Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Gonçalves Motta

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

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática, do Departamento de Informática da PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Marcos Kalinowski

Rio de Janeiro
September 2025

Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Gonçalves Motta

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

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Marcos Kalinowski
Advisor
Departamento de Informática – PUC-Rio

Profª Simone Diniz Junqueira Barbosa
Departamento de Informática – PUC-Rio

Prof. Helio Côrtes Vieira Lopes
Departamento de Informática – PUC-Rio

Rio de Janeiro, September 17th, 2025

Eduardo Gonçalves Motta

Bachelor's in Computer Science, PUC-Rio

Bibliographic data

Gonçalves Motta,Eduardo

BDD4ML: A Framework for Applying Behaviour-Driven Development to Test the Performance of Supervised Machine Learning Models / Eduardo Gonçalves Motta; advisor: Marcos Kalinowski. – 2025.

112 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2025.

Inclui bibliografia

1. Informática – Teses. 2. BDD. 3. Desenvolvimento Orientado por Comportamento. 4. ML. 5. Aprendizado de Máquina. 6. framework. I. Kalinowski, Marcos. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my father, Carlos Augusto, for instilling in me the drive for excellence and supporting me unfailingly in all my endeavors.

To my mother, Maria Lucia, for always being a patient listener, guiding me in the right direction, and teaching me to follow my heart.

And to my brother, Leonardo, who shares my values at heart and continually brings me joy as I witness his perseverance on his own path

Acknowledgments

To my adviser, Professor Marcos Kalinowski, for his guidance, encouragement, and partnership throughout the development of this work.

To Professors Simone D.J. Barbosa and Hélio Lopes, for serving on the assessment board and providing valuable feedback that contributed to the improvement of this dissertation.

To CAPES and PUC-Rio, for the support granted, without which this work would not have been possible.

To the Exacta Laboratory, for continuously challenging me to grow, while providing the tools and friendships that supported my pursuit of knowledge.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Gonçalves Motta,Eduardo; Kalinowski, Marcos (Advisor). **BDD4ML: A Framework for Applying Behaviour-Driven Development to Test the Performance of Supervised Machine Learning Models**. Rio de Janeiro, 2025. 112p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[Context] Machine Learning (ML) systems present unique testing challenges due to their non-deterministic nature and lack of formal specifications. These problems result in a gap in communication among stakeholders. To address this, we developed BDD4ML, a framework that adapts Behaviour-Driven Development (BDD) principles for ML model testing. This study adopts Gorschek *et al.*'s Technology Transfer Model (TTM) as its guiding methodology.

[Goal] In this research, we expect to bring BDD to the specification of ML-enabled systems and help to bridge the communication gap between stakeholders. The main objective is to build a framework for testing the performance of machine learning models. This framework, called BDD4ML, focuses on testing models for both regression and classification problems.

[Method] Following the TTM, the work began with a literature review on BDD frameworks, identifying current practices, gaps, and transferable lessons that informed the design of BDD4ML. The framework supports classification and regression model testing through natural language clauses written in Gherkin syntax and is implemented using the Python Behave framework. Its evaluation involved two stages: an academic observational study, where graduate students applied BDD4ML to an industrial ML model, and a static validation through a focus group involving the practitioners who built the model, discussing its applicability to real-world projects.

[Results] The literature review revealed the importance of creating a language to facilitate the communication between technical and non-technical stakeholders, as well as the value of reusable and adaptable BDD artifacts for ML contexts. In the observational study, participants successfully specified and executed BDD4ML scenarios with minimal mistakes, reporting positive perceptions of its usefulness (84.62%), ease of use (76.92%), and intention to adopt (76.92%). In the focus group, practitioners emphasized the framework's

potential for transparency, decision support, model monitoring, and collaborative requirement specification.

[Conclusion] Our evaluations indicate the technical feasibility and practical relevance of using BDD4ML in industrial settings. The results indicate that BDD4ML is a promising approach to bring BDD practices to ML model testing. The framework is openly available for use and extension, contributing to advancing responsible and collaborative testing methodologies in machine learning.

Keywords

BDD; Behaviour Driven Development; ML; Machine Learning; framework.

Resumo

Gonçalves Motta,Eduardo; Kalinowski, Marcos. **BDD4ML: Um Framework para Aplicar Desenvolvimento Orientado a Comportamentos para Testar a Performance de Modelos de Aprendizado de Maquina Supervisionado**. Rio de Janeiro, 2025. 112p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

[Contexto] Sistemas de Aprendizado de Máquina (ML) apresentam desafios de teste únicos devido à sua natureza não determinística e à falta de especificações formais. Esses problemas resultam em uma lacuna na comunicação entre as partes interessadas. Para lidar com isso, desenvolvemos o BDD4ML, um framework que adapta os princípios do Desenvolvimento Orientado por Comportamento (BDD) para testes de modelos de ML. Este estudo adota o Modelo de Transferência de Tecnologia (TTM) de Gorschek como metodologia norteadora.

[Objetivo] Nesta pesquisa, buscamos trazer o BDD para a especificação de sistemas habilitados por ML e contribuir para reduzir a lacuna de comunicação entre os stakeholders. O objetivo principal é construir um framework para o teste de desempenho de modelos de aprendizado de máquina. Esse framework, denominado BDD4ML, concentra-se em testar modelos tanto para problemas de regressão quanto de classificação.

[Método] Seguindo o TTM, o trabalho começou com uma revisão da literatura sobre frameworks de BDD, identificando práticas atuais, lacunas e lições transferíveis que orientaram o design do BDD4ML. O framework suporta testes de modelos de classificação e regressão por meio de cláusulas em linguagem natural escritas na sintaxe Gherkin e é implementado utilizando o framework Python Behave. Sua avaliação envolveu duas etapas: um estudo observacional acadêmico, no qual estudantes de pós-graduação aplicaram o BDD4ML a um modelo industrial de ML, e uma validação estática por meio de um grupo focal envolvendo os profissionais que construíram o modelo, discutindo sua aplicabilidade a projetos reais.

[Resultados] A revisão da literatura revelou a importância da criação de uma linguagem para facilitar a comunicação entre stakeholders técnicos e não técnicos, bem como o valor de artefatos BDD reutilizáveis e adaptáveis para contextos de ML. No estudo observacional, os participantes especificaram

e executaram com sucesso cenários BDD4ML com erros mínimos, relatando percepções positivas de sua utilidade (84,62%), facilidade de uso (76,92%) e intenção de adoção (76,92%). No grupo focal, os profissionais enfatizaram o potencial do framework para transparência, suporte à decisão, monitoramento de modelos e especificação colaborativa de requisitos.

[Conclusão] Nossas avaliações indicam a viabilidade técnica e a relevância prática do uso do BDD4ML em ambientes industriais. Os resultados indicam que o BDD4ML é uma abordagem promissora para levar práticas de BDD aos testes de modelos de ML. O framework está disponível abertamente para uso e extensão, contribuindo para o avanço de metodologias de testes responsáveis e colaborativas em aprendizado de máquina.

Palavras-chave

BDD; Desenvolvimento Orientado por Comportamento; ML; Aprendizado de Máquina; framework.

Table of contents

List of figures

List of tables

List of Abreviations

BDD – *Behaviour-Driven Development*

ML – *Machine Learning*

The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence.

Brooks, Frederick P., *The Mythical Man-Month:.*

1
Introduction

1.1
Context and Motivation

NAHAR et al. (2023) systematically reviewed the literature covering challenges related to the construction of systems integrated with machine learning components (ML-enabled systems). Focusing on aspects of quality assurance in the development of ML-enabled systems, the authors pointed out several common issues, including:

- Testing and debugging ML models is difficult due to the lack of specifications;
- Testing of model interactions, pipelines, and the entire system is considered challenging and often neglected;
- Testing and monitoring models in production are considered important but difficult, and often not done;
- There are no standard processes or guidelines on assessing system qualities such as fairness, security, and safety in practice.

Regarding quality assurance, the authors highlighted the great demand for better software engineering practices when testing ML-enabled systems. Their findings are consistent with an international survey, which also pointed out the lack of good software engineering practices when developing ML-enabled systems (KALINOWSKI et al., 2025). A possible reason is the complexity of creating and implementing these systems in real production environments. This complexity can be mitigated by centering the development process around stakeholder concerns. By taking an approach that prioritizes user needs, developers gain a clearer understanding of which problems to solve and the functionalities that truly matter to users. One such approach is to use Behaviour-Driven Development (BDD) (SMART; MOLAK, 2023).

Studies have supported the use of BDD to improve software engineering processes. For instance, NASCIMENTO et al. (2020) identified benefits and challenges in using BDD by monitoring the development lifecycle of mobile applications. The study concludes that BDD, although having a steep learning curve, provided developers with a better understanding of features. However, the use of BDD for ML-enabled systems has not been properly investigated yet: ALVES et al. (2023) has shown that a small group of Machine learning

developers use BDD scenarios to document ML-enabled system projects. This may be due to a lack of standard processes, tools, and references towards BDD for the machine learning environment.

1.2 Goal

In this research, we expect to bring BDD to the specification of ML-enabled systems and help to bridge the communication gap between stakeholders. The main objective is to build and evaluate a framework for testing the performance of machine learning models. This framework, called BDD4ML, will focus on testing models for both regression and classification problems.

1.3 Research Questions

To guide this research, we formulated three overarching research questions. Each of them addresses a different perspective that is essential for understanding, evaluating, and positioning the BDD4ML framework within both the academic and industrial contexts.

- RQ1: What is the state of the art of BDD frameworks in terms of principles and practices applied or adapted to ML development?
- RQ2: How is BDD4ML perceived in an academic setting in terms of participants' ability to use it effectively, challenges, and acceptance?
- RQ3: 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?

Each research question will be addressed in a dedicated chapter. Chapter 3 will address RQ1 by reviewing the state of the art on BDD frameworks and examining their relationship to ML concepts. Chapter 5 will address RQ2 through an observational study with graduate students. Finally, Chapter 6 will address RQ3 by conducting a focus group with industry professionals.

1.4 Methodology

This dissertation followed the Technology Transfer Model proposed by GORSCHEK et al. (2006), shown in Figure 1.1. This model is a structured seven-step process designed to connect academic research and industry practice in a tight development process that benefits both sides. It is a valuable asset aimed at close cooperation and collaboration between researchers and practitioners throughout the research process.

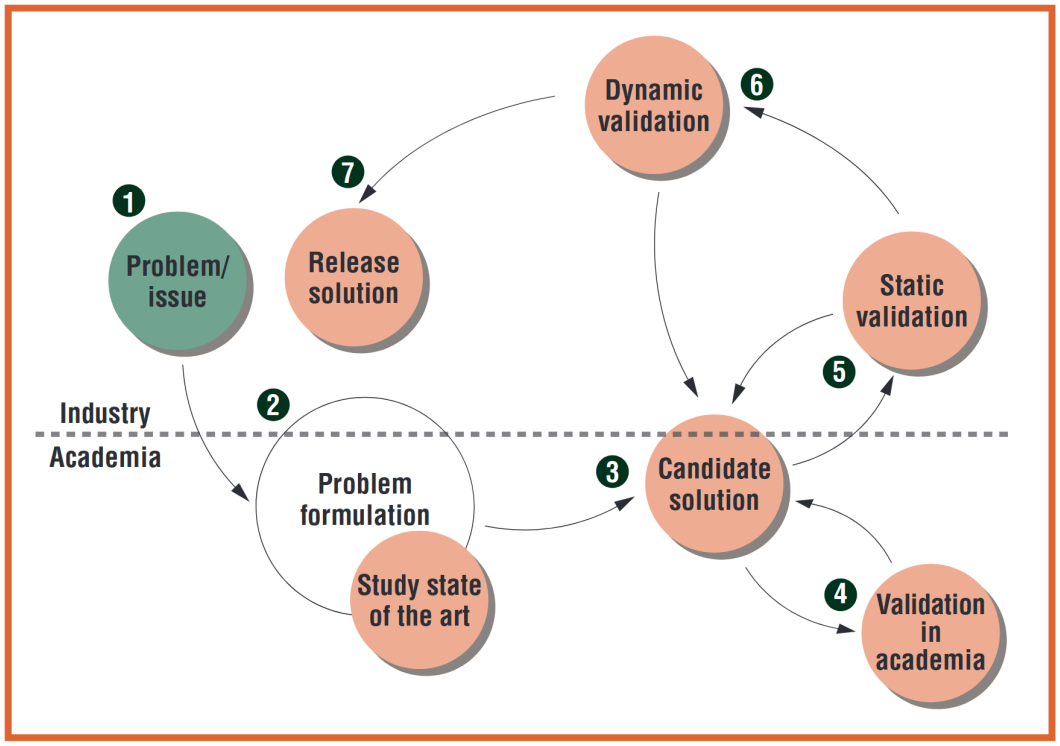

Figure 1.1: Overview of the technology transfer model. (GORSCHEK et al., 2006)

The first step involves identifying potential improvement areas by assessing current practices, observing business settings, and identifying industry needs. This ensures that the research is connected to real-world demands, fostering relevance and commitment from industry partners. Following this, a research problem is formulated in Step 2 based on gaps identified when studying the state of the art.

Once the problem is formulated, a candidate solution is proposed collaboratively with practitioners in Step 3, ensuring it is realistic and fits current practices. This solution then undergoes lab validation in Step 4 while in a controlled environment to catch early issues and gather initial feedback on usability and scalability.

Step 5 involves static validation, where the candidate solution is presented to industry personnel (*e.g.*, through seminars and interviews) to gather extensive feedback. This stage ensures upper management supports the solution, understands its potential benefits, and manages risks. Following this, dynamic validation is conducted in Step 6, where the solution is tested in real development settings, providing realistic feedback that helps refine it further.

Finally, in Step 7, the solution is released and implemented in partnering companies' official development and management processes. The Technology Transfer Model is a viable method to achieve successful technology transfer from academia to industry by emphasizing flexibility, iterative validation, and practical collaboration among multiple interest groups.

In our research, we use this model as a roadmap for developing the BDD4ML framework by documenting our findings and following the steps proposed. Steps 1 through 5 roughly equate to a chapter in the dissertation outline in section 1.6; meanwhile, steps 6 and 7 are left as future work.

1.5 Resouces

The BDD4ML framework, together with the materials from the observational study and focus group, is publicly available in our Zenodo repository (MOTTA; KALINOWSKI, 2025), enabling access to the source code and experimental results for review.

1.6 Dissertation Outline

In Chapter 1, we performed the first step of the Technology Transfer Model (TTM) by identifying potential improvement areas in the machine learning landscape. The remainder of this dissertation is organized as follows.

In Chapter 2, we summarize key theories and previous research on BDD and machine learning, identifying gaps in the literature. This is equated to the second step in the TTM.

Chapter 3 elaborates on the second step of the TTM by reviewing the literature surrounding BDD frameworks, highlighting gaps and points of improvement for our solution to tackle.

Chapter 4 elaborates on building the framework itself by outlining its main features. It is akin to the third step on the TTM.

Chapter 5 describes the fourth step of the TTM, describing the methodology that was used to evaluate the solution in an academic environment and its results.

Chapter 6 describes the fifth step of the TTM, describing the methodology that was used to evaluate the solution in a static manner in an industry setting.

Chapter 7 concludes the dissertation outline, highlighting the key takeaways from the project along with describing future work.

2
Theoretical Background and Related Work

This chapter starts by explaining the background of test-driven development (TDD) in section 2.1 and its evolution into behaviour-driven development (BDD) in section 2.2. Section 2.3 then discusses the unique challenges of testing ML systems and presents ongoing efforts to standardize their evaluation through international standards and dedicated frameworks. The chapter continues by bridging BDD with ML concepts in section 2.4. Finally, section 2.5 discusses related work that applies Behaviour-Driven Development to the field of ML-enabled systems.

2.1
Test Driven Development (TDD)

Test Driven Development (TDD) emerged in software development as a pivotal methodology to enhance code quality and foster a robust development process. Although its roots trace back even to early practices in XP and projects like NASA's Project Mercury, TDD was popularized by Kent Beck in his influential 2002 book *Test-Driven Development by Example*, which formalized the core cycle of writing tests-before-code followed by refactoring (BECK, 2003; FOWLER, 2023). This approach became a cornerstone of Extreme Programming (XP) and the broader Agile movement (CONTRIBUTORS, 2025b).

At its core, TDD centers on writing tests before the actual code implementation. Developers write small, automated unit tests that define desired behaviour, then implement code to satisfy those tests, followed by refactoring—establishing an iterative "red–green–refactor" cycle that promotes design clarity and confidence.

While TDD garnered acclaim for improving code reliability, maintainability, and developer confidence, it also revealed limitations—particularly in communication with non-technical stakeholders. TDD's technical, implementation-driven focus means it often fails to express system behaviour in business-centric terms, potentially hindering shared understanding and alignment (CONTRIBUTORS, 2025a). This communication gap spurred the development of Behaviour-Driven Development (BDD), which emphasizes language and scenarios accessible to both technical and business stakeholders.

Empirical studies reinforce this divide: whereas TDD enhances internal technical quality, BDD contributes more to stakeholder collaboration and

user satisfaction through its behaviour-oriented structure (JAMES, 2024; CUI, 2024).

2.2 Behaviour-Driven Development (BDD)

Behaviour-Driven Development (BDD) emerged as a natural complement to TDD, emphasizing collaboration and communication among stakeholders involved in software development. Created by NORTH (2006), BDD extended beyond the technical aspects of TDD by introducing a framework that encouraged the use of a ubiquitous language understandable to both technical and non-technical team members.

In the context of BDD, the notion of ubiquitous language refers to the establishment of a shared and consistent vocabulary across all stakeholders involved in the development process. Originating from Domain-Driven Design (DDD) (EVANS, 2003), this concept ensures that the terms used in conversations with business experts are the same as those employed in feature files, scenarios, and even code artefacts. By adopting such language, BDD enables the specification of requirements in plain, precise terms that are understandable to both technical and non-technical participants. This practice reduces ambiguity, fosters alignment between business goals and technical implementation, and allows scenarios to serve simultaneously as documentation and as executable specifications. For instance, when stakeholders refer to a "registered customer" or a "password reset," these exact terms are mirrored in Gherkin scenarios and supporting code, thereby helping to bridge the communication gap between business understanding and system behaviour.

BDD describes a software system's behaviour by describing each system feature as a collection of scenarios. Each scenario is described as a sequence of stages: The first stage describes the initial state of the system; the second stage describes an event or action that triggers the scenario; and the final stage describes the expected outcome or result following the action. To represent each stage, most BDD frameworks use the Gherkin language (SMART; MOLAK, 2023). This language uses simple clauses to represent each stage, commonly called "Given-When-Then" clauses. Here is an overview of each clause:

Given

- **Purpose:** Sets up the context and describes the initial state before the scenario begins.
- **Usage:** Outlines preconditions and necessary setup, including environment details, configurations, and user state.
- **Example:** "Given I am on the login page" establishes the starting point of a login feature scenario.

When

- **Purpose:** Specifies the event or action that triggers the scenario.
- **Usage:** Details the specific action by the user or system leading to an observable outcome.
- **Example:** "When I enter a correct username and password and click on the login button" describes the login attempt.

Then

- **Purpose:** Describes the expected outcome or result following the action.
- **Usage:** Articulates the changes or outcomes expected as a result of the action taken.
- **Example:** "Then I should be redirected to the homepage and I should see a message saying 'Login successful"' defines the success criteria for the scenario.

And

This clause repeats the purpose of the clause before it while maintaining a cohesive structure in terms of natural language.

Here is a simple scenario example for a login feature on a website. Note that the examples given for describing each clause are applied in these next examples:

Feature: User Login

User Story: As a User **I wish to** log in to my account **so that** I can access my personal information.

Scenario 1: Successful login with correct credentials

- **Given** I am on the login page
- **When** I enter a correct username and password
- **And** I click on the login button
- **Then** I should be redirected to the homepage
- **And** I should see a message saying "Login successful"

Scenario 2: Unsuccessful login with incorrect credentials

- **Given** I am on the login page
- **When** I enter an incorrect username or password
- **And** I click on the login button
- **Then** I should remain on the login page
- **And** I should see a message saying "Incorrect username or password"

BDD focuses on describing the behaviour of a software system through scenarios expressed in natural language, commonly termed "Given-When-Then" clauses. These scenarios, written in a language that stakeholders understand, bridge technical implementation and business requirements. By aligning the development process with these behaviour-driven scenarios, BDD facilitates a clearer understanding of the system's functionality and ensures that the software meets the intended business objectives.

The transition from Test Driven Development to Behaviour-Driven Development exemplifies the iterative nature of software development methodologies, each addressing specific challenges and evolving to meet the dynamic demands of modern software engineering.

2.3 Machine Learning Testing

Testing machine learning (ML) systems has emerged as a critical necessity to ensure the trustworthiness and reliability of these technologies. However, traditional software engineering methods are often inadequate to address the unique challenges posed by ML-enabled systems, motivating the development of dedicated methodologies and standards.

Oviedo et al. (LAMA; MONJE; VELTHUIS, 2023) emphasize the importance of establishing international standards to guide the evaluation of AI systems. Their work presents a methodology and a technological environment for evaluating the functional suitability of AI systems, aligned with the ISO/IEC 25059 standard[1]. The authors argue that adapting traditional software quality models to the specific characteristics of AI is essential, given the emergent nature of AI engineering and the inadequacy of purely traditional software engineering practices for ML systems. They also propose metrics, evaluation processes, and automated tools aimed at operationalizing the assessment of quality attributes, thereby contributing towards a standardized and repeatable evaluation of AI-based products.

Complementing this need for structured evaluation, Maffey et al. (MAFFEY et al., 2023) propose the Machine Learning Test and Evaluation (MLTE) framework. MLTE offers a comprehensive process that spans the entire ML development life cycle, incorporating requirement negotiation, metric definition, evidence-based validation, and automated reporting. By providing domain-specific tooling to define and validate system requirements programmatically, MLTE facilitates interdisciplinary collaboration and ensures that model qualities like robustness, fairness, and efficiency are not only specified but also continuously verified. This framework represents a significant step toward automating and standardizing ML testing practices, ultimately supporting the responsible deployment of ML systems in production environments.

Together, these contributions highlight ongoing efforts to formalize and systematize the evaluation of ML systems, bridging the gap between traditional software engineering principles and the evolving needs of AI system development.

2.4 BDD for Machine Learning

With the widespread adoption of BDD in many fields, it feels natural to bring it to machine learning. Many non-technical personnel creating these

[1] https://www.iso.org/standard/80655.html

systems lack the language required to understand the underlying processes of developing intelligent systems and do not fully comprehend if development goals are being met.

With BDD, all excepted system behaviour could be described as scenarios. The description of scenarios in natural language can mitigate misunderstandings among non-technical stakeholders. For instance, these stakeholders may initially hold inflated expectations that could never be met by the development team, such as a model that has impeccable accuracy. By defining the expected accuracy in a BDD statement at the beginning of the development cycle, all parties can reach an agreement and understand a reasonable accuracy threshold to be met early in the development cycle.

2.5 Related Work

BINAMUNGU (2023) published a mapping study on BDD, summarizing recent and relevant research in the area. From this review, only a few primary studies were identified that adapt BDD to machine learning contexts. FUNG et al. (2021) proposed a development framework aimed at addressing accountability issues in systems with machine learning components, while DENG et al. (2021) introduced BMT, a BDD-based metamorphic testing approach that generates augmented test scenarios for autonomous driving models. As further discussed in Chapter 3, these contributions are among only a very limited set of studies that explicitly connect BDD with machine learning, highlighting that there remains substantial room for research and improvement in this field.

2.5.1 Accountability-Driven Development (ADD)

Within the Accountability-Driven Development (ADD) Framework depicted in Figure 2.1, accountability is defined as the capacity to scrutinize, assess, or question an AI system. More specifically, it encompasses the following objectives:

(i) ensuring transparency across all stages of the system's life cycle;

(ii) demonstrating compliance with both hard laws (e.g., legal regulations) and soft laws (e.g., standards and guidelines);

(iii) enabling investigation of failures or erroneous decisions and the identification of responsible parties.

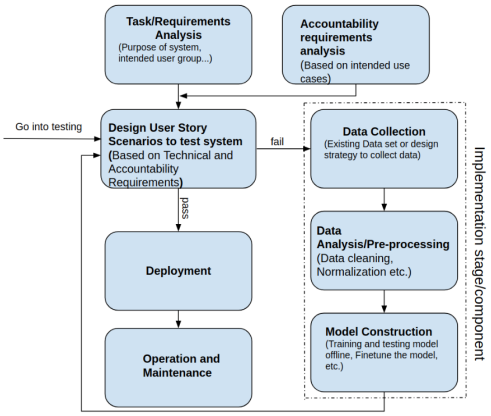

Figure 2.1: Accountability Driven Development Framework (FUNG et al., 2021)

On this basis, the authors argue that development teams and potential users must look beyond technical aspects such as performance metrics (e.g., accuracy and efficiency). Equally important are discussions about accountability criteria—including transparency, explainability, and fairness—which then inform the creation of user stories and test scenarios, initiating the testing phase.

To develop tests for machine learning using BDD paradigms, the authors use natural language clauses to describe user story scenarios and system behaviours. The authors propose initial user stories as starting points to engage all parties in further discussions, aligning accountability requirements with business objectives.

While the proposed ADD framework represents a valuable attempt to integrate accountability considerations into ML development, it remains largely conceptual and preliminary. The framework focuses primarily on the implementation stage, leaving accountability across the broader ML lifecycle (e.g., deployment, monitoring, and auditing) insufficiently addressed. Furthermore, the illustrative scenarios provided to capture requirements such as fairness and transparency are simplistic, relying on stakeholder consensus without offering mechanisms for conflict resolution or rigorous enforcement.

In addition, the framework lacks concrete tooling or automation support, which limits its practical adoption in real-world development pipelines. The examples presented are toy cases rather than empirically validated studies, leaving questions about scalability and effectiveness unresolved. As such, ADD provides an important starting point, but significant work is still required to transform it into a robust, enforceable methodology that can systematically address complex challenges like bias, accountability trade-offs, and lifecycle-

wide governance.

2.5.2 Behavior Driven Development-based Metamorphic Testing (BMT)

Deng et al. (2021) propose BMT (Behavior Driven Development-based Metamorphic Testing), a framework that leverages BDD principles to generate data augmentation scenarios for testing autonomous driving models. Instead of relying on traditional equality-based metamorphic relations, BMT enables domain experts to specify custom driving behaviours (e.g., "if a pedestrian appears in front, the vehicle should decelerate by 60–70%") using a Cucumber-like syntax. These specifications are then automatically translated into metamorphic relations and combined with transformation engines (GANs, CNNs, and image processing tools) to synthesize realistic road images with added objects or altered conditions, such as cars, bicycles, pedestrians, or weather effects.

To validate their approach, the authors implemented five behaviour templates (e.g., inserting obstacles or changing lighting/weather) and tested three speed-prediction driving models (Epoch, VGG16, and ResNet101) trained on the Cityscapes dataset. Their results show that BMT was able to systematically generate augmented test cases that revealed a high percentage of erroneous predictions across all models, with violation ratios ranging from 78% to 100%. These violations were then reviewed by six human judges, who confirmed that the majority corresponded to meaningful traffic rule violations, lending credibility to the framework's effectiveness in surfacing safety-critical failures.

Despite these contributions, BMT remains preliminary and leaves several gaps. The framework is limited in expressiveness, supporting only a small set of behaviours and simple scenarios, while real-world driving requires modeling complex, interdependent behaviours. It also lacks a user-friendly interface and comprehensive tooling, which hinders adoption by non-technical experts. Moreover, the evaluation is restricted to image-based speed prediction on synthetic transformations, leaving open questions about scalability to other driving tasks (e.g., steering, multi-agent interactions) or deployment in realistic simulation and on-road environments. Thus, while BMT demonstrates the potential of combining BDD with metamorphic testing for autonomous vehicle robustness, it has yet to evolve into a fully generalizable and lifecycle-wide testing methodology.

2.6 Concluding Remarks

Exploring BDD within the broader field of machine learning represents a novel frontier that holds promise for addressing some fundamental challenges in designing ML systems. The insightful meta-summary provided by NAHAR et al. (2023) sheds light on these challenges. From our conclusions, we suggest that BDD could offer significant relief through its structured and behaviour-focused approach.

The synthesis of BDD principles with ML system development is innovative and largely uncharted, as evidenced by the mapping study conducted by BINAMUNGU (2023). According to this study, only a very limited number of investigations explicitly adapt BDD for ML. Notably, FUNG et al. (2021) proposed the Accountability Driven Development (ADD) framework to address accountability in ML systems, while DENG et al. (2021) introduced BMT, a BDD-based metamorphic testing framework for generating augmented driving scenarios in autonomous vehicle models. The scarcity of such studies demonstrates both the pioneering nature of our research and the opportunities for further exploration in applying BDD to ML contexts. This can potentially facilitate test creation and requirements specification for ML-enabled software.

However, while the theoretical background of ML and BDD has been established, there is still a need to examine the state of the art regarding the creation of BDD frameworks as a whole. Accordingly, the following chapter presents our literature review on the subject.

3
BDD Frameworks: A Literature Review

Before presenting the methodology for developing our framework, it is important to further delve into the second step of the technology transfer model: understanding the current state of BDD frameworks. This requires a structured literature review. To this end, we built on existing studies that have proposed BDD frameworks in different domains and discussed the application of BDD principles in software development. These insights guided our approach to building a framework for testing machine learning models.

The main objective of this chapter is therefore to conduct a literature review on BDD frameworks. The findings serve as a foundation for the design and implementation of our proposed framework.

The remainder of this chapter is organized as follows: Section 3.1 describes the method used for selecting and reviewing the literature. Section 3.2 defines the research questions. Section 3.3 details the inclusion and exclusion criteria, the quality assessment checklist, the scoring system, and the evaluation procedure. Section 3.5 presents the data extraction framework used to collect information from the selected papers. Section 3.6 reports the results, and Section 3.7 concludes the chapter.

3.1
Performing the Literature Review

Numerous studies have explored the development of Behaviour-Driven Development (BDD) frameworks. Insights from these discussions on applying BDD principles to software development provide valuable guidance for shaping our own methodology. To build on this foundation, we conducted a literature review of existing BDD frameworks to identify relevant advances and practices.

One of the ways we could develop this research was by doing a systematic literature review. According to KEELE et al., the purpose of performing a systematic literature review goes beyond compiling information. It plays a crucial role in identifying gaps in current research, thereby highlighting potential areas for further investigation. A systematic literature review also establishes a comprehensive framework or background for appropriately positioning new research activities within the broader context of existing knowledge.

Although performing a systematic literature review from the ground up would be the most comprehensive way of acquiring information, BINAMUNGU already published a mapping study on BDD, summarizing all recent and

relevant research. Additionally, according to MENDES et al., when all of the most recent information on the topic is readily available on a recent mapping study, performing a new mapping study from the ground up is unnecessary. Since a mapping study on BDD has been conducted recently, we decided it would be best to perform forward snowballing on select articles of the BDD mapping study as suggested by WOHLIN et al.. Since this study focuses on BDD frameworks, we analyzed the mapping study on BDD while aiming on studies made on tools, such as frameworks.

In section 3.2, we started by defining research questions that directed the selection of the relevant papers. These questions directed our research to papers on creating BDD frameworks in any field, especially on attempts at creating BDD frameworks for machine learning since this ML focus is personally relevant to our broader research.

We have structured this review by making a thematic synthesis of BDD frameworks. We took inspiration from the guidelines set by CRUZES; DYBA. To perform a thematic synthesis of the contents of the selected papers, we defined which information codes we would collect. Information codes were strings of information in the research papers relevant to our own research. By reading papers with a clear sense of the context in which information was presented, we captured relevant information for our research. By applying the same approach to multiple papers, we labeled many strings of information with similar codes and grouped them into themes. These themes were a way to demonstrate tendencies in research within a given field, through which we produced a final text highlighting common ground among the researched papers.

Our target codes were inspired by the data extraction framework set in Section 3.5. As a result, we came up with the thematic map in Figure 3.1 with the overall themes we intended to extract from the papers. Each theme highlighted relevant information on the experience of creating BDD frameworks.

3.2 Literature Review Research Questions

To address RQ1 from Section 1.3, which investigates the state of the art of BDD frameworks and the extent to which their principles and practices have been applied or adapted to machine learning development, the following research questions were defined to guide this process. They aim to explore the foundations of BDD, its applications across software engineering domains, and its emerging role in ML-enabled systems:

RQ1: What are the core principles and practices of BDD in software development?

RQ1.1: How are these principles applied in existing BDD frameworks?

RQ2: How have BDD frameworks been adapted or applied in various software engineering domains?

RQ2.1: What lessons can be learned from these applications?

RQ3: What challenges and limitations were associated with integrating BDD frameworks into machine learning development workflows, particularly regarding findings' reliability, validity, and replicability?

RQ4: Are there specific case studies or examples where BDD frameworks or principles have been applied to machine learning projects?

RQ4.1: What were the outcomes and lessons learned from these projects?

RQ5: What are the reported benefits of applying BDD frameworks to machine learning projects?

3.3 Building On Existing Research

3.3.1 Inclusion/Exclusion Criteria

The inclusion and exclusion criteria ensured that the literature review focused on the most relevant and high-quality studies. These criteria helped in systematically selecting studies that aligned with the research objectives and questions, thereby enhancing the findings' reliability, validity, and replicability. By clearly defining what types of studies were included or excluded, we ensured a comprehensive yet focused review of the existing literature on BDD frameworks.

3.3.1.1 Inclusion Criteria

Peer-Reviewed Articles: Include only peer-reviewed articles, conference papers, and journals to ensure the credibility of the sources.

Relevance to Research Questions: The paper must address at least one of the research questions related to BDD frameworks in machine learning, specifically **RQ3**, **RQ4**, or **RQ5**.

English Language: Only papers published in English are included to ensure that the review process is manageable and consistent.

3.3.1.2 Exclusion Criteria

Duplicate Studies: Exclude duplicates or multiple reports of the same study to avoid redundancy.

Incomplete Studies: Ongoing studies, or for which full text is unavailable.

3.4 Ranking And Quality Assessment

To ensure the credibility and relevance of the studies included in this literature review, a structured approach to ranking and assessing the quality of each study was essential. This section outlined the criteria and procedures for evaluating the selected studies' methodological rigor, relevance, and overall quality. By applying a standardized quality assessment checklist, detailed in section 3.4.1, we systematically identified and prioritized high-quality studies that provided valuable insights into BDD frameworks. This checklist was graded on a scale of 0 to 3, detailed in section 3.4.2.

3.4.1 Quality Assessment Checklist

Study Relevance: Does the study directly address one or more research questions? Is the study's context related to BDD's application or adaptation in machine learning?

Methodological Rigor: Is the research design appropriate for the study's objectives? Are the methods of data collection and analysis clearly described and appropriate?

Results and Findings: Are the results presented and supported by the data? Do the findings contribute new insights or evidence on using BDD in frameworks?

Discussion and Implications: Does the discussion adequately interpret the findings in the context of existing literature? Are the practical or theoretical implications for BDD or ML clearly outlined?

Quality of Reporting: Is the study well-written and logically organized? Are all relevant aspects of the study (background, methods, results, discussion) adequately covered?

3.4.2 Scoring System

- 0: Not applicable or not addressed
- 1: Poorly addressed or of low quality
- 2: Adequately addressed or of moderate quality
- 3: Well addressed or of high quality

3.4.3 Procedures for Quality Assessment

The procedures for quality assessment are designed to ensure a thorough and consistent evaluation of the selected studies. This section details the step-by-step process used to review, score, and document the quality of each study based on predefined criteria. By following these procedures, we can systematically assess the studies' relevance, methodological rigor, and overall quality, ensuring that only the most robust and pertinent research is included in our review. This structured approach enhances our findings' credibility and provides a transparent and replicable methodology for future research.

- Initial Review: We performed an initial review of each study using the QA checklist from Section 3.4.1 to assign preliminary scores based on titles and abstracts. To pass this initial review, papers had to score at least a 3 on "Study Relevance".

– Full-Text Review: For studies that passed the initial review, we performed a full-text review to assess each study against the checklist items in detail.
– Final Scoring: We reassessed each study's preliminary score and gave it a final score that prioritized inclusion based on its total score or specific criteria most relevant to our research questions.
– Documentation: We also documented the process and scores for each study. This included noting reasons for excluding studies, which was essential for the transparency and replicability of the review. All documentation was presented in the appendix A.1.

3.4.4 Conclusion on Ranking And Quality Assessment

There were 48 papers to review in the target mapping study's combined "tools" and "machine learning" categories. After careful filtering using the procedures from section 3.4.3, we were left with only 10 papers strongly related with the research questions. Table 3.1 presents all the papers obtained.

To raise the number of research papers, we used forward snowballing in the initial selection of papers to find new studies relevant to the research. This process brought 8 new papers to evaluate. All the papers obtained were presented in table 3.2.

3.5 Data Extraction Framework

After selecting all relevant papers using the criteria from the previous sections, we systematically extracted pertinent information to inform our research on Behaviour-Driven Development (BDD) frameworks. The data extraction framework is designed to capture detailed insights from each study, ensuring a comprehensive analysis of their methodologies, findings, and implications.

This framework encompasses various elements essential for understanding the scope and impact of the selected studies. By documenting bibliographic details, study objectives, methodological approaches, key findings, limitations, and future work suggestions, we can build a robust foundation for our analysis. This approach facilitates a deeper understanding of the current state of BDD frameworks and highlights areas for further research and development. The data that was collected from the selected papers is detailed below:

– Bibliographic Details: Author(s), publication year, title, journal/conference.

- Study Objectives and Research Questions: The primary objectives or research questions the study aimed to address.
- BDD Frameworks Discussed: Names and descriptions of the BDD frameworks or tools examined.
- BDD Focus: The specific tasks or domains where BDD was applied.
- Methodological Approach: Study design, data collection, and analysis methods.
- Key Findings and Results: Main results related to the proposed solution's application, effectiveness, or challenges.
- Limitations: Limitations of the study as reported by the authors.
- Implications for Practice and Research: Authors' discussions on the implications of their findings for BDD and ML areas and further research.
- Suggestions for Future Work: Recommendations for future research directions or applications in BDD and ML.

3.6 Results of the Thematic Synthesis

To perform a thematic synthesis of the contents of the selected papers, we needed to define which information codes we would collect. We created codes using a deductive approach when coding relevant information in papers. By taking this approach, we created a "start list" of ideas to be extracted from the papers. Since our research goals were well defined, we decided on this approach instead of an inductive or integrated one. All these approaches were detailed in the work by CRUZES; DYBA.

Our target codes were inspired by the data extraction framework set in section 3.5. As a result, we came up with the thematic map in figure 3.1 with the overall themes we extracted from the papers. Each theme highlighted relevant information on the experience of creating BDD frameworks. Table 3.3 presents the details of each theme.

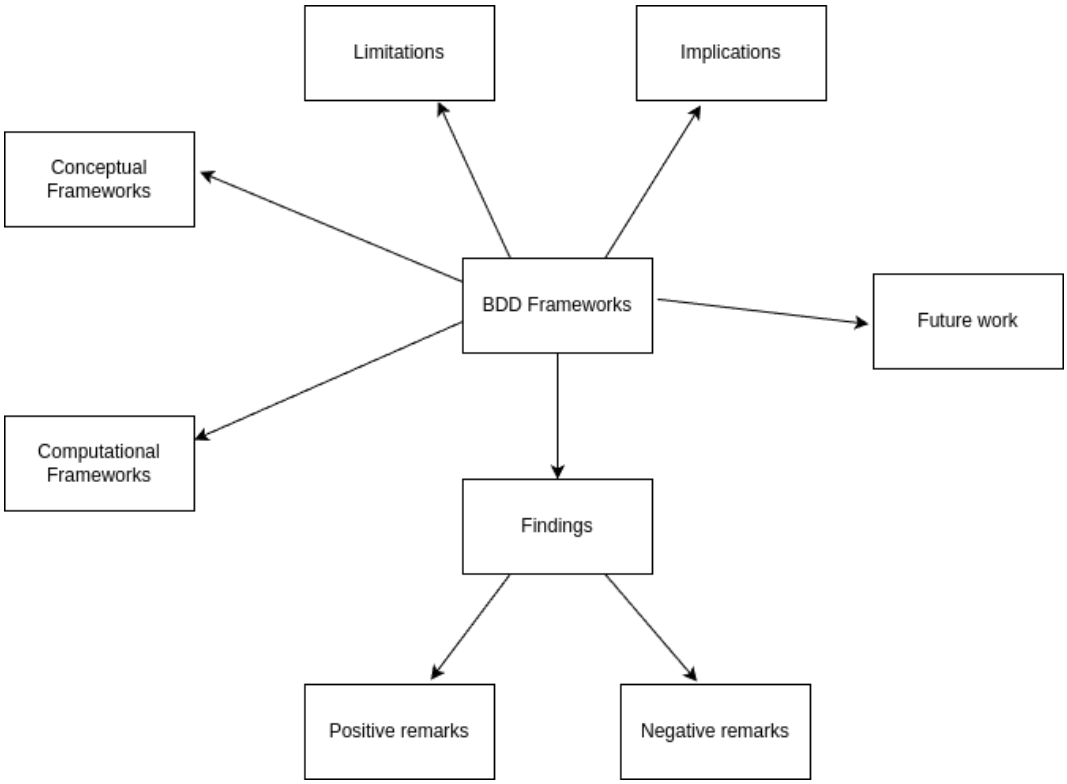

Figure 3.1: Thematic Map

3.6.1 Computational Frameworks

Various studies have explored applying and integrating computational frameworks to enhance the effectiveness and efficiency of BDD methodologies in software development and testing. Many studies detailed specific tools and frameworks actively used in BDD processes. Studies were conducted using JBehave, Cucumber, RSpec, BEAST Methodology, ScrumOntoBDD, and T-BDD, which were implemented in various programming environments and used directly in software development (FAROOQ et al., 2023). Similarly, other research also analyzed multiple BDD frameworks such as JBehave, NBehave, RSpec, MSpec, StoryQ, Cucumber, and SpecFlow, emphasizing their role as facilitators for testing software (SOLIS; WANG, 2011).

Aside from actual BDD frameworks, studies also suggested external tools such as parsers for BDD stories (SILVA; FITZGERALD, 2021) and auto-complete functionality for scenario re-usability (MUGHAL, 2024). It was also common for studies to use external tools to inform decisions on the actual framework being developed, such as BenchFlow and ContinuITy for automating load testing (SCHULZ et al., 2019).

Frameworks may also introduce their own language for representing the context they are applied to (RAHARJANA; HARRIS; JUSTITIA, 2020;

SCHULZ et al., 2019). These contexts vary in scope, ranging from blockchain applications (LIAO et al., 2017), GUI testing (BÜNDER; KUCHEN, 2019; BüNDER; KUCHEN, 2019), system performance (SCHULZ et al., 2019), web development (RAHARJANA et al., 2023), UML component development (LAZĂR; MOTOGNA; PÂRV, 2010) and finally, computer vision (DENG et al., 2021).

3.6.2 Conceptual Frameworks

Even if BDD frameworks are mostly perceived as implemented code, these implemented frameworks have a conceptual basis, meaning they are methodologies or theoretical applications that became executable tools. This section focuses on the concepts behind these BDD frameworks.

A BDD approach must focus on creating a ubiquitous language and an iterative process that guides development phases from planning to implementation (SOLIS; WANG, 2011). This process should include user stories that ensure consistency between requirements and artifacts, increasing the capacity for the code to generate tests that reflect system requirements.

Some frameworks, such as ScrumOntoBDD, conceptualize a hybrid methodology that mashes code implementation with external elements to improve the clarity and implementation of software requirements (SOUZA; SOUZA; PIRES, 2021). This implies that even computational frameworks may still be partially conceptual in nature. This hybrid nature is due to the potential integration of Scrum methodologies, BDD practices, and ontologies to enhance communication and reduce ambiguities in requirements.

One framework that deserves special attention is the Accountability Driven Development (ADD) framework. The ADD framework is built upon BDD principles and extends them to include accountability requirements in ML systems. It aims to guide developers and users in recording necessary accountability information during the design and implementation stages. The framework emphasizes transforming accountability requirements into specific scenarios, using natural language to describe them. This approach ensures that technical and non-technical stakeholders can participate in the development process (FUNG et al., 2021).

3.6.3 Implications

Developing BDD frameworks is usually driven by the expectation that their release will enhance software development practices, especially testing.

Of course, this expectation applies to any software tool. However, BDD frameworks bring their own particular characteristics and broader implications, which we outline in this section.

Studies on BDD suggest that the approach can improve customer satisfaction despite requiring more resources than similar approaches (ABUSHAMA; ALASSAM; ELHAJ, 2021; FAROOQ et al., 2023). There is an overall sentiment that, by facilitating the creation of test plans, BDD can improve communication, reduce ambiguities and the reliance on extensive technical knowledge and enhance overall software quality (SOUZA; SOUZA; PIRES, 2021; SOLIS; WANG, 2011; FAROOQ et al., 2023; ORUÇ; OVATMAN, 2016; DENG et al., 2021). These advantages can be further enhanced by the inclusion of external tools for continuous deployment (MUGHAL, 2024), testing re-usage (SILVA, 2023), and automatic creation of tests scripts (RAHARJANA et al., 2023).

As a result, BDD can potentially raise testing efficiency and accuracy, which is beneficial for developers and testers, especially in environments requiring rapid development cycles (BÜNDER; KUCHEN, 2019). BDD can also closely align development efforts with business objectives (LIAO et al., 2017). By improving the alignment between prototypes and user requirements, the proposed approach has significant practical implications for enhancing design accuracy and reducing inconsistencies in system development (SILVA; WINCKLER; TRÆTTEBERG, 2019; LAZĂR; MOTOGNA; PÂRV, 2010). The approach can significantly reduce the time and effort required to create automated GUI test cases, making it particularly valuable for developers and testers in reducing manual coding and ensuring consistency (BüNDER; KUCHEN, 2019).

Regarding BDD frameworks centered on machine learning components, current solutions are structured to incorporate accountability into the development process, which could lead to more transparent, explainable, and fair ML systems. By using natural language and involving non-technical stakeholders, the framework promotes better collaboration and understanding among team members (FUNG et al., 2021).

3.6.4 Limitations

Even if BDD frameworks are theoretically sound, they are not free from falling short of expectations. In this section, we highlight common limitations found when developing BDD frameworks. For starters, BDD frameworks are not very common, which affects the ability to generalize findings about their

impact on project success factors (ABUSHAMA; ALASSAM; ELHAJ, 2021).

A first common limitation is when frameworks do not implement all Gherkin features. This limitation reduces utility and can be circumvented using more scenarios and steps. This limitation probably exists since certain domains of expertise cannot be easily translated to the Gherkin syntax. Certain domain ontologies also require significant expertise and resources (SOUZA; SOUZA; PIRES, 2021).

It is intuitive to think that a ubiquitous language designed specifically for an ontology or domain of expertise could make a framework very useful. However, designing a comprehensive language may not always be possible. Some frameworks report that making a ubiquitous language is challenging and may hinder the ability to express complex scenarios (SCHULZ et al., 2019; LIAO et al., 2017).

It is common for frameworks that require many external tools to be difficult to implement. This detail heavily impacts the ease of adoption of a framework (MUGHAL, 2024). This limitation may exclude non-technical personnel from taking part in interviews, which is also a problem that some frameworks studied had (BÜNDER; KUCHEN, 2019).

Regarding BDD frameworks centered on machine learning components, some state-of-the-art frameworks lack detailed instructions for implementing them in specific programming languages or environments (FUNG et al., 2021). As a broader problem in ML-centered frameworks, each one addresses one specific aspect and does not extensively cover other stages of the ML lifecycle (FUNG et al., 2021; DENG et al., 2021).

3.6.5 Future Work

BDD framework developers usually highlight future work on the development of their tools, emphasizing how it should focus on the long-term impacts of BDD and optimize it to balance cost, time, and customer satisfaction (ABUSHAMA; ALASSAM; ELHAJ, 2021). There are also the added challenges of validating BDD frameworks, addressing adoption challenges, and exploring the application of BDD across various domains (FAROOQ et al., 2023).

In certain works, future work is suggested to extend existing BDD frameworks to better support planning and analysis phases and implement additional features to enhance automation (SOLIS; WANG, 2011; LIAO et al., 2017; LAZĂR; MOTOGNA; PÂRV, 2010). Such suggested automation features are: developing a high-level domain-specific language to improve the

identification of elements in software artifacts (SILVA; FITZGERALD, 2021); extending frameworks to support Scenario Outlines and Data Tables and conducting broader evaluations with end users in real settings (SILVA, 2023); and developing AI-driven autocompletion tools (MUGHAL, 2024).

Regarding improving BDD languages, future work is usually directed to improve capabilities to handle complex event scenarios and parameter combinations (SCHULZ et al., 2019). There is also the need to compare developed languages to measure how well they compare in the same domain (BÜNDER; KUCHEN, 2019). In regard to domain coverage, some BDD frameworks can be extended to multiple domains (SILVA; WINCKLER; TRÆTTEBERG, 2019).

Finally, various studies emphasize the importance of developing a graphical user interface (GUI) to enhance the usability of tools and systems (DENG et al., 2021; ORUÇ; OVATMAN, 2016; RAHARJANA; HARRIS; JUSTITIA, 2020). Currently, many of the presented tools rely heavily on command-line prompts, which can be intimidating and cumbersome for users unfamiliar with command-line operations. By contrast, a GUI offers a more intuitive and visually appealing way for users to interact with the system. However, It seems to be expected that most frameworks presented do not use a user interface since they are just MVPs.

In regard to ML-centered frameworks. future work suggests frameworks will be expanded to cover the entire lifecycle of ML systems instead of focusing on a single aspect (FUNG et al., 2021).

3.6.6 Positive Remarks

The authors made many positive remarks on BDD frameworks based on empirical evidence. BDD can potentially enhance customer satisfaction compared to traditional methods (ABUSHAMA; ALASSAM; ELHAJ, 2021). BDD also improved team communication and enhanced requirement verification (ABUSHAMA; ALASSAM; ELHAJ, 2021; FAROOQ et al., 2023; SOLIS; WANG, 2011). These positive remarks lead to enhanced software quality and structured development approaches.

The approach has improved communication, reduced ambiguities and manual effort, and facilitated a better understanding and implementation of user stories and scenarios, even for entry-level programmers (SOUZA; SOUZA; PIRES, 2021; SILVA, 2023; SILVA; FITZGERALD, 2021; RAHARJANA; HARRIS; JUSTITIA, 2020; BÜNDER; KUCHEN, 2019). Studies also report significant improvements in test writing speed and efficiency, as well as

enhanced modularity and maintainability of test suites. (MUGHAL, 2024; ORUÇ; OVATMAN, 2016; BÜNDER; KUCHEN, 2019; LAZĂR; MOTOGNA; PÂRV, 2010; BüNDER; KUCHEN, 2019). Frameworks that manage to express all test concerns while relying on a natural language to interface with test code are notable positives (SCHULZ et al., 2019). Certain frameworks manage to facilitate the specification and testing of complex behaviours (DENG et al., 2021). Finally, BDD has also successfully identified inconsistencies between user requirements and prototypes, demonstrating its effectiveness in improving design alignment and artifact quality (SILVA; WINCKLER; TRÆTTEBERG, 2019).

Regarding ML-centered frameworks, a net positive described is that BDD frameworks have the potential to properly distribute responsibility to the entire project team, including intended users, rather than placing the burden solely on ML engineers. Another positive note is that using natural language in describing scenarios allows stakeholders without technical expertise to participate actively in the development process (FUNG et al., 2021).

3.6.7 Negative Remarks

The authors also made many negative remarks about BDD frameworks based on empirical evidence. It was noted that BDD and TDD have higher time and cost associated with them than traditional testing methods (ABUSHAMA; ALASSAM; ELHAJ, 2021; ORUÇ; OVATMAN, 2016). These costs manifest as challenges, such as the learning curve associated with BDD tools, necessary collaboration, and potential test case duplication (FAROOQ et al., 2023). Other problems are the limitations related to the lack of support for certain Gherkin features and the limited scope of the proposed framework, leading to increased complexity in managing scenarios and steps (SILVA, 2023; LIAO et al., 2017; SILVA; WINCKLER; TRÆTTEBERG, 2019; DENG et al., 2021). When developing a BDD language, a developer may face challenges in creating and maintaining ontologies, highlighting a potential drawback, as this process can be complex and resource-intensive (SOUZA; SOUZA; PIRES, 2021).

There were also concerns highlighting the need for a balance between sophistication and usability, where the users could feel overwhelmed with advanced functionalities (MUGHAL, 2024). Users also remark on the non-chronological execution of test cases as a net negative in test execution (RAHARJANA et al., 2023). Study validity is also questioned with the narrow scope of participant expertise and the limited comparison with other tools, which pose significant negatives (BÜNDER; KUCHEN, 2019; BüNDER;

KUCHEN, 2019).

When it comes to framing the negative aspects of current BDD frameworks. These frameworks currently have problems solving potential ambiguities in role responsibilities within the development team, which could complicate accountability tracking (FUNG et al., 2021).

3.7 Concluding Remarks

This literature review analyzed BDD frameworks. The findings highlight the evolution of BDD from Test-Driven Development (TDD), emphasizing BDD's enhanced communication and collaboration capabilities among stakeholders. The review underscores the significance of a ubiquitous language in BDD, facilitating a clear understanding of software behaviours through natural language scenarios.

The analysis of various studies revealed the practical applications of BDD frameworks across different domains, including GUI testing, blockchain, web services, and machine learning. These frameworks have substantially improved software quality, testing efficiency, and alignment with business objectives. However, the review also identifies several limitations, such as the challenges of implementing comprehensive Gherkin features and the complexity of developing domain-specific languages.

Future work in BDD frameworks should address these limitations by enhancing automation features, supporting complex event scenarios, and developing intuitive graphical user interfaces. Additionally, expanding the applicability of BDD frameworks to multiple domains and conducting broader evaluations with end users will further validate their effectiveness and usability.

The thematic synthesis conducted in this study provides valuable insights into BDD frameworks' current state and future directions, contributing to the ongoing development and refinement of methodologies that bridge the gap between technical implementation and business requirements. More specifically, this review can assist in the development of a BDD framework centered around machine learning components.

Developing a BDD framework centered on machine learning can be facilitated by the wide range of existing BDD frameworks and examples documented in the literature. The extensive application of BDD in various fields provides a rich repository of practices and lessons learned that can be adapted to the unique challenges posed by machine learning systems. By leveraging these existing frameworks, developers can build a robust BDD framework tailored to the specific needs of machine learning projects.

Creating a ubiquitous language that simplifies the understanding of machine learning components is crucial when developing a machine learning framework. This language should bridge the gap between technical and non-technical stakeholders, ensuring that everyone involved can clearly articulate and comprehend the system's behaviours and requirements. A well-defined ubiquitous language enhances communication, reduces misunderstandings, and aligns development efforts with business goals.

Moreover, the framework should comprehensively address all components related to machine learning without becoming overly cumbersome to use. BDD artifacts, such as feature files and scenarios, should be reusable and adaptable to various contexts within the machine learning domain. This reusability streamlines the development process, promotes consistency, and facilitates the maintenance and evolution of the framework over time.

In summary, developing a BDD framework focused on machine learning benefits from existing examples and best practices, a carefully crafted ubiquitous language, and the ability to efficiently handle complex machine learning components. By addressing these key areas, the framework can enhance machine learning projects' quality, efficiency, and alignment with business objectives, ultimately advancing BDD methodologies in this rapidly evolving field.

Article Title	**Reasons for Grade 3 on "Study Relevance"**
Behaviour-driven load testing using contextual knowledge-approach and experiences	Describes the creation of a BDD framework centered at system load testing. A complete language syntax is proposed to cover many aspects of load testing.
Tool for generating behaviour-driven development test-cases	Describes the creation of a tool for generating automated GUI tests from BDD scenarios.
Towards behaviour-driven graphical user interface testing	Proposes the specification language Slang, which automatically generates executable interface test cases from BDD scenarios. These tests are then automatically integrated to low-fidelity prototypes, automating graphical user interface test cases.
Toward a service platform for developing smart contracts on blockchain in BDD and TDD styles	Presents a service platform that supports BDD-style (Behaviour-Driven Development) smart contract development, testing, and deployment.
Testing of web services using behaviour-driven development	Uses the Gherkin language to define general test cases for web services. The user can then bring a configuration file that tailors the general use test for their necessities. They also make use of the JMeter tool to automatically test all scenarios.
Ensuring the Consistency Between User Requirements and GUI Prototypes: A Behaviour-Based Automated Approach	Presents an approach based on BDD and User Stories to support the specification and the automated assessment of user requirements on low-fidelity web GUI prototypes.
Behaviour-driven development of foundational UML components	Created a BDD framework for executable UML components.
A model-driven approach for behaviour-driven GUI testing	Presented a framework that automatically generates graphical user interface test cases from BDD specifications and low-fidelity graphical user interface (GUI) descriptions.
BMT: Behaviour Driven Development-based Metamorphic Testing for Autonomous Driving Models	Created a BDD framework focused on the creation of tests for machine learning models for self-driving vehicles. This framework allows domain experts to specify custom traffic behaviours, such as introducing traffic elements or changing the weather conditions in pictures.
Towards accountability driven development for machine learning systems	Proposed creating a development framework to address accountability issues in developing systems with machine learning components.

Table 3.1: Table of Article Titles and Reasons for Grade 3 on "Study Relevance"

Table 3.2: Summary of Articles Obtained Performing Forward Snowballing

Article Title	**Reasons for Grade 3 on "Study Relevance"**
Conversion of User Story Scenarios to Python-Based Selenium Source Code for Automated Testing	This project is a converter of user stories to executable tests of the Selenium framework in Python.
Advancing BDD Software Testing: Dynamic Scenario Re-Usability And Step Auto-Complete For Cucumber Framework	This project proposes the reuse of stories in other stories, which makes the process more efficient to implement and run. All this is done with Java configurations.
ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven development	This project combines Scrum methodology with BDD and ontologies to clarify software development.
Towards a Domain-Specific Language for Behaviour-Driven Development	This study proposes the creation of a high-level domain-specific language that can be adapted to multiple domains in systems engineering, making it possible to apply BDD principles at different levels of abstraction.
Empirical Findings on BDD Story Parsing to Support Consistency Assurance between Requirements and Artifacts	This study evaluates different approaches in assuring consistency between BDD stories and software artifacts.
A Study of the Characteristics of Behaviour Driven Development	This study evaluates the features of BDD tools available at the time of writing and determines which characteristics a proper BDD solution should have.
The effect of Test-Driven Development and Behaviour-Driven Development on Project Success Factors: A Systematic Literature Review Based Study	This study evaluates the impacts of BDD and TDD in software development by making a systematic literature review on several projects. It concludes that TDD and BDD raise the overall cost of projects and only BDD has a justifiable improvement in customer experience.
Behaviour Driven Development: A Systematic Literature Review	This systematic literature review highlights many papers on the field of BDD including several framework approaches and their findings.

Theme	Description
Computational Frameworks	If the framework is implemented in a certain language and can be actually tested.
Conceptual Frameworks	If the framework has concepts that inform the code being implemented OR if the framework is just a concept to be applied in a methodology.
Limitations	What are the limitations of the study?
Implications	What are the relevant findings this study has that could impact further research?
Future Work	What are the future works proposed by the authors in regard to the development of the proposed framework or other frameworks?
Positive Remarks	What are the positive remarks of said framework?
Negative Remarks	What are the negative remarks of said framework?

Table 3.3: Themes and Their Descriptions

4
Building the Framework

In the previous chapter, we outlined the current state of the art on BDD frameworks. In this chapter, aligning with Step 3 of the technology transfer model, we built upon the insights gained from existing frameworks to design and present our candidate solution, a BDD framework specifically centered on testing the performance of machine learning models.

4.1
Defining Concerns for the Framework

We aimed to develop a ubiquitous language to capture stakeholder concerns within a user-centered framework. This language should express complex concepts in an intelligible way while emphasizing the stakeholder value provided by each framework feature.

As emphasized in the previous chapter, a BDD framework must address ML project concerns in a manner that makes the expected behaviour transparent to non-technical stakeholders. This does not require covering every possible issue that may arise in ML projects. Instead, the framework should concentrate on the behavioural concerns that truly matter to the stakeholders. One effective way to identify which issues are most relevant is to examine the existing literature on challenges commonly reported in ML projects.

Therefore, we have taken inspiration from VILLAMIZAR et al.'s Perspective-Based ML Task and Concern diagram in Figure 4.1 (2024). This diagram provides a holistic view of ML-enabled systems, making producing system descriptions and requirements easier. The diagram is a visual depiction of how different aspects interact and relate to ML projects. It gives a broad view of how diverse viewpoints influence the tasks involved, considering the unique concerns of each task. Moreover, it emphasizes the engagement of multiple stakeholders, offering expertise and perspectives during the development phase.

Since BDD's focus is on aligning technical development with business goals and end-user needs, we need to determine which concerns in the diagram could be most relevant to the customers and end users of ML-enabled systems. Something like "which algorithm to use" might not be relevant for the customer and can be freely decided on by the developers. However, something like throughput, the speed at which an inference is made, may be relevant for the customer, given that it directly impacts the user experience.

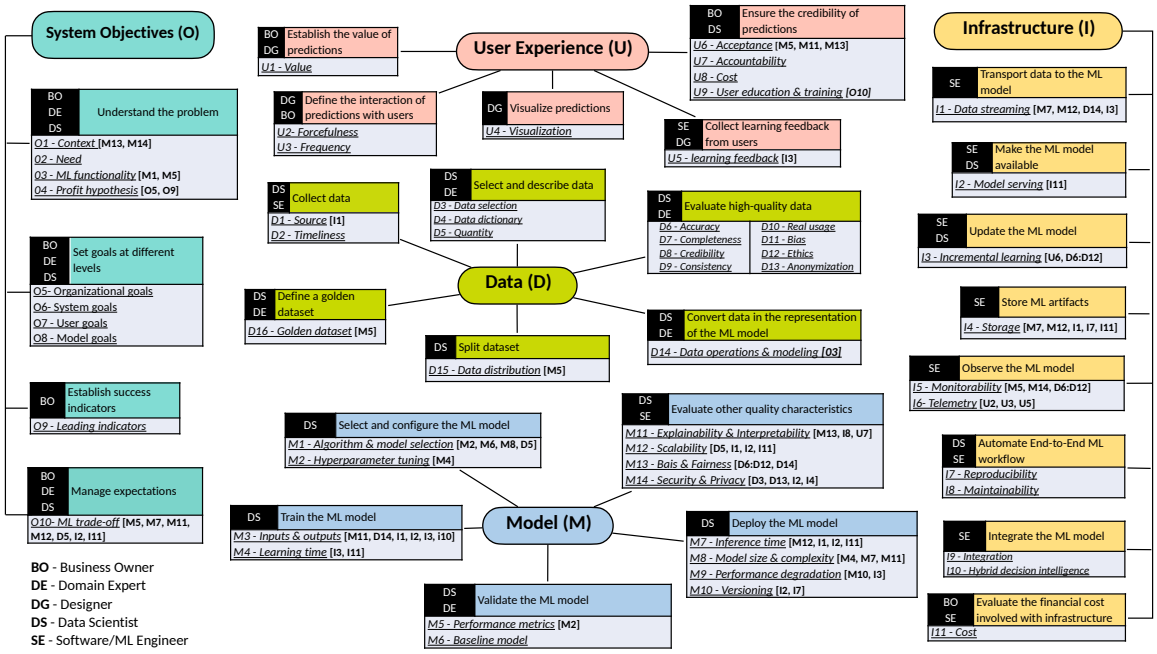

Figure 4.1: Perspectives and Concerns (VILLAMIZAR et al., 2024)

From the concerns presented in the diagram, we identified six categories that reflect issues that affect non-technical stakeholders when developing machine learning systems. These categories are Model Performance, Accountability, Explainability, System Performance, Data, and Legality. Each subsequent section introduces a definition relevant to the corresponding category and explicitly connects it to the related concerns identified in the work of VILLAMIZAR et al. (2024).

4.1.1 Model Performance

Model performance is most directly evaluated through the use of performance metrics (*cf.*, concern M5 of Figure 4.1), which provide measurable criteria to determine how effectively a trained model accomplishes its intended task. Since no single metric captures all dimensions of performance, different problem types demand different measures. For example, classification problems commonly rely on accuracy, precision, recall, and F1 score, while regression tasks make use of metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), or R-squared (R^2). Each of these offers a distinct perspective: accuracy highlights overall correctness, precision and recall emphasize

trade-offs between false positives and false negatives, and error-based metrics quantify the magnitude of deviations in regression outputs.

Interpreting these metrics requires care, as optimizing for one may mask weaknesses in another. For instance, a model achieving high accuracy might still be biased or overfitted if class imbalance is present. Similarly, minimizing error in regression may come at the expense of generalization to unseen data. Therefore, practitioners must not only select metrics suited to the problem but also weigh their trade-offs in the broader system context.

Ultimately, performance metrics are central to ensuring the reliability and effectiveness of machine learning models. By grounding evaluation in well-defined measures while recognizing their limitations, stakeholders gain a clearer understanding of a model's strengths, weaknesses, and readiness for deployment.

4.1.2 Accountability

As explained in Section 2.5, accountability refers to the capacity to scrutinize, assess, or question an AI system. Within the context of ML-enabled systems, accountability ensures that stakeholders can identify responsibility for unexpected or undesired outcomes. This concern is not only about assigning blame but also about enabling transparency and traceability, so that system decisions can be explained, justified, and improved over time.

In PerSpecML, accountability is part of the User Experience Perspective, specifically captured in concern U7 (*cf.* Figure 4.1). While the Accountability Driven Development article first introduced the idea of embedding accountability into software practices (FUNG et al., 2021), its treatment in the ML context remains superficial. By incorporating accountability explicitly as a concern, we acknowledge the importance of providing mechanisms for users and organizations to question predictions, understand decision rationales, and ensure that corrective actions can be taken. This integration strengthens trust in ML-enabled systems and promotes responsible adoption.

4.1.3 Explainability

Explainability in AI systems refers to understanding and interpreting the decisions and outputs generated by artificial intelligence algorithms. It provides insights into how these algorithms arrive at specific conclusions or predictions, ensuring transparency and fostering trust among users, stakeholders, and regulators. Within the context of ML-enabled systems, explainability plays

a crucial role in building confidence and promoting adoption, particularly in sensitive domains such as healthcare, finance, and law, where understanding why a model produces a specific output is essential for ethical and legal reasons.

This concern, captured in M11 of the Model Perspective (*cf.* Figure 4.1), emphasizes the need for models to summarize the reasons behind their inferences and to offer transparency regarding their internal logic (MISHRA; STURM; DIXON, 2017; GARCíA; AZNARTE, 2020). While some models, such as decision trees or linear regressions, are inherently interpretable, others, like deep neural networks, require additional techniques to enhance explainability. These include post-hoc methods such as LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive Explanations), as well as visualization techniques like heatmaps, graphs, or feature importance scores that help uncover which inputs most strongly influenced the outcome.

However, achieving explainability is not without trade-offs. In many cases, there is a tension between the predictive power of complex models and the interpretability of simpler ones. Organizations must therefore balance the need for accuracy with the demand for transparent and understandable outputs. Addressing explainability as a model concern ensures that ML-enabled systems remain not only performant but also accountable, trustworthy, and aligned with societal and regulatory expectations.

4.1.4 System Performance

System performance in AI systems refers to the overall efficiency and effectiveness of the computational setup required to achieve the desired objectives. This encompasses not only the algorithms but also the hardware and software components that support their execution. Evaluating system performance is essential to ensure that AI-enabled systems operate optimally and meet predefined benchmarks in real-world scenarios.

Within the PerSpecML framework, system performance spans concerns from both the Infrastructure and Model perspectives. From the Infrastructure perspective, performance is shaped by how the model is served (*cf.* concern I2 in Figure 4.1), the need to monitor outputs and data (*cf.* I5), and the ability to reproduce results reliably under different environments (*cf.* I7). It also involves maintainability (*cf.* I8), where system components must adapt to evolving requirements, and cost considerations (*cf.* I10), since even high-performing models can become impractical if computational resources are inefficiently utilized. From the Model perspective, performance concerns include the acceptable time required to train (*cf.* M4) and execute models (*cf.* M7), as well as the scalability

of the system to handle larger datasets or increased workloads (*cf.* M12).

Three central aspects help evaluate system performance: *throughput*, *latency*, and *resource utilization*. Throughput measures the rate at which an AI system processes data or tasks, which is critical in real-time or high-volume applications. Latency refers to the delay between input and system response, a decisive factor in time-sensitive domains such as autonomous driving or healthcare diagnostics. Resource utilization captures how efficiently the system consumes hardware resources (e.g., CPU, GPU, memory), guiding optimization strategies for cost-effective performance.

By addressing these concerns in tandem, system performance becomes a holistic measure that ensures AI-enabled systems are not only accurate but also efficient, scalable, and sustainable in practice. This integration highlights the necessity of aligning computational efficiency with broader operational and organizational goals.

4.1.5 Data

Data quality is a cornerstone of AI systems, dictating their accuracy, reliability, and effectiveness. It encompasses the completeness, consistency, accuracy, relevance, and credibility of the data used to train, validate, and operate AI models. High-quality data ensures that models make accurate predictions or classifications, whereas poor-quality data can lead to biased models, erroneous conclusions, or unreliable outputs. Within the PerSpecML framework, these issues are captured in concerns such as accuracy (*cf.* D6 in Figure 4.1), completeness (*cf.* D7), credibility (*cf.* D8), real usage (*cf.* D9), bias (*cf.* D10), and consistency (*cf.* D11). Furthermore, the notion of a golden dataset (*cf.* D16) provides a validated reference for monitoring and maintaining data quality across the lifecycle of ML-enabled systems.

In the context of model testing, high-quality data is equally crucial. Test datasets must mirror the quality of training data while also being distinct enough to evaluate generalization to unseen cases. To achieve this, strategies such as cross-validation, stratified sampling, and time-based splitting are often employed. Test datasets should be sufficiently large to assess model performance and must represent all relevant classes or categories in the problem space, ensuring fairness and robustness in evaluation.

Sustaining data quality requires continuous monitoring and refinement. Feedback derived from model predictions or user interactions can reveal issues, enabling iterative improvements in datasets. Automated checks and anomaly detection algorithms further streamline the process of identifying inconsisten-

cies or errors. By addressing these concerns, organizations can ensure that data remains a reliable foundation for AI development and deployment.

In summary, ensuring high-quality data is fundamental to the success of AI systems. By systematically addressing data-related concerns, practitioners strengthen the reliability and trustworthiness of AI models, thereby enabling them to deliver accurate, ethical, and meaningful insights.

4.1.6 Legality

Legal concerns in AI systems encompass a broad array of issues arising from their development, deployment, and operation. As AI technologies become increasingly embedded in societal functions, legal challenges emerge in areas such as privacy, liability, bias, accountability, and intellectual property rights. Within the PerSpecML framework, these concerns are reflected in the need for system integration with legal and regulatory requirements (*cf.* I9 in Figure 4.1), safeguards for privacy and security (*cf.* M14), fair and unbiased model behavior (*cf.* M13), and ethical data handling practices such as anonymization and prevention of societal harm (*cf.* D12 and D13).

AI systems often rely on large volumes of data, which raises concerns about individual privacy and compliance with regulations such as GDPR in Europe and LGPD in Brazil. Addressing these concerns requires strict anonymization practices and ethical guidelines for data management. Beyond data protection, legal frameworks also impose requirements for transparency and accountability, ensuring that individuals can understand how their data is processed and how automated decisions that affect them are made. These requirements directly connect to accountability mechanisms that attribute responsibility among developers, operators, and organizations.

Another critical legal dimension involves liability and intellectual property. AI-generated outputs in creative or technical domains raise unresolved questions about ownership, patents, and copyright. Determining responsibility for the consequences of AI-driven decisions is similarly complex and necessitates collaboration between policymakers, technologists, and legal experts. Furthermore, adherence to international standards, such as ISO 25059 [1], provides a foundation for aligning AI systems with legal, ethical, and quality requirements by emphasizing transparency, fairness, and accountability.

By situating legality as a cross-cutting concern across infrastructure, model, and data perspectives, we highlight its role in promoting trust and societal acceptance. Addressing legal concerns requires more than compliance

[1] https://iso25000.com/index.php/en/iso-25000-standards/iso-25059

with regulations: it demands the integration of ethical guidelines and proactive governance structures. Balancing innovation with safeguards that protect individuals and communities is therefore essential for the responsible development and deployment of AI technologies.

4.1.7 Considerations about the concerns

Regarding the separation of concerns across these six topics, it is important to recognize that, since BDD is inherently test-centered, some categories can be directly addressed through measurable testing metrics. Others, however, may require more indirect approaches to establish meaningful evaluation criteria. Another challenge lies in potential overlap between categories. For example, legal concerns often intersect with accountability—both touching on issues such as responsibility and compliance—while still retaining unique aspects of their own, such as intellectual property rights.

Out of the six defined categories, we focus on model performance in the scope of this dissertation. Hence, our work tackles issues related to communicating model metrics to non-technical stakeholders. These metrics should be communicated in an easily understandable way so as not to alienate non-technical stakeholders.

4.2 Functional Requirements

This section describes the functionalities of the BDD4ML framework. Each functionality is represented by a BDD clause, which allows users to specify behaviour in natural language. Clauses begin with one of three keywords: `Given`, `When`, or `Then`. The keyword `And` can be used to extend or repeat the context of the preceding clause.

Example:

```
Given we obtain a model from the file earthquakeKNN
And we obtain test data from the file test_data_earthquake.csv
```

Each keyword indicates a specific type of clause: - `Given` clauses define the initial configuration or context. - `When` clauses describe the execution of the test. - `Then` clauses specify the expected outcomes or evaluations.

Figure 4.2 provides an overview of the `Given` and `When` clauses, while Figure 4.3 illustrates the `Then` clauses.

Users must write clauses in a `.feature` file, following the execution flow defined in the framework's diagram. The following sections describe each clause type in detail, along with their associated behaviours.

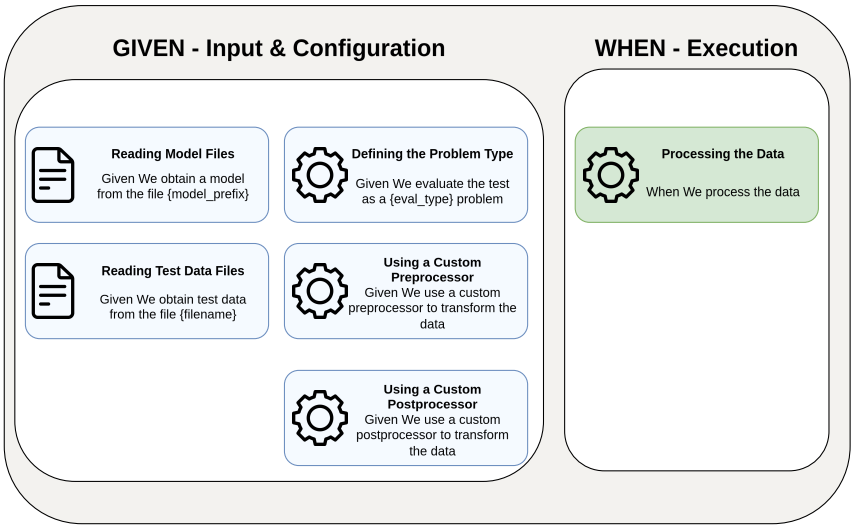

Figure 4.2: Overview of `Given` and `When` Clauses

4.2.1 Given Clauses

Reading Model Files The framework reads machine learning models stored as `pickle` files. The model files must be named with the suffix `_model.pkl` and placed in the `estimators` directory at the root of the project. If a scaler is associated with the model, it must have the same prefix and the suffix `_scaler.pkl`. A configuration file (`.json`) with the same prefix must also be provided in the `model_configs` directory, specifying input features and targets.

```
Given We obtain a model from the file {model_prefix}
```

`model_prefix` - Prefix of the model, scaler, and configuration files (excluding file extensions).

Defining the Problem Type The type of evaluation must be defined as either classification or regression.

```
Given We evaluate the test as a {eval_type} problem
```

`eval_type` can be either `classification` or `regression`.

Reading Test Data Files The framework reads test data stored in CSV format, located in the `test_data` directory at the project root.

```
Given We obtain test data from the file {filename}
```

`filename` - Name of the CSV file containing test cases.

Using a Custom Preprocessor (Optional) If specific preprocessing is needed, a preprocessor class must be implemented and imported. The following clause can then be used to activate it.

```
Given We use a custom preprocessor to transform the
data
```

Using a Custom Postprocessor (Optional) If specific postprocessing is needed, a postprocessor class must be implemented and imported. The following clause can then be used to activate it.

```
Given We use a custom postprocessor to transform the
data
```

4.2.2 When Clauses

Processing the Data Once the model, configuration, and test data are provided, the framework processes the data and stores the results for evaluation.

```
When We process the data
```

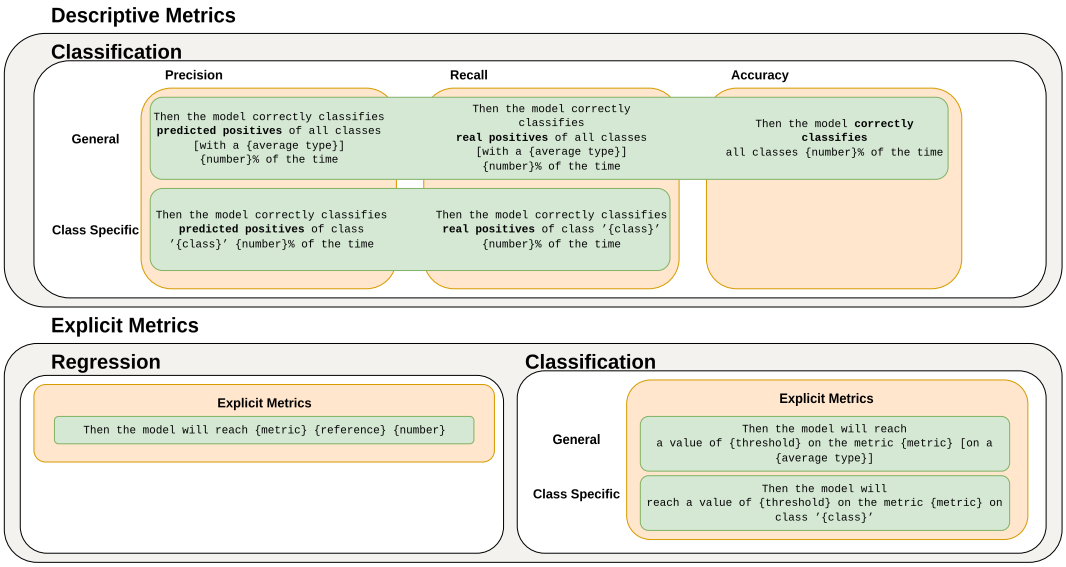

Figure 4.3: Overview of `Then` Clauses

4.2.3 Then Clauses

To make metrics as easy to understand as possible, we should avoid directly naming the metric in the BDD "Then" clause. Referring to metrics such as *recall* or *precision* by name risks alienating stakeholders who are unfamiliar with these technical terms.

Instead, BDD clauses should be phrased in a way that highlights the practical meaning of the metric. The goal is to connect the metric's underlying concern with how stakeholders perceive value in the model's behaviour. By describing the outcome directly, we make the clause more intuitive and transparent.

For example, when dealing with a classification problem and evaluating **recall**, the clause should not state, "The model reached 70 percent recall." Instead, it should be expressed as:

```
Then the model correctly classifies real positives of
all classes 70 percent of the time.
```

Similarly, for **precision**, the BDD clause becomes:

```
Then the model correctly classifies predicted positives
of all classes 70 percent of the time.
```

Finally, when focusing on **accuracy**, the clause could read:

```
Then the model correctly classifies all classes 70
percent of the time.
```

This way of phrasing avoids jargon while keeping the intent of the metric intact. It makes the meaning of the clause self-evident, allowing non-technical stakeholders to immediately grasp what the system is achieving. More importantly, it highlights how each metric contributes to business objectives in a way that feels concrete and relevant, rather than abstract or statistical.

The "Then" clauses are organized according to Figure 4.3. Each subsection elaborates on a specific part of the figure in detail.

Table 4.1: BDD Clauses for Model Evaluation Metrics

Metric Type	**BDD Clause**
General Recall	`Then the model correctly classifies real positives of all classes [with a {average_type}] {number} percent of the time`
Class-Specific Recall	`Then the model correctly classifies real positives of class {class} {number} percent of the time`
General Precision	`Then the model correctly classifies predicted positives of all classes [with a {average_type}] {number} percent of the time`
Class-Specific Precision	`Then the model correctly classifies predicted positives of class {class} {number} percent of the time`
General Accuracy	`Then the model correctly classifies all classes {number} percent of the time`

4.2.3.1 Descriptive Performance Metrics

These clauses enable a user-friendly evaluation of model performance without requiring the explicit mention of the metric name. They are illustrated in the upper part of Figure 4.3, and their syntax is highlighted in Table 4.1. The table is organized so that the first column specifies the metric being evaluated, while the second column presents the corresponding BDD clause.

Each clause in the table may include curly brackets or square brackets. Square brackets indicate that the enclosed content is optional, while curly brackets indicate a placeholder that must be filled with a value. In the case of curly brackets, they are used solely to mark where a value should appear and must not be included when writing the clause. The same applies to Tables 4.2 and 4.3. Each placeholder has a specific meaning, as detailed below.

- `{number}` – A numeric value between 0 and 100 that indicates the percentage threshold the metric (e.g., recall, precision, or accuracy) should achieve.
- `{average_type}` – Specifies the type of averaging method to be applied when calculating the metric across classes. This option is often involved around brackets, meaning it's specification is optional. The available options are:
  - `macro average`: Does not take into account the proportion of each class in the test results. Each class contributes equally, regardless

of its frequency. It is the default value when {average_type} is not specified.
  - `weighted average`: Takes into account the proportion of each class in the test results. Classes with more samples weigh more in the overall metric.

- `{class}` – The specific classification output for which the metric should be measured. For example, if your dataset's possible classes are *setosa* and *virginica*, then one of these should be used as the value for `{class}`. If the dataset outputs only binary values (`0` and `1`), these should be written in the clause as `False` or `True`, respectively.

4.2.3.2 Explicit Metrics for Classification Models

This subsection addresses the explicit evaluation of traditional classification metrics, including recall, precision, accuracy, and F1-score. The corresponding syntax is presented in Table 4.2.

Table 4.2: BDD Clauses for Explicit Metrics

Metric Type	**BDD Clause**
General Explicit Metric	`Then the model will reach a value of {threshold} on the metric {metric} [on a {average_type}]`
Class-Specific Explicit Metric	`Then the model will reach a value of {threshold} on the metric {metric} on class {class}`

4.2.3.3 Explicit Metrics for Regression Models

Explicit evaluation of regression metrics. Their syntax is summarized in Table 4.3.

Table 4.3: Explicit metric clauses for regression models

Metric Type	**BDD Clause**
Regression Metric	`Then the model will reach {metric} {reference} {number}`

Where `metric` can be any of the values indicated in Table 4.4 and `reference` can be either `above` (for R^2 Score) or `below` (for error metrics).

Metric	Value
R^2 Score	`an R2 Score`
Mean Squared Error (MSE)	`a Mean Squared Error`
Mean Absolute Error (MAE)	`a Mean Absolute Error`
Root Mean Squared Error (RMSE)	`a Root Mean Squared Error`
Median Absolute Error (MedAE)	`a Median Absolute Error`

Table 4.4: Metrics and their placeholder values.

4.3 Non-Functional Requirements

4.3.1 Performance Efficiency and Scalability

The framework must ensure the rapid execution of tests to maintain an efficient development cycle without causing delays. It should be designed in a way that the performance efficiency and scalability depend mostly on the model prediction time and the test data size.

4.3.2 Usability

The framework should be usable, with comprehensive documentation to help users effectively create and manage BDD scenarios. It should also integrate well with predominant machine learning frameworks and tools, facilitating its adoption into users' workflows.

4.3.3 Maintainability

The framework should be designed with ease of maintenance in mind. This requires a modular architecture, enabling individual components to be updated or replaced without affecting the entire system. It should also be straightforward to update the framework to support new versions of machine learning frameworks and libraries.

4.3.4 Portability

The framework should be operable across various operating systems and environments commonly used for machine learning development, such as different cloud platforms or local setups.

4.4 Architectural and Design Decisions

To implement the framework, based on the requirements, we grounded ourselves on pre-existing libraries to support the main functionalities required for specifying and evaluating machine learning models. For handling the creation, loading, and evaluation of models, we decided that a suitable choice is the Scikit-Learn Python library[2]. This library provides access to multiple evaluation metrics and supports a wide range of model implementations. In addition, to incorporate BDD practices into the framework, we adopted the Behave library[3]. Behave enables the specification of test scenarios in natural language using the Gherkin syntax, which are then directly connected to Python step definitions. This combination allows us to integrate the expressiveness of BDD with the computational capabilities of Scikit-Learn, ensuring that model evaluations are both technically rigorous and accessible to stakeholders. Finally, to guarantee that the framework remains modular, maintainable, and extensible, we also structured its design according to the SOLID principles.

4.4.1 Behave

Behave is an open-source framework for Behavior-Driven Development (BDD) in Python. It enables developers, testers, and business stakeholders to collaborate by writing system specifications in natural language while binding those specifications to executable Python code. Behave uses the Gherkin syntax, which structures scenarios around the keywords `Feature`, `Scenario`, and the steps `Given`, `When`, and `Then`. This structure ensures that requirements are described in a way that is understandable by non-technical stakeholders and directly verifiable through automated execution.

The framework organizes tests in a dedicated `features` directory, which contains feature files written in Gherkin, step definitions implemented in Python, and optional configuration files. Each step written in plain text within a feature file is linked to a corresponding Python function called a step definition. This design attempts to bridge the gap between high-level requirements and low-level implementation, ensuring that system behaviour is consistently validated against agreed-upon acceptance criteria.

To extend flexibility, Behave allows project-specific configurations through files such as `behave.ini`, which define tags, reporting formats, and execution options. It also supports lifecycle hooks via an optional

[2]https://github.com/scikit-learn/scikit-learn
[3]https://github.com/behave/behave

`environment.py` file, enabling users to execute setup and teardown routines before and after test runs or scenarios. These hooks allow the reuse of fixtures, such as database connections or API clients, making Behave suitable for complex testing environments.

A significant advantage of Behave lies in its ecosystem and integrations. It is commonly used in combination with libraries such as Selenium or Appium for acceptance testing of web and mobile applications, and with frameworks such as `behave-django` for integration with Django-based projects. This makes Behave versatile across different types of software testing, from user interface validation to API testing.

With its emphasis on shared understanding, natural language specifications, and executable documentation, Behave provides a practical framework for aligning technical implementation with business goals. In the context of this research, Behave illustrates how BDD practices can be applied in Python projects and informs how BDD4ML extends similar principles to testing machine learning models.

4.4.2 Scikit-Learn

Scikit-Learn is a powerful and widely used open-source machine learning library for the Python programming language. It provides a simple and efficient set of tools for data mining, data analysis, and machine learning. Built on NumPy, SciPy, and matplotlib, Scikit-Learn offers a range of supervised and unsupervised learning algorithms through a consistent interface, making it accessible for beginners and experienced practitioners.

This subsection focuses on describing the Scikit-Learn library to understand how the framework interacts with the models and the training data required to use the framework.

To understand how models work in Scikit-Learn, we need to understand estimators. An "estimator" refers to any object that learns from data. This can be a classification algorithm, a regression algorithm, a clustering algorithm, or even a transformer for preprocessing data. All estimators in Scikit-Learn implement a 'fit' method for learning from data and a 'predict' method for making predictions or tests.

When bringing new data for testing a model, this data should be in the same format as the training data when the model was fitted. This means that any transformations to the training data should also be applied to the testing data. This consistency can be enforced using the 'Pipeline' estimator, which combines all transformations and estimators in a single estimator object. This

means raw data can be used as input to train and test the estimator object when it is a 'Pipeline' instance.

After training the model and ensuring the input data is formatted consistently, we must save our model to load it in the framework. Scikit-Learn does not offer a direct way to save models into files. However, we can save models by using Serialization methods. Serialization converts a Python object into a format that can be easily stored in a file and reconstructed later. In Scikit-Learn, the 'joblib' library is commonly used to serialize estimator objects. When loading back an estimator object, you must also define any custom transformers used in the estimator creation in the framework code; otherwise, custom transformation operations cannot be done.

Another issue when using the Scikit-Learn library is defining the feature names and types used to predict a model's outcome and which feature names and types are predicted by the model. Some estimators, such as the 'KNeighborsClassifier' class, do not have any methods to recover their features. The only way to address this issue is by making the user state the features the model uses in a separate configuration file.

With these considerations of the Scikit-Learn library, we have a clear understanding on how to cover our desired features for the framework regarding data and model loading.

4.4.3 Applying SOLID Principles to the Framework Design

Evidence from empirical studies in ML projects suggests that applying software design principles can significantly improve code understanding among data scientists with diverse backgrounds. In particular, CABRAL et al. (2024) conducted a controlled experiment with three trials and 100 data scientists and found statistically significant gains in code understanding when ML code was refactored to adopt SOLID principles. Their results reinforce that spreading software engineering design principles within the data science community can enhance the maintainability of ML code.

The design of the BDD4ML framework deliberately follows the SOLID principles to improve comprehensibility, extensibility, and maintainability of the code base. Their usage is detailed as follows:

Single Responsibility (S). Each component in the framework has a single, well-defined responsibility: *DataLoader* only loads test data, *Preprocessor* transforms inputs, *ModelLoader* instantiates and/or loads models, *Evaluator* computes metrics, and *Postprocessor* formats results for stakeholders. BDD

clause handlers solely translate stakeholder concerns into executable checks. This separation reduces coupling and makes individual pieces easier to test and reason about.

Open/Closed (O). The framework is open for extension but closed for modification. New models, metrics, or BDD clauses can be added by implementing the corresponding interfaces and registering them, without changing existing, stable code. For example, introducing a new metric-based clause (e.g., a cost-sensitive accuracy) does not require edits to the *Evaluator* or *Controller*; it only adds a new clause strategy using the same interfaces.

Liskov Substitution (L). All interchangeable parts (*DataLoader*, *Preprocessor*, *ModelLoader*, *Evaluator*, *Postprocessor*) conform to abstractions that allow them to be substituted transparently. Any component respecting the interface can replace another without breaking BDD scenarios. This property enables experiments (e.g., swapping a CSV loader for a Parquet loader, or a scikit-learn model for a wrapped PyTorch model) without altering test logic.

Interface Segregation (I). Interfaces are kept small and focused on the needs of their clients. For instance, BDD clause handlers depend on a minimal *Evaluator* interface that exposes only the metric computations they actually use, avoiding large "god interfaces." This keeps clauses lightweight and easier to understand for non-expert stakeholders who read the clauses and their supporting code.

Dependency Inversion (D). High-level modules (BDD clause handlers and the *Controller*) depend on abstractions, not concretions. Concrete implementations (specific loaders, preprocessors, models, and evaluators) are injected at composition time. This design inverts dependencies and isolates high-level policy (stakeholder intent expressed via BDD) from low-level details (data access, model instantiation, metric formulas).

Seamless Integration via a Controller. A thin *Controller* composes the framework at runtime by receiving the concrete instances that implement the required interfaces. This keeps BDD clauses *swappable*: a clause can be rewritten or replaced without touching model, data, or evaluation pipelines. Likewise, a new evaluator or model can be introduced without rewriting clauses. The following composition line illustrates the dependency-injected assembly:

```
context.controller = Controller(
    context.test_dataLoader,
    context.preprocessor,
    context.postprocessor,
    context.model_loader,
    context.evaluator
)
```

Practical Impact on BDD Clauses. Because each clause handler depends only on small interfaces, the same BDD scenario can run with different implementations by changing bindings at composition time. For instance, a clause like: *"Then the model correctly classifies 90% of real positives"* relies on an `Evaluator` interface (e.g., `RegressionEvaluator(...)`) and a `Postprocessor` to render stakeholder-facing outputs. To try a different metric definition or a new reporting style, one can swap the evaluator or postprocessor concretes—no clause text or controller logic must be modified.

Grounded on the positive empirical findings of CABRAL et al. (2024), the framework's SOLID-oriented design ensures that (i) responsibilities are isolated, (ii) extensions do not require risky edits, (iii) components remain interchangeable, (iv) interfaces stay lean, and (v) high-level BDD logic depends only on abstractions. This enables rapid, low-friction evolution of BDD clauses and model/evaluation components while preserving clarity for both engineers and stakeholders.

4.5 Concluding Remarks

In this chapter, we outlined the objectives that guided the construction of our BDD framework. The framework addresses behaviour-centered concerns in ML model testing by expressing system behaviour through BDD clauses. Written in human-readable language, these clauses help clarify for stakeholders what the system is doing and how their concerns are being addressed.

We began by selecting the most relevant concerns to develop, following the taxonomy proposed by VILLAMIZAR et al. (2024), and mapping each of them into categories suitable for demonstrating system behaviour. From this set, we restricted our assessment to the category of *model performance*, which provided the foundation for specifying the framework's functional requirements through a detailed set of BDD clauses.

The framework's functional requirements focus on enabling users to specify which models and data to load, to choose the metrics they wish to evaluate,

and to define the values they expect the model to achieve. Importantly, these metrics are, whenever possible, expressed in a self-descriptive way: rather than presenting abstract statistical terms, the framework communicates what each metric represents in practical, user-oriented language. This self-descriptive emphasis makes the evaluation more transparent and directly connects technical measures with end-user expectations. Alongside these functional requirements, we also established the non-functional requirements and defined the technologies needed to implement them.

The development of this BDD framework represents an important step toward making machine learning model testing more accessible, interpretable, and aligned with stakeholder concerns. By bridging the gap between technical implementation and human-readable specification, the framework supports both transparency and trust in model evaluation.

The next chapters extend this work by moving beyond framework design and into evaluation in practice. We first conduct observational studies with end users to assess how they interact with the framework, and then validate its applicability and usefulness in industry-oriented focus groups. These steps allow us to better understand not only the framework's functionality, but also its value in real-world settings.

5
Observational Study

The methodology outlined in this chapter closely aligns with Step 4, "Validation in Academia" from the technology transfer model presented by GORSCHEK et al. (2006). In this step, the candidate solution undergoes initial validation in a controlled academic environment, ensuring that the solution is theoretically sound and practically viable before advancing to further validation stages. Similarly, to evaluate the effectiveness and usability of our BDD framework, we conducted an observational study involving students from an academic setting. The primary goal of this evaluation was to collect qualitative and quantitative data to help us understand how users interact with the technology and its acceptance.

5.1
Goal, Research Questions, and Variables

The research objective of this study (related to RQ2 from Section 1.3) is to evaluate BDD4ML in terms of its effectiveness for maintainers and their acceptance of it. Using the GQM template (BASILI; ROMBACH, 1988), the goal is defined as follows: "Analyze <the BDD4ML framework> for the purpose of <characterization>, with respect to <task completion rates, challenges encountered, perceived usefulness (PU), ease of use (PEU), and behavioural intention to use (BI)>, from the point of view of <software engineering graduate students> in the context of <specifying test scenarios for an industrial ML model>".

We designed the following research questions to address this goal:

The first research question we aimed to explore (RQ1) was related to the obstacles participants faced when interacting with the BDD4ML framework. Specifically, for each task assigned in the study, we sought to evaluate how well participants performed and identify any difficulties they encountered. To better understand how accessible and effective BDD4ML is for participants, we recorded the percentage of successfully completed tasks and gathered self-reported feedback on their experience. Based on this approach, we formulated the following sub-questions to gain insights into participants' performance:

- **RQ1:** What kinds of challenges do participants experience while utilizing BDD4ML to specify ML model test requirements?

  - (a) What challenges do participants face when specifying actionable ML model test requirements using BDD4ML?
  - (b) What challenges do participants face when analyzing test results and adjusting ML model test requirements accordingly?

Some prior knowledge and experience might influence participants' ability to apply BDD4ML. Hence, we were interested in understanding how these factors affect the effectiveness of participants in using BDD4ML. Therefore, in our study, we also investigated whether knowledge and experience in machine learning, behaviour-driven development (BDD), test-driven development (TDD), and software testing have an influence on the participants' performance. Additionally, we considered participants' prior experience in developing and maintaining ML systems.

The second research question (RQ2) focused on the acceptance of BDD4ML by participants. To evaluate this, we applied the Technology Acceptance Model (TAM) questionnaire (see Table 5.3) to assess the perceptions of participants about the BDD4ML framework. Using the TAM constructs, we aimed to answer the following questions:

- **RQ2:** Would participants accept BDD4ML as a tool for specifying and testing ML requirements?
  - (a) How do participants perceive BDD4ML in terms of ease of use?
  - (b) How do participants perceive BDD4ML in terms of usefulness?
  - (c) Do participants intend to continue using BDD4ML after the experiment?

Since the TAM questionnaire uses positively framed questions about the technology (see Table 5.3), we analyzed the frequency of participants agreeing with these questions. Additionally, we sought to better understand the difficulties participants encountered while using BDD4ML. The frequency of specific difficulties might indicate potential areas for improvement in the framework.

Tables 5.1 and 5.2 describe the set of independent and dependent variables used in the study, along with their types and scales.

Table 5.1: Independent Variables

Type	Variable name and definition	Scale
Level of knowledge and experience	(L-SE) Experience with software engineering	0 = None, 1 = Beginner, 2 = Novice, 3 = Intermediate, 4 = Advanced, 5 = Expert
	(L-BDD) Knowledge of BDD concepts	
	(L-ML) Knowledge of general ML concepts	
	(L-MLS) Knowledge of ML specification concepts (e.g., model requirements, documentation, testing)	
	(L-MLD) Experience developing ML systems	
Academic background	(A-LEVEL) Level of academic background of the participant	1 = Pursuing a Bachelor's, 2 = Completed a Bachelor's, 3 = Pursuing a Master's, 4 = Completed a Master's, 5 = Pursuing a Ph.D., 6 = Completed a Ph.D.

Table 5.2: Dependent Variables

Type	Variable name and definition	Value
Percentage of Participants	(P-A-Q1) Who slightly agree or agree with TAM Q1 (performance)	Percentage (0 to 100%)
	(P-A-Q2) Who slightly agree or agree with TAM Q2 (effectiveness)	
	(P-A-Q3) Who slightly agree or agree with TAM Q3 (productivity)	
	(P-A-Q4) Who slightly agree or agree with TAM Q4 (usefulness overall)	
	(P-A-Q5) Who slightly agree or agree with TAM Q5 (ease of learning)	
	(P-A-Q6) Who slightly agree or agree with TAM Q6 (ease of use for requrements)	
	(P-A-Q7) Who slightly agree or agree with TAM Q7 (ease of becoming skilled)	
	(P-A-Q8) Who slightly agree or agree with TAM Q8 (overall ease of use)	
	(P-A-Q9) Who slightly agree or agree with TAM Q9 (intention to use)	
	(P-C-T1) Who successfully completed Task 1	
	(P-C-T2) Who successfully completed Task 2	
Time used	(AVG-TIME) Average time taken by participants to complete all tasks	Time in minutes

5.2 Participants

We selected the study subjects out of convenience. All thirteen participants were graduate students in informatics from the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). Before conducting the experiment, two other students from separate courses were selected to make pilot sessions, helping to refine the study process before applying it to a bigger group.

When it comes to the students' strengths and weaknesses, even though all students were part of the same course, all of them had varying backgrounds in the fields of BDD, requirements engineering, and machine learning.

5.3 Study Structure and Instrumentation

5.3.1 Characterization Questionnaire

At the start of the study participants were asked to fill a characterization questionnaire. This initial form aims to evaluate participants' backgrounds in software engineering and machine learning, focusing on their academic qualifications, general experience, and familiarity with specific concepts such as BDD and ML specifications. The questionnaire incorporates a proficiency scale inspired by the NIH Proficiency Scale[1], which provides a structured framework to self-assess expertise levels. This scale ensures consistency in responses by defining clear criteria for levels ranging from "Beginner" to "Expert". The data collected enabled researchers to contextualize and interpret the interactions and performance of the participants within the observational study, as well as to define which characteristics are the best predictors of success when performing tasks with the framework.

5.3.2 Framework Familiarity

After the initial questionnaire, participants were subjected to a presentation explaining the framework's purpose and features. After being directed towards the documentation, participants were left to their own devices to familiarize themselves with a few examples of working framework scripts that they could also use as a reference for their own activities.

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

Finally, they were briefed on the project to be worked on: designing test requirements for a ML predictor on carbon emissions for shipment planning of a large oil and gas company.

5.3.3 Framework Usage

In the first activity, users were given a regression task to design metrics to use the tool, simulating a real-world scenario that the framework is designed to address. Their task was to design test scripts using the BDD clauses to evaluate the machine learning model. Participants were also assisted by the framework's developer to solve common issues and clarify ambiguities in the documentation.

After 20 minutes, a second activity was conducted using the same prediction model but applying a post-processing module to treat the data after inference. This treatment converted the output of the model from continuous values to discrete classes, changing the problem from a regression task to a classification one. This change not only showed the framework's versatility by being able to integrate different transformations of data but also allowed testing of the framework's classification features without changing the problem's context.

5.3.4 Follow-up Questionaire

After performing each task of the study, maintainers were asked to fill out a follow-up questionnaire. This questionnaire was designed based on the Technology Acceptance Model (TAM) (DAVIS, 1989). The answers are provided in a five-point Likert scale where users rate the questions by how much they agree with the provided statement. It was structured to gather comprehensive feedback on the use of the BDD4ML framework, focusing on the main TAM constructs. The first construct, Perceived Usefulness (PU), evaluates how beneficial participants find the framework for achieving their objectives, particularly in specifying and testing ML requirements. Questions related to this construct examine whether the framework improves performance, productivity, and effectiveness. For example, participants are asked if using BDD4ML enhances their ability to complete tasks successfully or helps them accomplish more with less effort.

The next construct, Perceived Ease of Use (PEU), focuses on the simplicity and intuitiveness of the BDD4ML framework. These questions assess whether participants find the framework user-friendly, straightforward to ap-

ply, and easy to learn. Participants reflect on their ability to operate and become proficient with the framework, with questions exploring the ease of using it for ML requirements and testing tasks.

Finally, Behavioural Intention to Use (BI), measures participants' willingness to incorporate the BDD4ML framework into their regular workflows. This section explores their likelihood of adopting the framework for ML specification and testing in the future. By understanding participants' intentions, the study aims to gauge the potential long-term acceptance and integration of the tool. The constructs and questions are shown in Table 5.3.

Table 5.3: TAM questions used in the BDD4ML study

Construct	**ID**	**Question**
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.

The questionnaire also features an "Additional Comments" section, allowing participants to share qualitative feedback. This open-ended space encourages reflections on their experiences, highlighting what they enjoyed most and least when using the framework, as well as suggestions for improvement. Such insights were essential for refining the framework based on user feedback.

The participants also provided objective data related to the activity through the BDD4ML specification file designed to perform the tests, allowing the evaluators to ensure that all tasks were completed correctly. This data provides context for interpreting the responses and evaluating the efficiency of the framework.

Together, these data points ensure a holistic evaluation of the BDD4ML framework, combining quantitative measures with qualitative insights to inform its development and usability.

5.4 Study Results

After conducting the study, the collected data was analyzed to evaluate the impact of the BDD4ML framework on specifying ML requirements. The analysis involved both quantitative and qualitative data to assess the task completion rates, challenges encountered, perceived usefulness, ease of use, and behavioural intention to use. This section presents the results organized by tasks, completion time, and user feedback on challenges, usefulness, ease of use, and intention to adopt BDD4ML.

5.4.1 Proficiency Distribution of Participants

To better understand the participants' background, we analyzed their proficiency levels on relevant topics, including knowledge of machine learning concepts, BDD principles, and software engineering experience. Figure 5.1 illustrates the proficiency levels of participants across these topics. It highlights variations in their familiarity with the required concepts, which might influence task completion rates and user perceptions.

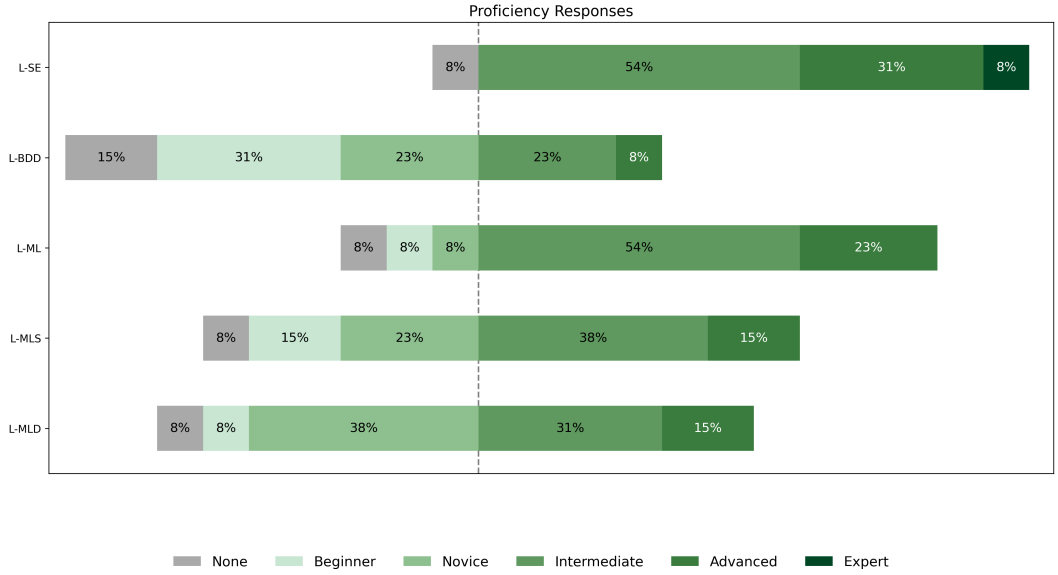

Figure 5.1: Characterization of the participants in terms of proficiency distribution.

Table 5.4: Completion rate

Task	Completion Rate (%)
Task 1	92.31
Task 2	61.54

5.4.2 Task Completion Analysis

The study involved two main tasks: Task 1 asks the participant to make a BDD feature file to test a regression model on the problem of estimating carbon emissions. Task 2 builds on top of Task 1 by using the same model but adding a post-processor to convert continuous emission values into discrete class values, turning the problem from a regression task to a classification one.

Table 5.4 presents the percentage of participants who completed each task. Of all participants, 92.31% successfully completed Task 1, while 61.54% completed Task 2. These results indicate a noticeable drop in task completion rate for Task 2, potentially reflecting its higher complexity.

For quantitative analysis, we used the Wilcoxon-Mann–Whitney test (TEUGELS, 2014), which is a non-parametric statistical test used to determine whether there is a significant difference between two independent groups. Unlike parametric tests, it does not assume that the data follows a normal distribution, making it particularly suitable for small sample sizes or data that may not be normally distributed. Given that our experiment involves a relatively small number of participants, with uneven group sizes (Group that completed both tasks: n=8, Group that completed one or zero tasks: n=5), this test was chosen because it allows us to robustly compare the levels of knowledge and experience across groups without relying on distributional assumptions.

The results of the test are presented in Table 5.5. We found statistically significant differences ($p < 0.05$) for the following variables: L-ML ("Knowledge of General Machine Learning (ML) Concepts"), L-MLS ("Knowledge of ML Specification Concepts") and L-MLD ("Experience Developing ML Systems"). For the remaining variables L-SE ("General Experience with Software Engineering", L-BDD ("Knowledge of Behaviour-Driven Development (BDD) Concepts") and A-LEVEL ("Level of academic background"), no statistically significant differences were found.

The analysis reveals that participants with greater expertise in general machine learning concepts, ML specification concepts, and ML system development were more likely to successfully complete both tasks. These findings suggest that proficiency in ML-related areas plays a key role in task performance, while general software engineering experience and BDD knowledge were

Table 5.5: Wilcoxon-Mann–Whitney test

	U Statistic	p-value
L-SE	9.5	0.962432
L-BDD	28.5	0.114508
L-ML	31.0	0.046294
L-MLS	32.5	0.034157
L-MLD	32.5	0.033137
A-LEVEL	20.0	0.532790

not significant differentiators in this context. However, it is noteworthy to emphasize our limited sample size (n=13) and that by analyzing the types of errors, they do not relate to ML concepts.

Participants made different types of errors during the tasks. Table 5.6 summarizes the most common errors and the number of participants who encountered each error. Errors were separated into two types: "Incorrect clause syntax" and "Incorrect definition". The first relates to mistakes where the participant did not write the code correctly even though they had the right idea. The second error is when a participant does not define the problem correctly such as using a wrong file or forgetting to set up a component. The first type of error is more forgiving since it just means that they forgot a small syntax detail, while the second is less forgiving due to showing that the participant did not understand the assignment correctly.

It is worth noting that all but one of the 'Definition' type errors were committed by a single participant, who can be considered an outlier. 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.

5.4.3 TAM questionaire

The Technology Acceptance Model (TAM) questionnaire results were derived from participant responses to questions regarding the perceived usefulness, ease of use, and intention to use the BDD4ML framework. Figure 5.2 presents the percentages of agreement for each question. Further discussion about the test results takes place in the forthcoming subsection.

Table 5.6: List of Identified Errors

Error Type	Occurrences	Description
Definition Error	1	Incorrect model name used that does not exist in the test configuration.
	2	Wrong class name specified in a classification clause.
	1	Classification metric value not properly updated or configured.
	1	Missing required post-processing configuration before the clause.
Syntax Error	1	Redundant use of the keyword "of" in the clause.
	1	Single quotation marks used instead of double quotation marks.
	2	Extra space introduced within the clause, causing a syntax error.

5.5 Discussion

This subsection answers the research questions by critically analyzing the results from the TAM questionnaire and the comments provided by users. The Likert scale graph presented in Figure 5.2 demonstrates overwhelmingly positive responses regarding the acceptance of the proposed framework. A majority of participants either agreed or strongly agreed with the statements, indicating a favorable perception of the framework's usefulness and potential adoption.

Only a small proportion of respondents slightly disagreed with certain aspects, suggesting that negative opinions were minimal and mostly tied to individual preferences rather than fundamental flaws in the approach. Specifically, regarding Perceived Usefulness (PU), one detractor mentioned, "I prefer more structured code, especially for versioning comparison," highlighting a personal inclination toward working with direct source code rather than adopting a low-code framework due to versioning concerns. Similarly, in terms of Behavioural Intention of Use (BI), another respondent stated, "It is not exactly my field, so I am not sure if I will use it anywhere soon," which indicates that their reluctance stems from a lack of immediate application in their domain rather than any perceived shortcomings in the framework itself. Only one detractor mentioned some error related to the usage: "It was difficult to understand the errors for invalid structure, missing parameters.", revealing a possible challenge for the framework's adoption. Overall, these

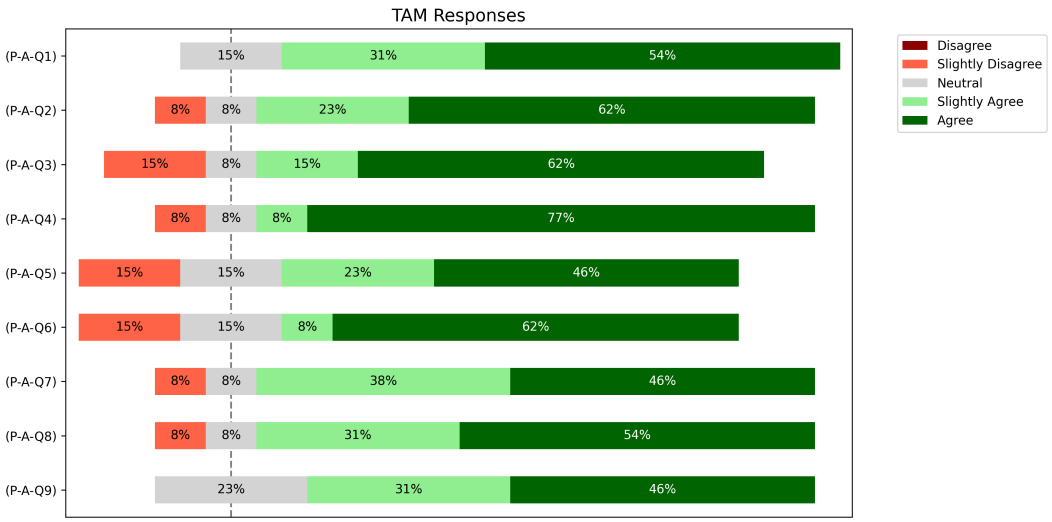

Figure 5.2: TAM Responses

responses suggest that while a few individuals had reservations, they were mostly context-dependent, reinforcing the overall positive reception observed in the study.

- **RQ1:** What kinds of challenges do participants experience while utilizing BDD4ML to specify ML model test requirements?
- (a) What challenges do participants face when specifying actionable ML model test requirements using BDD4ML?

Several participants highlighted an initial learning curve when using BDD4ML. Some noted that getting accustomed to the syntax and structure of clauses was challenging at first. One participant commented that they felt a bit lost when trying to find the correct commands to use the framework, and another mentioned that it took time to adapt to the format and parameters required by BDD4ML. Despite this initial difficulty, many participants agreed that once they overcame the learning barrier, the framework became more intuitive and valuable for specifying ML requirements and tests.

Furthermore, some participants expressed that error messages were not always clear, making it harder to understand what went wrong when issues arose. For instance, a participant stated, “It was difficult to understand the errors for invalid structure and missing parameters,” indicating that better error handling or documentation could improve usability. Another participant suggested that a video tutorial or more interactive examples would help accelerate the initial learning process.

Another one of the recurring issues when using the tool was related to the rigid structure of clause writing. One participant mentioned that

the framework's structured format was somewhat limiting, requiring frequent alternations and parameter adjustments until they became accustomed to it. Another noted that specifying metrics for a specific class versus the entire model was not always straightforward, particularly in the "THEN" clauses, where subtle differences in syntax could lead to errors.

Additionally, the need to manually write feature files posed another challenge. As a participant pointed out, "It would be interesting to have a mechanism to automate the completion of the feature file," suggesting that an automated or low-code tool for writing feature files could enhance the user experience. This reflects a common desire among users for more streamlined processes when specifying requirements.

Finally, defining tests for ML models also presented difficulties due to errors related to incorrect syntax or file naming conventions. One participant described confusion when naming model files and test files, stating that they found it odd that they had to use part of the model file name and the complete test file name. This inconsistency in naming conventions added to the complexity of defining tests. Another participant pointed out that they faced issues when specifying preprocessing and post-processing steps at the beginning of feature files, suggesting that clearer instructions or automated support could reduce confusion.

Despite these obstacles, several participants noted the benefits of using BDD4ML for defining tests. They appreciated the clarity and structure that the framework brought to the process, with one participant stating, "I found BDD4ML useful because it organizes ML tests in a clear and structured manner, facilitating understanding among different teams."

– (b) What challenges do participants face when analyzing test results and adjusting ML model test requirements accordingly?

Participants also encountered difficulties in analyzing test results and adjusting ML requirements based on those results. A notable comment highlighted issues with interpreting error messages and understanding what actions to take in response to specific failures. The participant stated, "The error messages were not very clear about what I was doing wrong," emphasizing the need for more explicit guidance or improved error reporting mechanisms.

However, there were positive experiences as well. One participant highlighted the utility of the output provided by BDD4ML during test execution, stating that it explicitly indicated which clause failed, thus improving their perception of what needed to be done to overcome the failure. This indicates that while error clarity is an area for improvement, the existing feedback mechanism still provided value.

- **RQ2:** Would participants accept BDD4ML as a tool for specifying and testing ML requirements?
  - (a) How do participants perceive BDD4ML in terms of ease of use?
  - (b) How do participants perceive BDD4ML in terms of usefulness?
  - (c) Do participants intend to continue using BDD4ML after the experiment?

The TAM questionnaire was used to assess the BDD4ML acceptance by maintainers. Overall, the TAM results in Figure 5.2 indicate a high level of perceived usefulness and ease of use, with positive agreement percentages exceeding 69% for all questions. Agreement rates for each TAM construct 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" (PU) included four questions;

we first computed the agreement rate for each individual question and then averaged these values to obtain the overall agreement rate for the PU construct.

Notably, participants rated the questions related to the PU construct (questions P-A-Q1, P-A-Q2, P-A-Q3, and P-A-Q4) on average with an 84.62% agreement, highlighting its potential to enhance task performance and effectiveness. Regarding the questions related to the PEU construct (P-A-Q5, P-A-Q6, P-A-Q7, and P-A-Q8), they were rated with a slightly lower average agreement of 76.92%, suggesting that while the framework is generally intuitive, there may be room for improvement in initial onboarding and usability. Finally, BI was measured with a 76.92% agreement.

5.6 Threats to Validity

To ensure a rigorous evaluation of the BDD4ML framework on this observational study, we identify and discuss the main threats to the validity of our findings, following the categories suggested by WOHLIN et al. (2024).

5.6.1 Internal Validity

Internal validity concerns whether the observed effects can be attributed to the BDD4ML framework and not to other confounding variables. Participants came from the same course and institution, which could introduce bias due to shared background or instructional context. To mitigate this, pilot sessions were conducted, and a structured methodology was adopted to minimize

variation. However, differences in individual ML proficiency may still have influenced task performance.

5.6.2 Construct Validity

Construct validity refers to whether the constructs under investigation—such as usability, perceived usefulness, and task success—were accurately measured. The use of the Technology Acceptance Model (TAM) and structured task completion rates helps address this concern. Nonetheless, while the TAM is a widely used instrument, its subjective nature makes responses sensitive to individual interpretation. We attempted to control this through clear task briefings and consistent instructions across all participants.

Another potential threat to construct validity arises from the design of the self-descriptive BDD clauses for classification metrics (e.g., accuracy, precision, and recall). These clauses were directly inspired by formal definitions of the metrics, which may not represent the only—or necessarily the best—way to phrase them in a self-descriptive manner. Alternative formulations might have provided clearer communication to end users or stakeholders unfamiliar with technical terminology. A more thorough exploration, such as conducting a workshop or focus group to compare different phrasings, could have offered stronger evidence regarding the most effective way to convey metric meaning through BDD clauses.

5.6.3 External Validity

External validity pertains to the generalizability of the results. The study was conducted with graduate students, who may not fully represent professional software engineers. To address this limitation, we complemented this study with a focus group in an industrial context involving experienced ML practitioners described in the next chapter. Even so, the scope of participants and specific domains (e.g., carbon emissions estimation) may limit broader applicability to other ML application areas.

5.6.4 Conclusion Validity

Conclusion validity involves the extent to which the conclusions drawn are credible based on the data. To assess the statistical significance of task performance differences, we employed the Wilcoxon-Mann-Whitney test, which is particularly suited for small sample sizes and non-parametric data distribu-

tions. This strengthens confidence in our findings despite the limited number of participants (n=13). Nevertheless, while the results suggest that ML expertise influenced task success, replication with larger and more diverse groups would further validate our conclusions.

5.7 Concluding Remarks

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

6
Focus Group

The methodology outlined in this chapter closely aligns with Step 5, "Static Validation" from the technology transfer model presented by GORSCHEK et al. (2006). In this step, the candidate solution is subjected to industry professionals. Therefore, we conducted a focus group involving data science professionals. The primary goal of this evaluation is to collect qualitative feedback that helps us understand how the usage of the BDD4ML framework is perceived in the opinion of industry practitioners in terms of its coverage, completeness, and acceptance.

We presented the BDD4ML framework and its main features to the development team responsible for the same carbon emission estimation model used in the observational study discussed in Chapter 5. The objective of this focus group was to validate whether the BDD4ML testing framework for machine learning models is usable by the team and brings value to the development process.

6.1
Goal and Research Questions

The research objective of this study (related to RQ3 from Section 1.3) is to evaluate BDD4ML in terms of its effectiveness for industry practitioners and their acceptance of it. Using the GQM template (BASILI; ROMBACH, 1988), the goal is defined as follows: "Analyze <the BDD4ML framework> for the purpose of <characterization>, with respect to <the general perception, its coverage and completeness, perceived usefulness (PU), ease of use (PEU), and behavioural intention to use (BI)>, from the point of view of <software engineers in the industry>, in the context of <specifying test scenarios for an industrial ML model>".

The following research questions were derived from this goal:

- **RQ1 - General Perception:**
  - RQ1.1: What are your initial thoughts on the framework?
  - RQ1.2: Do you see an immediate benefit to your current process?
- **RQ2 - Coverage & Completeness:**
  - RQ2.1: Does the framework test the right things?
  - RQ2.2: Are there any gaps in what it evaluates?

- **RQ3 - Acceptance (based on the TAM model):**
  - RQ3.1: I believe this framework would be easy to use when testing ML models.
  - RQ3.2: I believe this framework would be useful for ML-enabled system projects.
  - RQ3.3: I would use this framework in my future ML-enabled system projects.

6.1.1 Selection of Subjects

We deployed a characterization questionnaire to assess the participants' backgrounds and capacities related to software engineering and machine learning, ensuring their profiles matched the experimental context.

6.1.2 Methodology - Focus Group Planning and Design

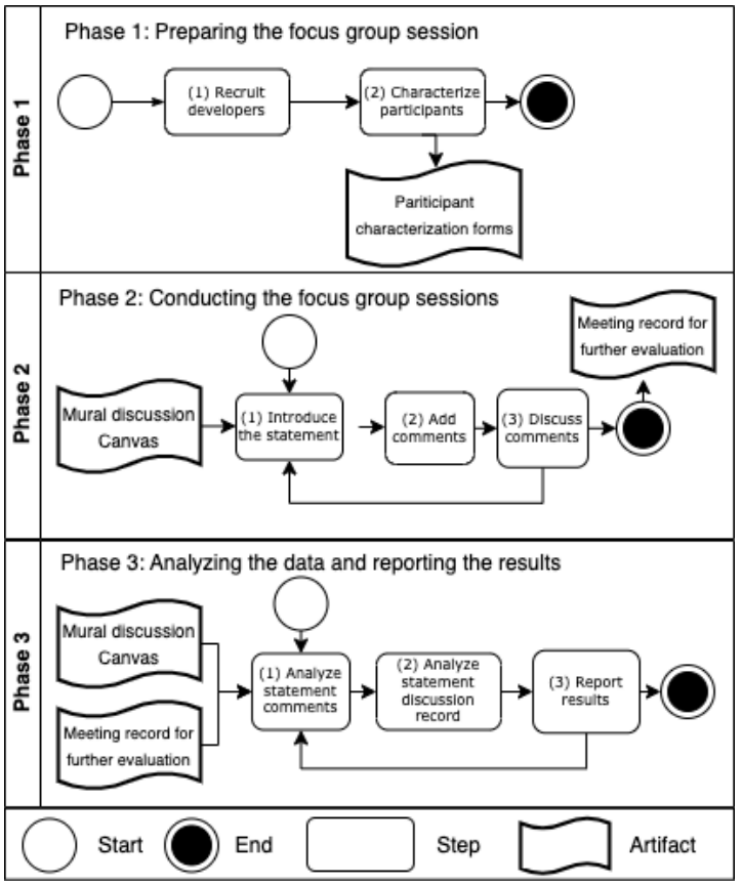

Figure 6.1: Focus Group Overview

The focus group followed the guidelines set by KONTIO; BRAGGE; LEHTOLA (2008). Figure 6.1 illustrates the structure used in the process of

conducting the focus group and analyzing results. The discussion was divided into 3 main phases: (1) Preparing the focus group session; (2) Conducting the focus group sessions; and (3) Analyzing the data and reporting the results. In the following, we describe each phase and step.

6.1.2.1 Focus Group Question Design

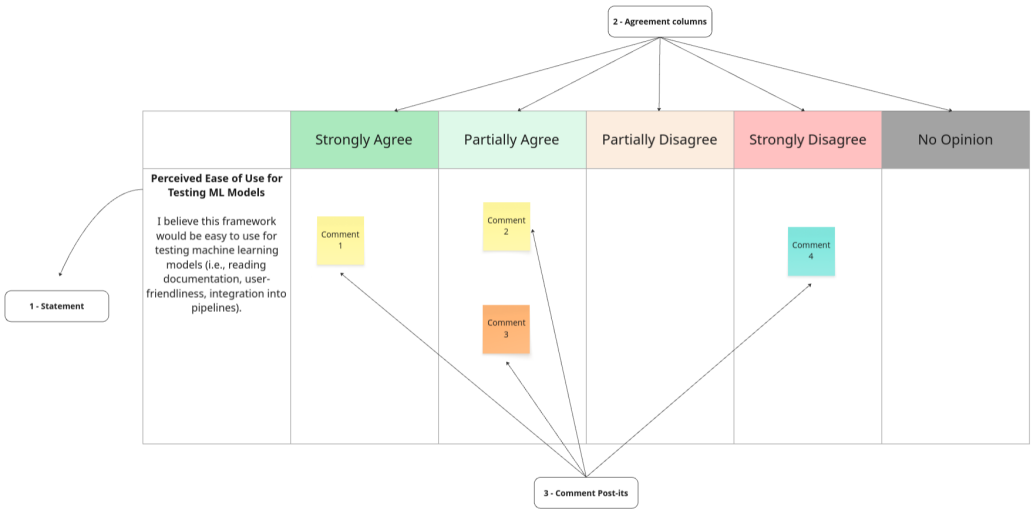

Figure 6.2: Example of MIRO board with participant post-its (color-coded).

The discussions took place in an online environment using the MIRO collaborative platform, where we designed an interactive template to facilitate the focus group, as shown in Figure 6.2. The mural was divided into five columns where participants could express their level of agreement with each statement using virtual post-its: strongly agree, partially agree, partially disagree, strongly disagree, or no opinion. Furthermore, Participants could make multiple comments on different levels of agreement. Each statement was discussed separately following a structured three-step process:

Step 1: Introduce the statement The moderator began by reading the statement aloud to the group.

Step 2: Add comments Participants were then asked to write one or more post-its containing their comments on the statement and place them in the column that best reflected their opinion. Some questions did not have multiple columns for level of agreement and served just to prompt discussion.

Step 3: Discuss comments Next, participants explained the reasoning behind their comments — why they agreed or disagreed — and the group engaged in discussion around those views. Each comment was summarized on its corresponding post-it, and participants were encouraged to elaborate by sharing relevant knowledge and experiences. If a comment was unclear or

insufficiently detailed, the moderator prompted the participant to expand on it to support a richer conversation and deeper understanding.

By using this tone, we sought to foster an environment where participants felt comfortable highlighting both strengths and limitations, offering suggestions for improvement, and discussing potential use cases or obstacles. The phrasing of the questions encouraged constructive feedback rather than prescriptive answers, aligning with our goal of refining the framework based on diverse perspectives.

6.1.2.2 Focus Group Structure

Phase 1 - Preparing the focus group session This phase involves gathering the initial resources needed to carry out the focus group session. To do so, we followed two steps.

Step 1 - Recruit developers: This step involved identifying and recruiting developers with experience in ML to take part in the discussions. We reached out to developers responsible for the model employed on the observational study from Chapter 5. The team is comprised of three professionals, who all agreed to participate after signing a consent form that outlined our research objectives and assured them that their information would remain confidential and be used solely for research purposes.

Step 2 - Characterize participants: The goal of this step was to gather basic information about the participants using a Participant Characterization Form. This allowed us to better interpret the results of our study by profiling each participant. The form asked about their academic background, years of experience working with ML, and a self-assessment of their knowledge in ML and BDD, rated according to the NIH Proficiency Scale[1].

Phase 2: Conducting the focus group sessions This phase involves gathering data on participants' perceptions of perceived benefits and limitations of the BDD4ML framework on their current workflow. To do so, we followed these steps:

Step 1 - Framework presentation: To familiarize participants with the framework, we prepared a set of slides presenting its structure and key concepts, how it fits into their current workflow, and its business value and usability, avoiding technical overload.

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

Step 2 - Research question presentation: We have presented the participants with the research questions so they could provide feedback on the framework. The first research questions serve to capture immediate impressions of the framework and the perceived benefits of the features proposed. The following questions concern the coverage of the Technology Acceptance Model (TAM).

Phase 3 – Analyzing the Data and Reporting the Results The focus group session was conducted online via the Discord video-conference application, with both video and audio recordings captured to support the subsequent data analysis. These sessions took place in April 2025.

To analyze the feedback, we examined the participants' comments for each research statement. This analysis was informed by detailed transcripts generated from the audio recordings, allowing us to interpret the intent and meaning behind each contribution more accurately. The transcription process was performed using a widely adopted speech-to-text model called whisper [2]. The transcript text file is available in our Zenodo repository (MOTTA; KALINOWSKI, 2025).

In the following section, we present the results of these discussions, organized around each research statement. The analysis reflects both the individual responses and the collective themes that emerged from the participants' interactions.

6.2 Results - Analyzing the Feedback

To evaluate the perceptions of participants regarding the proposed framework, we conducted a qualitative analysis of a recorded focus group session. The discussion was transcribed using automatic speech-to-text methods, and subsequently reviewed to identify relevant quotes and refine transcription accuracy. In parallel, we analyzed the digital post-it notes written by participants on the virtual board during the session.

Our analysis method consisted of triangulating the comments from the collaborative board with the verbal feedback extracted from the transcription. This allowed us to contextualize agreement or disagreement across participants and provide illustrative quotes supporting each point. The goal was to surface not only the general sentiment toward the framework but also detailed perceptions of its value, usability, and coverage.

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

In the following subsections, we present the results organized by each research question (RQ), as outlined in our evaluation plan. Each subsection includes a synthesis of board responses and associated transcription segments. While we highlight relevant quotes to substantiate findings, the full qualitative analysis and additional details are available in our online repository. Complete transcripts and video recordings were not made publicly available to preserve participant anonymity.

Furthermore, Figure 6.3 illustrates the distribution of attributes obtained from the characterization questionnaire administered to the focus group participants. At first glance, the figure may seem unusual since it represents only three participants. However, it is important to note that these participants are active professionals engaged in machine learning model development, in contrast to the graduate students who composed the observational study.

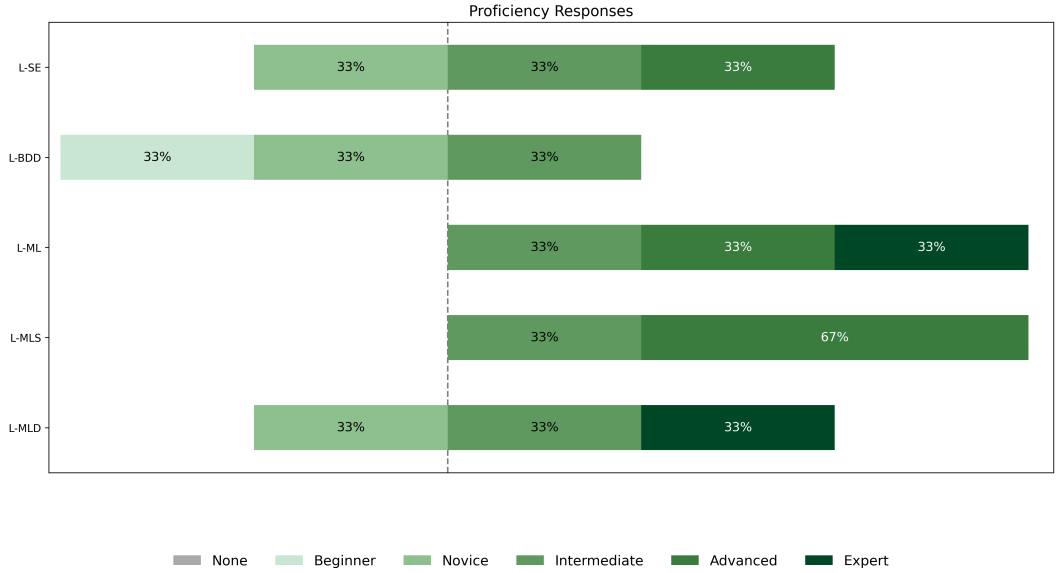

Figure 6.3: Proficiency distribution of participants on topics relevant to the focus group.

RQ1.1 – What are your initial thoughts on the framework?

Participants expressed a generally positive first impression of the framework, highlighting its potential to support quality control, improve transparency, and facilitate collaboration across technical and non-technical stakeholders.

A central theme raised by **Participant 1 (green)** was the *ability to block the deployment of poor-performing models*, acting as a safeguard mechanism:

> "We can stop a bad model from being implemented. Somehow, it doesn't get accepted... that's a good thing."

They also emphasized the usefulness of the framework for *model monitoring and identifying issues such as data drift*, noting that the system made such detection easier.

Another important point brought up was the *possibility of delegation to non-technical stakeholders*, such as Product Owners (POs), especially when supported by low-code tools. As Participant 1 noted:

> "The PO could write the BDD clauses while developers integrate them. That's one less thing for the team to worry about."

Participant 2 (blue) reinforced the idea that the framework *clearly communicates whether performance expectations were met*, particularly in relation to selected metrics. They appreciated how the system distinguishes between different performance levels, e.g., flagging when a minimum requirement is met versus an optimal threshold:

> "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 3 (orange) echoed these observations, valuing the tool's role in *detecting undesirable metric shifts and potential model drifts*, as well as making such information *more accessible through automated 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."

Additionally, participants brainstormed potential future extensions, including:

- A comparison mode to *evaluate two models side by side*, especially useful for production deployments;
- A *true/false flag* output to automate decisions like model acceptance or rollback;
- Integration of *version control and history tracking* of model evaluation outcomes.

Taken together, the responses reflect an initial perception of the framework as **a practical, flexible, and automation-friendly tool** that supports not only technical validation but also *cross-functional usability and auditability.*

RQ1.2 – Do you see an immediate benefit to your current process?

Participants identified several practical and immediate benefits that the framework could bring to their current workflow, particularly regarding model validation, deployment autonomy, and collaboration with clients.

Participant 1 (green) noted that the framework could help *prevent the deployment of suboptimal models* by establishing clear and verifiable acceptance criteria:

> "It helps avoid deploying models that don't meet the minimum expected performance."

They also emphasized the usefulness of defining *explicit targets for the team* and suggested that the *test specifications could be collaboratively built with the client*, especially in scenarios involving low-code tools. These features, they argued, could reduce friction during delivery:

> "The file can be built together with the client, especially when considering low-code platforms."

Participant 2 (blue) pointed out that the framework offers support for *improving model result monitoring by the client*, by clearly outlining which metrics should be tracked and how they should be interpreted:

> "It can tell the client which metrics to observe... It helps monitor the model's results."

Participant 3 (orange) highlighted the potential for the framework to *facilitate model optimization and parameter tuning*, particularly when comparing versions of a model that differ only in hyperparameters. They also emphasized how the system could make it easier to *hand off the model to the client*, giving them greater autonomy in post-deployment validation:

> "It helps when testing models with different hyperparameters... and makes it easier for the client to monitor things on their own."

Beyond these immediate uses, participants also explored *extension possibilities* such as:

- Generating automatic reports beyond console output;
- Supporting warning-level thresholds for metrics to avoid binary pass/fail logic;
- Allowing users to define and inject custom metric functions;

- Including model metadata such as *hyperparameters* in the output report for reproducibility and performance comparison;
- Integration with *cloud platforms* (e.g., AWS, GCP, Azure) for operational scaling.

These discussions reflect a perception of the framework as an **immediately valuable tool** for improving quality assurance and enhancing collaboration in ML pipelines, while also revealing areas of improvement for deeper integration and automation support.

RQ2 – Coverage & Completeness

To assess the coverage and completeness of the framework, participants were invited to discuss whether the framework evaluates the right aspects of an ML system (RQ2.1), and whether there are any missing elements in its current implementation (RQ2.2). These two questions were discussed together during the focus group.

Overall, participants considered the framework's current coverage satisfactory, especially regarding the use of widely adopted metrics such as *precision* and *recall* through the `scikit-learn` library. However, several suggestions emerged for extending the framework's capabilities and increasing its usefulness in production scenarios.

Multiple participants emphasized the need for a richer output mechanism, including generating a formal execution report instead of only printing results to the console. This report could include not only metric values but also signals such as warnings or errors, helping teams quickly identify issues. For example, Participant 3 (orange) proposed highlighting metrics that fall outside expected ranges, even if they are not strictly incorrect, to support trend monitoring and early detection of performance degradation.

Participants also suggested creating a standardized result schema for the BDD script to return a general status, such as `"Success"`, `"Warning"`, or `"Error"`. This would facilitate integration with other parts of a deployment pipeline, especially when decisions depend on test outcomes. A particular point of agreement was that warnings should not block the process but should still be logged and treated explicitly in the output.

Another relevant point was the ability to pass custom metric functions and loaders. Participant 1 (green) noted that the framework should support user-defined metrics by allowing users to pass their own functions, as long as they comply with a defined interface (e.g., returning a boolean or scalar).

This makes the solution more flexible and adaptable to project-specific needs, especially when using models that do not follow the `scikit-learn` interface.

There were also suggestions to improve the framework's compatibility with alternative execution environments. One example was integrating the solution with cloud providers such as AWS, Azure, and GCP, leveraging native tools for script execution and monitoring.

Additionally, Participant 3 (orange) brought up the importance of including model hyperparameters in the execution output. This would enable comparisons between different model configurations, especially in A/B testing contexts or optimization workflows.

In summary, while the framework was seen as sufficiently covering core evaluation functionalities, participants pointed out several actionable enhancements that could improve its completeness, usability, and adaptability to real-world deployment contexts. These included better reporting, more expressive status codes, flexible metric and model handling, and support for cloud integration.

RQ3.1 – I believe this framework would be easy to use when testing ML models

Table 6.1: Statement 1 of focus group opinion

	P1	P2	P3
Strongly Agree			
Partially Agree	X	X	X
Partially Disagree			
Strongly Disagree			
No Opinion			

To assess the perceived **ease of use** of the framework, participants were asked to respond to the statement "I believe this framework would be easy to use when testing ML models", and justify their level of agreement, reported in Table 6.1. All participants selected the position *Partially Agree*, and their comments highlighted a number of factors influencing the usability of the framework in its current form.

Participants recognized that the framework provides valuable structure and guidance for testing ML models, but noted that its ease of use is currently **highly dependent on the technical expertise of the user**.

Participant 1 (green) remarked that using the framework without a low-code interface requires someone with Python programming skills. In such cases, the framework might not present a significant usability advantage

compared to writing custom scripts from scratch. However, they also pointed out that the inclusion of low-code features and integration with other parts of the development environment could substantially increase its usefulness and usability, particularly in production workflows.

Participant 2 (blue) echoed this point, stating that the ease of use "depends on the technical background of the user," but nonetheless acknowledged that the framework "undoubtedly facilitates the testing of models."

Participant 3 (orange) focused on the use of **custom input files and pre-/post-processing scripts**, expressing concerns about potential sources of error and the difficulty in debugging these components. They questioned how easy it would be for a user to identify and understand failures when the issue lies in the pipeline defined by the user rather than the framework itself. The host clarified that such failures would raise exceptions like in regular Python code, meaning the debugging experience would resemble standard development workflows in the `Behave` testing framework.

In summary, although participants agreed that the framework has promising usability potential, especially with future enhancements such as low-code tools and pipeline integrations, its current implementation requires a certain degree of programming knowledge. This makes it more suitable for technically skilled team members unless further abstractions are added to improve accessibility.

RQ3.2 – I believe this framework would be useful for ML-enabled system projects

Table 6.2: Statement 2 of focus group opinion

	P1	P2	P3
Strongly Agree		X	
Partially Agree	X		X
Partially Disagree			
Strongly Disagree			
No Opinion			

This question explored participants' perceptions regarding the **practical utility** of the framework in real-world projects involving ML-enabled systems. Participants were asked to consider aspects such as *industrial relevance*, *test efficiency*, and *support for decision-making*. As summarized in Table 6.2, two participants selected "Partially Agree" and one selected "Strongly Agree."

Despite this variation, the responses revealed a common recognition of the framework's usefulness, especially for human-in-the-loop processes.

Participant 1 (green) emphasized that the framework is **currently better suited for manual decision-making**, particularly before automation is in place. They noted that in order for the framework to be integrated into automated pipelines, it would need to return a standardized response (e.g., success/warning/error) that could be interpreted programmatically. Until such integration features are added, the framework's main value lies in supporting technical teams during evaluation and deployment decisions.

Participant 2 (blue) strongly agreed with the statement, suggesting that the framework could support decision-making in an industrial process. They envisioned scenarios in which a technician or operator—not necessarily a programmer—could read the output of the framework and use it to decide whether a process should continue. The participant clarified that this would not replace automation but could be used to inform human operators who are monitoring model performance in production. The host further elaborated that the framework could expose a decision signal (e.g., a boolean status) through an API, which could be queried by external services or used as a "trigger" for follow-up actions.

Participant 3 (orange) similarly noted that, with proper production integration, the framework could significantly ease decision-making around **model versioning and deployment**. For example, teams could use it to determine whether a new model should replace one already in use. However, in its current state, the framework would primarily support *manual monitoring and decision-making*, rather than being directly tied into automated deployment flows.

In summary, participants agreed that the framework is **useful for ML-enabled projects**, particularly in aiding human decision-making during model evaluation and monitoring. With added integration features, it has the potential to expand into more automated scenarios, increasing its relevance for production environments.

RQ3.3 – I would use this framework in my future ML-enabled system projects

In this final question related to the TAM model, illustrated in Table 6.3, participants reflected on whether they could see themselves using the framework in their own ML projects. Responses to this statement showed a positive outlook, with participants selecting positions between *Partially Agree*

Table 6.3: Statement 3 of focus group opinion

	P1	P2	P3
Strongly Agree	X		X
Partially Agree	X	X	
Partially Disagree			
Strongly Disagree			
No Opinion			

and *Strongly Agree*, depending on the level of integration and automation currently supported by the framework.

Participant 1 (green) was torn between the *Strongly agree* and *Partially agree* positions. On one hand, the participant expressed a strong interest in adopting the framework, highlighting its dual utility for both **decision support** and **requirements alignment**. They particularly appreciated its ability to formalize the acceptance criteria around model performance. On the other hand, they noted that the absence of a **single consolidated status output** (e.g., a Boolean summary of the evaluation) limited its current applicability for automated monitoring workflows. It was emphasized that if such a signal were available, it could be easily integrated into APIs or containerized environments, enabling real-time evaluation and continuous deployment scenarios. The participant also suggested exposing the framework as a reusable Python library to improve usability across both development and production contexts.

Participant 2 (blue), who chose *Partially Agree*, described a conditional use case: they would adopt the framework in scenarios where **constant monitoring of model performance** was needed. In such settings, the framework could play a key role in surfacing evaluation results to support ongoing quality assurance. However, the participant refrained from selecting a stronger level of agreement due to their perception that additional infrastructure would be necessary for seamless integration.

Participant 3 (orange) selected *Strongly Agree*, stating they would actively explore how the framework could facilitate day-to-day tasks. They recognized that its current use is already compatible with development workflows, such as notebook-based evaluations, and expressed enthusiasm about how future enhancements—especially those discussed in the focus group—could significantly improve its usefulness. They saw value in testing it further and iterating over how it could be embedded into real-world processes.

In summary, participants were **willing to use the framework** in their projects, especially where it supports metric validation, communication with stakeholders, and monitoring model quality. To reach broader adoption, several participants emphasized the importance of features such as a general status

output, Python package distribution, and smoother integration into existing pipelines. These improvements would support both manual and automated usage across the ML lifecycle.

Suggestions for Improvement

At the end of the focus group session, participants were invited to share any additional suggestions that could help improve the framework. While many suggestions had already emerged organically throughout the discussion, this final segment served to consolidate and reflect on the most actionable ideas.

Publish the framework as a Python library. All participants emphasized the importance of making the framework more accessible and reusable by exposing it as a standard Python package. This would allow users to import it in notebooks or pipelines, pass model objects directly (rather than loading them from disk), and streamline testing workflows during model development. Participant 1 (green) noted that such a library could help *"the ML engineer utilize these metrics from the beginning of development in a simpler way"*.

Support for alternative model inputs (e.g., endpoints or runtime objects). Participants discussed alternatives to loading serialized models (e.g., pickled files). Suggestions included supporting endpoints—where the framework could send feature inputs to a running inference service—and adapting loaders to dynamically handle models from registries or cloud platforms like SageMaker. This would enable better compatibility with real-world deployment architectures.

Generalized support for custom models. Participant 1 proposed following the `scikit-learn` estimator interface to allow models from other frameworks (e.g., PyTorch) to be adapted with minimal effort. This could be achieved by implementing standard methods expected by the framework, leveraging Python's duck typing philosophy. This would increase flexibility and lower the barrier for integrating a wider variety of ML models.

Low-code and no-code interface. To increase accessibility for non-programmers or less technical users, participants recommended developing a low-code interface (e.g., through GUI-based configuration) or exposing the framework in simplified execution environments. Participant 2 (blue) suggested that this would make it more appealing for business users or operators monitoring models in production.

Improved outputs and monitoring support. There was strong interest in generating more **user-friendly outputs**, including reports that clearly

state whether all criteria were met and what went wrong when applicable. Ideas included returning a single `True/False` value, generating visual alerts, or sending notifications such as emails in response to failed tests. This would increase usability in continuous monitoring and facilitate human decision-making.

Enhanced support for mixed evaluation types. Participants discussed the practical value of supporting mixed evaluation modes—such as combining classification and regression within the same test suite. This was particularly relevant in use cases like soft classification or domain-specific interval-based labels (e.g., carbon footprint bands), where regression outputs are later discretized for classification purposes.

Built-in support for comparing model versions. Finally, participants suggested adding functionality to **compare different versions of a model**, either by performance metrics or configuration. This would help evaluate whether a new model version should replace the current one, and could be particularly helpful for monitoring model drift over time.

These suggestions point toward a clear roadmap for improving the framework's usability, flexibility, and production-readiness—while keeping it accessible to a wide range of stakeholders, from developers to customers and end users.

6.3 Threats to Validity

To ensure a rigorous evaluation of the BDD4ML framework, we identify and discuss the main threats to the validity of our findings, based on the categories suggested by WOHLIN et al. (2024).

6.3.1 Internal Validity

Internal validity concerns whether the observed effects can be attributed to the BDD4ML framework itself rather than external factors. A potential threat is the varying background of participants in the focus group, which may have influenced the quality and depth of feedback. Although the methodology was structured to minimize bias, differences in participants' prior knowledge of BDD or ML may have affected their ability to engage equally in the study.

6.3.2 Construct Validity

Construct validity refers to whether the study design and instruments adequately captured the intended concepts. The evaluation relied on partici-

pants' interpretations of BDD clauses and their ability to map them to ML model testing concerns. There is a threat that participants' understanding of "self-descriptiveness" in clauses may not fully align with the intended construct. Additional workshops or alternative formulations of clauses could have strengthened this aspect.

6.3.3 Conclusion Validity

Conclusion validity relates to whether the inferences drawn from the study results are sound. Due to the relatively small number of participants in the focus group, statistical generalization is limited. Moreover, the qualitative nature of the feedback may introduce subjectivity in interpreting results. The limited dataset restricts the robustness of conclusions, and a follow-up focus group with more developers would be ideal.

6.3.4 External Validity

External validity refers to the generalizability of the findings beyond the study context. The focus group was conducted primarily with participants who had limited industry experience. This poses a threat, as the feedback may not fully reflect the challenges and expectations of seasoned professionals in real-world ML testing environments. Future work should include additional sessions with industry experts who have more extensive practical experience, thereby improving the representativeness and applicability of the findings to professional settings.

6.4 Concluding Remarks

This chapter presented the static validation of the BDD4ML framework through a focus group with industry professionals, in alignment with Step 5 of Gorschek's technology transfer model (GORSCHEK et al., 2006). The evaluation was guided by the Goal-Question-Metric (GQM) approach (BASILI; ROMBACH, 1988), and focused on assessing the framework's perceived usefulness, ease of use, and behavioural intention to use, alongside its coverage and completeness.

The focus group revealed that participants generally perceived the framework as a practical and promising tool for structuring the validation of machine learning models. It was considered especially helpful for improving communication, automating quality assurance, and supporting decision-making pro-

cesses in ML-enabled systems. Initial feedback confirmed that the framework is aligned with practitioner needs, particularly when it comes to transparency, monitoring, and collaboration.

However, several limitations were also identified. Chief among them were the reliance on technical expertise, limited automation support, and insufficient integration with existing pipelines. Participants proposed a number of actionable enhancements, including the publication of the framework as a Python package, support for model endpoints and custom metrics, a user-friendly output format, and low-code interfaces to increase accessibility for non-technical users.

In summary, the static validation confirmed that the BDD4ML framework addresses relevant industry concerns and has the potential to bring significant value to real-world ML development workflows. The feedback gathered in this chapter provides a concrete foundation for guiding further iterations of the framework toward broader adoption and integration into production environments.

7
Conclusion

This dissertation aimed to explore and develop a (BDD) framework specifically designed for testing machine learning models, contributing to a novel area in software testing. The journey through the development of this framework has been closely aligned with a structured approach to the technology transfer model (TTM), as outlined by GORSCHEK et al. (2006).

In this concluding chapter, we summarize the contributions of this dissertation by revisiting the steps of the Technology Transfer Model (TTM) used for constructing and validating the BDD4ML framework in Section 7.1. We then discuss the limitations of the current work and outline directions for future research in Section 7.2, emphasizing the need for broader industrial validation, integration with MLOps practices, and extension of the framework to address additional concerns beyond model performance. Finally, in Section 7.3, we reflect on the broader implications of introducing BDD to ML-enabled system testing, highlighting its potential to standardize practices, align stakeholder expectations, and strengthen trust in intelligent systems.

7.1
Contributions

This dissertation was structured according to the Technology Transfer Model (TTM), moving from problem identification to industrial validation. Chapter 2 identified opportunities at the intersection of BDD frameworks and machine learning testing. Chapter 3 established a research agenda through a structured literature review. Chapter 4 presented the design and implementation of the BDD4ML framework, built using Python, Behave, and Scikit-learn. Chapter 5 evaluated the framework in an academic environment through an observational study. Finally, Chapter 6 validated the framework in an industrial focus group, providing insights for its applicability in real-world ML projects.

The main contributions of this dissertation are:

- A literature review on BDD frameworks and their applications in ML.
- The BDD4ML framework, openly available in a Zenodo repository.
- An observational study assessing the framework in an academic environment.
- An industrial focus group providing insights into practical applicability.

7.2 Limitations and Future Work

Limitations of this work lie in the scope of the industrial focus group conducted for validation. Although the discussion provided valuable insights into the applicability of BDD4ML in real-world ML projects, the number of participants was limited, and their expertise covered only a subset of the diverse roles typically involved in ML-enabled system development. Future research should therefore expand this evaluation by engaging a broader range of industry experts, including professionals with extensive experience in deployment, monitoring, and compliance. Such a diversified perspective would strengthen the external validity of the findings and support a more comprehensive understanding of the framework's applicability across different organizational and technical contexts.

As for future work, following the Technology Transfer Model, this dissertation concluded at step 5 with the static validation of BDD4ML. Naturally, the next step would be dynamic validation (step 6), which involves applying the framework in real industrial development settings to evaluate its effectiveness under practical constraints. However, this step lies beyond the scope of the present work but represents a key direction for future research.

Future work should also expand the scope of BDD4ML beyond model performance to include the other concerns defined in Section 4.1, such as accountability, explainability, system performance, data quality, and legality. Incorporating these additional concerns will provide a more comprehensive framework capable of addressing the multifaceted challenges of ML-enabled systems.

Furthermore, while this dissertation has contributed by bridging Behaviour Driven Development (BDD) with the testing of ML models, several avenues remain open for future research. One promising direction concerns the integration of framework specifications with the lifecycle challenges of ML-enabled systems, integrating it with MLOps principles (ARAUJO et al., 2024) and continuous experimentation practices.

Another opportunity involves considering different model export formats. Recent work has shown that the selection of an appropriate model export format (e.g., ONNX, SavedModel, TorchScript) significantly affects the integration, portability, and maintainability of ML-enabled systems (PARIDA; GEROSTATHOPOULOS; BOGNER, 2025). Building on these findings, future research could extend the BDD4ML framework to incorporate concerns related to model export and deployment formats. By aligning BDD4ML with MLOps and deployment-related concerns, future work could provide practitioners not

only with behaviour-driven specifications but also with actionable guidance on sustainability and maintainability in real-world ML-enabled systems.

Finally, broader studies with diverse participant groups and industry partners will be important to validate the framework's usability, scalability, and long-term impact. Such evaluations can inform refinements to the framework and contribute to establishing BDD4ML as a practical tool for bridging communication gaps and improving quality assurance in ML system development.

7.3 Concluding Remarks

In the face of growing complexity and heightened demand for the satisfaction of requirements such as fairness and explainability, there also seems to be a growing demand for testing standardization of systems with machine learning components. The lack of standards for the creation of these systems brings uncertainty to the machine learning development scene.

Cases of AI-powered systems violating ethical guidelines and laws have become increasingly frequent worldwide. Right now, it seems that most of the scene is sustained by the novelty aspect of the technology. Unfortunately, the novelty eventually dies down, and if nothing is done, all that is gonna be left are growing concerns about this technology's use. If these concerns are not addressed in the following years, we may find ourselves in a situation where these systems will be generally discredited.

BROOKS (1982) said in his book "The Mythical Man-Month: Essays on Software Engineering" that "The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence." Brook's renowned work emphasizes the complexity of software and the necessity of understanding and addressing this complexity through essential practices like software testing.

More often than not, when describing ML-enabled systems, customers may come with surreal expectations that the product will deliver on the most abstract of requirements. Usually, developers are the first ones to cut down on these expectations. However, since they do not have a common approach to testing these systems, they often decide to "let loose and fail fast", an adventurous but often dangerous approach. Testing anchors software development in reality and sets grounded expectations for all parties involved.

By introducing BDD to the creation of ML-enabled systems, we expect to position testing as the linchpin of intelligent systems development. Testing

may bring cohesion to the process for all stakeholders. By establishing commonly used metrics to satisfy different concerns, we may bring clarity and predictability to the use of these models and help identify the main problems to be addressed in each of them.

In conclusion, this dissertation represents a step toward advancing the field of BDD frameworks for machine learning model testing. The work presented here sets the stage for future research and development, aiming to bridge the gap between academic innovation and practical implementation, ultimately making machine learning model testing more accessible, transparent, and aligned with stakeholder needs.

8
Bibliography

ABUSHAMA, H. M.; ALASSAM, H. A.; ELHAJ, F. A. The effect of test-driven development and behavior-driven development on project success factors: A systematic literature review based study. In: IEEE. **2020 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE)**. [S.l.], 2021. p. 1–9.

ALVES, A. P. S. et al. Status quo and problems of requirements engineering for machine learning: Results from an international survey. In: KADGIEN, R. et al. (Ed.). **Product-Focused Software Process Improvement**. Cham: Springer Nature Switzerland, 2023. p. 159–174. ISBN 978-3-031-49266-2.

ARAUJO, G. et al. Professional insights into benefits and limitations of implementing mlops principles. In: **Proceedings of the 26th International Conference on Enterprise Information Systems-Volume 2: ICEIS**. [S.l.: s.n.], 2024. v. 2, p. 305–312.

BASILI, V.; ROMBACH, H. The tame project: towards improvement-oriented software environments. **IEEE Transactions on Software Engineering**, v. 14, n. 6, p. 758–773, 1988.

BECK, K. **Test Driven Development: By Example**. Boston: Addison-Wesley, 2003. ISBN 978-0321146533.

BINAMUNGU, L. P. Behaviour driven development: A systematic mapping study. **Journal of Systems and Software**, v. 203, p. 111749, 2023. ISSN 0164-1212. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0164121223001449>.

BROOKS, F. P. **The Mythical Man-Month: Essays on Software Engineering**. [S.l.]: Addison-Wesley Pub. Co., 1982.

BüNDER, H.; KUCHEN, H. A model-driven approach for behavior-driven gui testing. In: **Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing**. New York, NY, USA: Association for Computing Machinery, 2019. (SAC '19), p. 1742–1751. ISBN 9781450359337. Disponível em: <https://doi.org/10.1145/3297280.3297450>.

BÜNDER, H.; KUCHEN, H. Towards behavior-driven graphical user interface testing. **ACM SIGAPP Applied Computing Review**, ACM New York, NY, USA, v. 19, n. 2, p. 5–17, 2019.

CABRAL, R. et al. Investigating the impact of solid design principles on machine learning code understanding. In: **Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI**. New York, NY, USA: Association for Computing Machinery, 2024. (CAIN '24), p. 7–17. ISBN 9798400705915. Disponível em: <https://doi.org/10.1145/3644815.3644957>.

CONTRIBUTORS, W. **Behavior-driven development**. 2025. Wikipedia. BDD is considered a refinement of TDD and improves collaboration using domain language.

CONTRIBUTORS, W. **Extreme programming**. 2025. Wikipedia. Test-first development used as early as NASA's Project Mercury.

CRUZES, D. S.; DYBA, T. Recommended steps for thematic synthesis in software engineering. In: IEEE. **2011 international symposium on empirical software engineering and measurement**. [S.l.], 2011. p. 275–284.

CUI, J. A comparative study on the impact of test-driven development (tdd) and behavior-driven development (bdd) on enterprise software delivery effectiveness. 2024.

DAVIS, F. D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. **MIS quarterly**, JSTOR, p. 319–340, 1989.

DENG, Y. et al. Bmt: Behavior driven development-based metamorphic testing for autonomous driving models. In: IEEE. **2021 IEEE/ACM 6th International Workshop on Metamorphic Testing (MET)**. [S.l.], 2021. p. 32–36.

EVANS, E. **Domain-Driven Design: Tackling Complexity in the Heart of Software**. Boston, MA: Addison-Wesley, 2003. See Chapter 2: The Ubiquitous Language. ISBN 978-0321125217.

FAROOQ, M. S. et al. Behavior driven development: A systematic literature review. **IEEE Access**, IEEE, 2023.

FOWLER, M. **Test Driven Development**. 2023. <https://martinfowler.com/bliki/TestDrivenDevelopment.html>. Developed by Kent Beck in the late 1990s as part of Extreme Programming.

FUNG, C. P. et al. Towards accountability driven development for machine learning systems. In: MARTIN, K.; WIRATUNGA, N.; WIJEKOON, A. (Ed.). **CEUR Workshop Proceedings: Proceedings of the SICSA eXplainable Artificial Intelligence Workshop 2021**. [s.n.], 2021. (SICSA Workshop on eXplainable Artificial Intelligence (XAI) 2021, v. 2894), p. 25–32. Disponível em: <http://ceur-ws.org/Vol-2894/>.

GARCíA, M. V.; AZNARTE, J. L. Shapley additive explanations for NO2 forecasting. **Ecological Informatics**, v. 56, p. 101039, 2020.

GORSCHEK, T. et al. A model for technology transfer in practice. **IEEE Software**, v. 23, n. 6, p. 88–95, 2006.

JAMES, C. The impact of test-driven development (tdd) and behavior-driven development (bdd) on user satisfaction and stakeholder engagement. 2024.

KALINOWSKI, M. et al. Naming the pain in machine learning-enabled systems engineering. **Information and Software Technology**, Elsevier, p. 107866, 2025.

KEELE, S. et al. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.]: Technical report, ver. 2.3 ebse technical report. ebse, 2007.

KONTIO, J.; BRAGGE, J.; LEHTOLA, L. The focus group method as an empirical tool in software engineering. In: SHULL, F.; SINGER, J.; SJøBERG, D. I. K. (Ed.). **Guide to Advanced Empirical Software Engineering**. Springer London, 2008. p. 93–116. Disponível em: <https://doi.org/10.1007/978-1-84800-044-5_4>.

LAMA, J. O.; MONJE, M. R.; VELTHUIS, M. P. Entorno para la evaluación de la adecuación funcional en sistemas ia. In: **Actas de las XXVII Jornadas de Ingeniería del Software y Bases de Datos**. [S.l.: s.n.], 2023.

LAZĂR, I.; MOTOGNA, S.; PÂRV, B. Behaviour-driven development of foundational uml components. **Electronic Notes in Theoretical Computer Science**, Elsevier, v. 264, n. 1, p. 91–105, 2010.

LIAO, C.-F. et al. Toward a service platform for developing smart contracts on blockchain in bdd and tdd styles. In: IEEE. **2017 IEEE 10th Conference on service-oriented computing and applications (SOCA)**. [S.l.], 2017. p. 133–140.

MAFFEY, K. R. et al. **MLTEing Models: Negotiating, Evaluating, and Documenting Model and System Qualities**. 2023. ArXiv preprint arXiv:2303.01998. Disponível em: <https://dl.acm.org/doi/10.1109/ICSE-NIER58687.2023.00012>.

MENDES, E. et al. When to update systematic literature reviews in software engineering. **Journal of Systems and Software**, Elsevier, v. 167, p. 110607, 2020.

MISHRA, S.; STURM, B. L.; DIXON, S. Local interpretable model-agnostic explanations for music content analysis. In: **ISMIR**. [S.l.: s.n.], 2017. v. 53.

MOTTA, E.; KALINOWSKI, M. **Artifacts - BDD4ML: A Behavior-Driven Development Framework for Machine Learning Models**. Zenodo, 2025. Accessed: 2025-09-02. Disponível em: <https://doi.org/10.5281/zenodo.17041759>.

MUGHAL, A. H. Advancing bdd software testing: Dynamic scenario re-usability and step auto-complete for cucumber framework. **arXiv preprint arXiv:2402.15928**, 2024.

NAHAR, N. et al. A meta-summary of challenges in building products with ml components–collecting experiences from 4758+ practitioners. In: IEEE. **2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)**. [S.l.], 2023. p. 171–183.

NASCIMENTO, N. et al. Behavior-driven development: A case study on its impacts on agile development teams. In: **Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops**. New York, NY, USA: Association for Computing Machinery, 2020. (ICSEW'20), p. 109–116. ISBN 9781450379632. Disponível em: <https://doi.org/10.1145/3387940.3391480>.

NORTH, D. **Introducing BDD**. 2006. Accessed: 2024-08-26. Disponível em: <https://dannorth.net/introducing-bdd/>.

ORUÇ, A. F.; OVATMAN, T. Testing of web services using behavior-driven development. In: **CLOSER (2)**. [S.l.: s.n.], 2016. p. 85–92.

PARIDA, S. K.; GEROSTATHOPOULOS, I.; BOGNER, J. How do model export formats impact the development of ml-enabled systems? a case study on model integration. In: IEEE. **2025 IEEE/ACM 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)**. [S.l.], 2025. p. 48–59.

RAHARJANA, I. K. et al. Conversion of user story scenarios to python-based selenium source code for automated testing. **TEM Journal**, UIKTEN-Association for Information Communication Technology Education and ..., v. 12, n. 1, p. 309–315, 2023.

RAHARJANA, I. K.; HARRIS, F.; JUSTITIA, A. Tool for generating behavior-driven development test-cases. **Journal of Information Systems Engineering and Business Intelligence**, v. 6, n. 1, p. 27, 2020.

SCHULZ, H. et al. Behavior-driven load testing using contextual knowledge-approach and experiences. In: **Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering**. [S.l.: s.n.], 2019. p. 265–272.

SILVA, T. R. Towards a domain-specific language for behaviour-driven development. In: IEEE. **2023 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)**. [S.l.], 2023. p. 283–286.

SILVA, T. R.; FITZGERALD, B. Empirical findings on bdd story parsing to support consistency assurance between requirements and artifacts. In: **Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering**. [S.l.: s.n.], 2021. p. 266–271.

SILVA, T. R.; WINCKLER, M.; TRÆTTEBERG, H. Ensuring the consistency between user requirements and gui prototypes: A behavior-based automated approach. In: SPRINGER. **Human-Computer Interaction–INTERACT 2019: 17th IFIP TC 13 International Conference, Paphos, Cyprus, September 2–6, 2019, Proceedings, Part I 17**. [S.l.], 2019. p. 644–665.

SMART, J. F.; MOLAK, J. **BDD in Action: Behavior-driven development for the whole software lifecycle**. [S.l.]: Simon and Schuster, 2023.

SOLIS, C.; WANG, X. A study of the characteristics of behaviour driven development. In: IEEE. **2011 37th EUROMICRO conference on software engineering and advanced applications**. [S.l.], 2011. p. 383–387.

SOUZA, P. Lopes de; SOUZA, W. Lopes de; PIRES, L. F. Scrumontobdd: Agile software development based on scrum, ontologies and behaviour-driven development. **Journal of the Brazilian Computer Society**, Springer, v. 27, n. 1, p. 10, 2021.

TEUGELS, J. L. Mann–whitney test. In: **Wiley StatsRef: Statistics Reference Online**. John Wiley Sons, Ltd, 2014. Disponível em: <https://onlinelibrary.wiley.com/doi/10.1002/9781118445112.stat06547>.

TURNER, M. et al. Does the technology acceptance model predict actual use? a systematic literature review. **Information and Software Technology**, v. 52, n. 5, p. 463–479, 2010. ISSN 0950-5849. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0950584909002055>.

VILLAMIZAR, H. et al. Identifying concerns when specifying machine learning-enabled systems: a perspective-based approach. **Journal of Systems and Software**, Elsevier, v. 213, p. 112053, 2024.

WOHLIN, C. et al. Guidelines for the search strategy to update systematic literature reviews in software engineering. **Information and software technology**, Elsevier, v. 127, p. 106366, 2020.

WOHLIN, C. et al. **Experimentation in Software Engineering**. [S.l.]: Springer Nature, 2024.

A
Appendix

A.1
Table of Scores For Each Paper

Tables A.1 and A.2 show the final scores for including relevant papers in the literature review performed in Chapter 3.

Table A.1: Grading of Articles from Mapping Study

Article Title	**Study Relevance**	**Results and Findings**	**Quality of Reporting**	**Discussions and Implications**	**Methodological Rigor**
Behavior-driven load testing using contextual knowledge-approach and experiences	3	3	3	2	2
Tool for generating behavior-driven development test-cases	3	3	2	2	3
Towards behavior-driven graphical user interface testing	3	2	2	2	2
Toward a service platform for developing smart contracts on blockchain in bdd and tdd styles	3	2	2	3	2
Testing of web services using behavior-driven development	3	2	3	3	3
Ensuring the Consistency Between User Requirements and GUI Prototypes: A Behavior-Based Automated Approach	3	3	3	3	3
Behaviour-driven development of foundational uml components	3	3	3	3	3
A model-driven approach for behavior-driven gui testing	3	3	2	2	3
BMT: Behavior Driven Development-based Metamorphic Testing for Autonomous Driving Models	3	3	3	3	3

Continued on next page

Continued from previous page

Article Title	Study Relevance	Results and Findings	Quality of Reporting	Discussions and Implications	Methodological Rigor
Towards accountability driven development for machine learning systems	3	1	2	3	1

Table A.2: Grading of Articles from Forward Snowballing

Article Title	**Study Relevance**	**Results and Findings**	**Quality of Reporting**	**Discussions and Implications**	**Methodological Rigor**
Conversion of User Story Scenarios to Python-Based Selenium Source Code for Automated Testing	3	3	3	3	3
Advancing BDD Software Testing: Dynamic Scenario Re-Usability And Step Auto-Complete For Cucumber Framework	3	3	2	3	3
ScrumOntoBDD: Agile software development based on scrum, ontologies and behaviour-driven development	3	2	3	3	3
Towards a Domain-Specific Language for Behaviour-Driven Development	3	2	2	2	2
Empirical Findings on BDD Story Parsing to Support Consistency Assurance between Requirements and Artifacts	3	2	3	3	3
A Study of the Characteristics of Behaviour Driven Development	3	3	3	3	3
The effect of Test-Driven Development and Behavior-Driven Development on Project Success Factors: A Systematic Literature Review Based Study	3	3	2	3	3

Continued on next page

Continued from previous page

Article Title	**Study Relevance**	**Results and Findings**	**Quality of Reporting**	**Discussions and Implications**	**Methodological Rigor**
Behavior Driven Development: A Systematic Literature Review	3	3	3	3	3