009.11.005

[16] Hugo Villamizar, Marcos Kalinowski, Hélio Lopes, and Daniel Mendez. 2024. Identifying concerns when specifying machine learning-enabled systems: a perspective-based approach. Journal of Systems and Software 213 (2024), 112053.

[17] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2024. Experimentation in Software Engineering. Springer Nature.