Gabriel de Andrade Busquim

On the Interaction between Software Engineers and Data Scientists when Building Machine Learning-Enabled Systems

Dissertação de Mestrado

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.

Advisor : Prof. Marcos Kalinowski
Co-advisor: Profª Maria Julia Dias de Lima

Rio de Janeiro
April 2024

Gabriel de Andrade Busquim

On the Interaction between Software Engineers and Data Scientists when Building Machine Learning-Enabled Systems

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:

Rio de Janeiro, April 30th, 2024

Gabriel de Andrade Busquim

Graduated in Computer Engineering from the Pontifical Catholic University of Rio de Janeiro. Pursued a Master's Degree in the Department of Informatics, specializing in Software Engineering.

Bibliographic Data

Busquim, Gabriel

On the Interaction between Software Engineers and Data Scientists when Building Machine Learning-Enabled Systems / Gabriel de Andrade Busquim; advisor: Marcos Kalinowski; co-advisor: Maria Julia Dias de Lima. – 2024.

119 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2024.

Inclui bibliografia

1. Aprendizado de Máquina. 2. Sistema Habilitado por Aprendizado de Máquina. 3. Ciência de Dados. 4. Engenharia de Software. 5. Colaboraçãoo. I. Kalinowski, Marcos. II. Lima, Maria Julia. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Acknowledgments

First, I would like to thank my dissertation advisor, Marcos Kalinowski, and my co-advisor, Maria Julia Lima, for their guidance during this research. Furthermore, I would like to thank all professionals from ExACTa and Tecgraf who participated in our studies for their insights during our sessions. They contributed immensely to our research and my development as a software engineer. Finally, I thank my family and friends who supported me during this journey.

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

Abstract

Busquim, Gabriel; Kalinowski, Marcos (Advisor); Lima, Maria Julia (Co-Advisor). **On the Interaction between Software Engineers and Data Scientists when Building Machine Learning-Enabled Systems**. Rio de Janeiro, 2024. 119p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In recent years, Machine Learning (ML) components have been increasingly integrated into the core systems of organizations. Engineering such systems presents various challenges from both a theoretical and practical perspective. One of the key challenges is the effective interaction between actors with different backgrounds who need to work closely together, such as software engineers and data scientists. This work presents three studies investigating the current interaction and collaboration dynamics between these two roles in ML projects. Our first study depicts an exploratory case study with four practitioners with experience in software engineering and data science of a large ML-enabled system project. In our second study, we performed complementary interviews with members of two teams working on ML-enabled systems to acquire more insights into how data scientists and software engineers share responsibilities and communicate. Finally, our third study consists of a focus group where we validated the relevance of this collaboration during multiple tasks related to ML-enabled systems and assessed recommendations that can foster the interaction between the actors. Our studies revealed several challenges that can hinder collaboration between software engineers and data scientists, including differences in technical expertise, unclear definitions of each role's duties, and the lack of documents that support the specification of the ML-enabled system. Potential solutions to address these challenges include encouraging team communication, clearly defining responsibilities, and producing concise system documentation. Our research contributes to understanding the complex dynamics between software engineers and data scientists in ML projects and provides insights for improving collaboration and communication in this context. We encourage future studies investigating this interaction in other projects.

Keywords

Machine Learning; ML-enabled System; Data Science; Software Engineering; Collaboration.

Resumo

Busquim, Gabriel; Kalinowski, Marcos; Lima, Maria Julia. **Sobre a Interação entre Engenheiros de Software e Cientistas de Dados Construindo Sistemas Habilitados por Aprendizado de Máquina**. Rio de Janeiro, 2024. 119p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nos últimos anos, componentes de aprendizado de máquina têm sido cada vez mais integrados aos sistemas principais de organizações. A construção desses sistemas apresenta diversos desafios, tanto do ponto de vista teórico quanto prático. Um dos principais desafios é a interação eficaz entre atores com diferentes formações que precisam trabalhar em conjunto, como engenheiros de software e cientistas de dados. Este trabalho apresenta três estudos distintos que investigam as dinâmicas de colaboração entre esses dois atores em projetos de aprendizado de máquina. Primeiramente, realizamos um estudo de caso exploratório com quatro profissionais com experiência em engenharia de software e ciência de dados de um grande projeto de sistema habilitado por aprendizado de máquina. Em nosso segundo estudo, realizamos entrevistas complementares com membros de duas equipes que trabalham em sistemas habilitados por aprendizado de máquina para obter mais percepções sobre como cientistas de dados e engenheiros de software compartilham responsabilidades e se comunicam. Por fim, nosso terceiro estudo consiste em um grupo focal onde validamos a relevância dessa colaboração durante várias tarefas relacionadas à sistemas habilitados por aprendizado de máquina e avaliamos recomendações que podem melhorar a interação entre os atores. Nossos estudos revelaram vários desafios que podem dificultar a colaboração entre engenheiros de software e cientistas de dados, incluindo diferenças de conhecimento técnico, definições pouco claras das funções de cada um, e a falta de documentos que apoiem a especificação do sistema habilitado por aprendizado de máquina. Possíveis soluções para enfrentar esses desafios incluem incentivar a comunicação na equipe, definir claramente responsabilidades, e produzir uma documentação concisa do sistema. Nossa pesquisa contribui para a compreensão da complexa dinâmica entre engenheiros de software e cientistas de dados em projetos de aprendizado de máquina e fornece recomendações para melhorar a colaboração e a comunicação nesse contexto. Incentivamos novos estudos que investiguem essa interação em outros projetos.

Palavras-chave

Aprendizado de Máquina; Sistema Habilitado por Aprendizado de Máquina; Ciência de Dados; Engenharia de Software; Colaboração.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
ODR	Online Dispute Resolution
RTA	Reflexive Thematic Analysis
TA	Thematic Analysis
UX	User Experience

1
Introduction

1.1
Context and Motivation

In the past decades, computer processing power and the amount of data produced have grown absurdly, creating a prosperous field for technological innovations such as Artificial intelligence (AI) and Machine Learning (ML). ML is used today in various applications, from processing natural language to recommending products to customers. Integrating ML components into existing systems has increased as companies seek to leverage vast amounts of data to enhance the business outcomes of their software products. In this dissertation, we refer to these systems as ML-enabled systems, since their behavior is dictated by explicitly defined rules and the data used by the ML component to make decisions. Typically, the ML component is only a small part of a larger system, which usually comprises other components for data collection, model consumption, and infrastructure requirements (Sculley et al., 2015).

This transition from developing traditional software systems to those integrated with ML components introduces new challenges from the software engineering viewpoint (Wan et al., 2019). The development of ML-enabled systems often involves completely separate workflows and different actors who must efficiently work together throughout various stages of the project (Aho et al., 2020). A team working on an ML-enabled system usually comprises:

– Business owners, who must have a comprehensive view of the problem at hand to set goals for the system based on business objectives;
– Designers, responsible for modeling the users' interaction with the model to maximize their satisfaction and engagement with the system;
– Domain experts, who use their knowledge of the domain to help other stakeholders by providing context to the analyzed data and the model's results;
– Data scientists, who play a vital role in preprocessing and analyzing the data, together with developing, tuning, and monitoring the performance of the model;

- Software engineers, responsible for designing how other system components will interact with the ML component and supervising scalability and maintainability for the ML-enabled system.

The last two actors mentioned above are responsible for developing and integrating the model with the system, and an ineffective interaction between them can cause misconceptions capable of harming the system (Lewis et al., 2021). This scenario highlights the importance of proper alignment and communication between the actors. It is vital for managers and team leaders to understand each actor's responsibilities inside the project and how to overcome challenges during their collaboration.

1.2 Research Goal and Objectives

Our goal with this study is to thoroughly illustrate how software engineers and data scientists collaborate when developing an ML-enabled system. We define collaboration, or interaction, as having two or more actors working together to accomplish a given task. This means that they must communicate with each other and divide responsibilities during task execution. In the context of this work, we do not consider isolated questions from one actor to another as collaboration, as this does not imply both of them are sharing responsibility over that task.

Investigating this interaction in-depth allows us to visualize where it can be improved. With this research, we seek to obtain recommendations that can promote better collaboration practices for software engineers and data scientists. Not only does this facilitate team management, but it can also enhance the team's performance.

As we investigate collaboration procedures inside teams developing software products with ML components, we also uncover new research possibilities. Another objective of this work is to analyze the current literature depicting the interaction between software engineers and data scientists in the context of ML-enabled systems and how it can be expanded. Based on the literature and our findings, we present several future work opportunities that can be explored by other researchers interested in this topic.

1.3 Methodology

To accurately characterize collaboration in a real-world context, we discussed this topic with data scientists and software engineers currently developing ML-enabled systems for industry-academia projects. This dissertation comprises three studies we conducted, each involving different teams and participants.

Our first study consists of a case study with a team developing an ML-enabled system for helping parties handle legal conflicts. We interviewed two data scientists and two software engineers from the team to map how they shared responsibilities and communicated since the beginning of the project. The participants revealed the challenges they faced due to ineffective collaboration during multiple tasks, such as integrating the ML model with other system components and updating the project's documentation.

In our second study, we interviewed members of two other teams building ML-enabled systems for a customer in the Oil and Gas sector. We used an interview script similar to the one designed in our case study to see if any new perspectives emerged. While we still noticed similarities between the challenges faced by the teams, the participants' statements complemented our vision of collaboration in ML projects. For example, we saw how data scientists may interact with software engineers who are not part of their team but work directly for the project's customer.

Finally, our third study comprises a focus group with experienced data scientists and software engineers. This study examined how collaboration between these two actors can affect tasks they must execute during development. Moreover, we asked participants to evaluate several recommendations we proposed based on the literature and our findings for improving this interaction. As a result of this study, we obtained a detailed view of the importance of collaboration for each task. We were also able to assess the advantages brought by each recommendation we had defined.

1.4 Dissertation Outline

This dissertation comprises six chapters. Chapter 2 explains the background behind ML-enabled systems and the challenges encountered by the software engineering community while building them. We also detail related work on the topic of ML systems development and collaboration practices adopted by teams developing them. Chapter 3 portrays our case study, presenting our research methodology and the results we obtained. Chapter 4 reports the complementary interviews we conducted to enrich our case study findings. Chapter 5 illustrates the focus group we performed with practitioners experienced in data science and software engineering. Finally, Chapter 6 summarizes our research's contributions and limitations, which may be addressed in future works.

2
Background and Related Work

2.1
Introduction

In this chapter, we present an overview of ML-enabled systems and the challenges associated with them. We examine the current literature to understand what studies have already been conducted and the main findings concerning collaboration between software engineers and data scientists. This investigation helped us comprehend how to plan our studies so that they could enrich the existing literature.

This chapter contains four sections. Section 2.2 briefly explains what ML is and how it differs from traditional non-ML systems. Section 2.3 discusses several papers in the field of software engineering for ML-enabled systems. Section 2.4 contains our concluding remarks.

2.2
Background

Within the domain of AI, ML stands out as a technology with profound contemporary relevance. An ML model is characterized by its capacity to learn through training and make predictions. ML applications are extremely diverse, ranging from speech recognition to product recommendation. As this field grows, it becomes extremely important to understand how to design, develop, and monitor ML systems efficiently. This involves preventing bugs and guaranteeing the desired performance in a production environment. The process behind creating an ML model is not a consensus between all teams, but they usually follow a similar workflow. This workflow, depicted in the work of Amershi et al. (2019), is illustrated in Figure 2.1.

The first stage involves determining model requirements, such as the system features that could be implemented with ML and the most appropriate ML algorithms for the problem. In the Data Collection stage, the team looks for datasets to train the model. It is important to remove any inaccurate data from the dataset, which is done in the Data Cleaning step. Truth labels may also be assigned to the data records depending on the chosen learning technique.

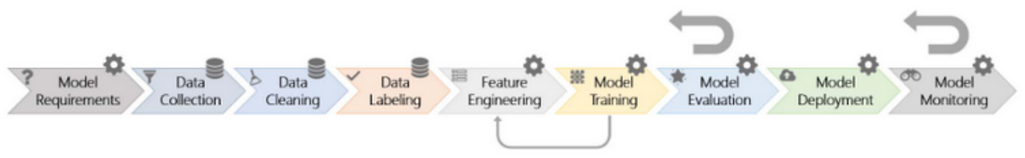

Figure 2.1: Stages of an ML Workflow

In the Feature Engineering stage, the dataset features the model will use are defined and extracted for model training. After that, the model's output is evaluated and adjusted until it is ready to be deployed. The model must be constantly monitored so errors and performance decays are instantly detected and resolved. ML development is an iterative process. Hence, model evaluation and training may occur multiple times during the project. This can happen when new training data is acquired, or when definitions made during the model requirements stage need to be revisited.

2.2.1 Challenges in Building ML-enabled Systems

Villamizar et al. (2024) define ML-enabled systems as software systems with an ML component. The development of ML-enabled systems presents several challenges that can significantly impact the interaction between team members. This is the case especially for software engineers and data scientists, who often share responsibilities for handling data and deploying models. For example, designing an appropriate architecture for these systems is not trivial, as the team must evaluate factors such as model performance degradation, uncertainty management, and proper integration between the model and other system components (Nazir et al., 2023).

Furthermore, requirements engineering practices for non-ML software development are not entirely applicable when developing systems with an ML component (Ishikawa and Yoshioka, 2019). There is a typical lack of requirements specifications for ML-enabled systems that clearly define the input data, expected model outputs, and how the ML component should integrate into the larger system (Villamizar et al., 2024). Without these specifications, data scientists may create models with assumptions that software engineers are unaware of, leading to integration issues when transitioning from development to production.

The different backgrounds of data scientists and software engineers can also impact their interactions. While data scientists may have strong mathematical and statistical skills, software engineers have expertise in programming, software design, and system architecture (Kim et al., 2017). This diversity can lead to variations in problem-solving approaches. In addition, their cultural differences can also play an important role. While the tasks performed by data scientists revolve around experimentation and dealing with the uncertainty of unpredictable results, software engineers often adhere to structured development methodologies (Aho et al., 2020). These cultural disparities can cause barriers in a collaborative environment.

2.3 Related Work

During our research, we found several studies investigating ML-enabled systems. We organized them into different categories according to their research topic. This categorization allowed us to understand how the current literature discusses challenges for building systems with ML components through different perspectives.

2.3.1 Papers investigating Challenges for ML Development

The literature confirms how developing ML components defies current software engineering practices. Wan et al. (2019) conducted a mixed study comparing ML and non-ML systems to grasp their most important differences according to practitioners' views. They noticed significant differences in both software engineering activities and work characteristics. For example, components' coupling in ML systems is less emphasized than in non-ML systems. In addition, ML systems rely heavily on data availability, quality, and experimentation. This dependency on data, in turn, introduces uncertainties when establishing system requirements and estimating task effort. Given this scenario, results showed that communication between team members is vital. Respondents stressed the importance of informing business owners about what is possible or not with ML models, which may be dictated by data availability. They also mentioned challenges when discussing project progress with customers due to difficulties interpreting the model's results.

Rahman et al. (2022) described challenges and lessons learned from an industry-academia collaborative ML-enabled system project for error correction in retail transactions. The authors reported challenges faced from a software engineering and an ML perspective. The discussed challenges portray how the development of ML components affects multiple phases of the software development life cycle, such as eliciting requirements, integrating all ML and non-ML system components, and evaluating infrastructure for deployment.

Amershi et al. (2019) presented a case study with Microsoft software development teams to gather best practices for ML systems. After grouping the participants according to their experience with AI, the authors ranked the most relevant challenges for each group. The results showed that "data availability, collection, cleaning, and management" was ranked first for all groups. This illustrates how the success of ML-enabled systems depends on quality data, which mobilized the teams toward data reusability between different projects and rigorous data versioning controls. Another challenge ranked high regardless of participants' experience with AI was "collaboration and working culture." The authors enforced that software engineers with traditional systems development knowledge must be able to work alongside ML specialists.

2.3.2 Papers investigating Actors inside the Team

Some studies explore the perspectives of specific actors inside teams building ML-enabled systems, such as UX designers. Zdanowska and Taylor (2022) interviewed interdisciplinary teams with designers to uncover how collaboration happened during the project and the main challenges they faced when designing an ML system. Results showed that the teams made design decisions collectively in multiple project stages. Involving technical team members in design discussions from the beginning allowed designers to familiarize themselves with technical details, which prevented issues later in the project. Furthermore, designers helped technical team members better understand system requirements and think more about the user when discussing the product. The authors cited siloed organizational structures, in which actors struggle to communicate with each other, as a challenge for collaboration. Another obstacle reported was aligning product expectations with business owners and requesting buy-in for project continuity.

In another study investigating UX practitioners, Yang et al. (2018) interviewed thirteen designers to understand their activities and extract insights for UX in an ML context. The findings illustrated the participants' design process, which started with identifying a goal for the ML system together with the data scientists. As designers experimented with different ideas, data scientists tested them and returned possible technical limitations. This iterative process created a shared vision of the product and the company's goals, considered critical by all participants. Based on these results, the authors emphasized how UX educators should provide opportunities for students to engage in interdisciplinary collaborative experiences, as this would help qualify them for the industry.

Previous studies also depict the perspective of data scientists. Kim et al. (2017) surveyed 793 data science employees and enthusiasts at Microsoft to uncover their work activities and obstacles. While respondents pointed out challenges related to data characteristics, such as availability and quality, a separate category of challenges was dedicated to team interaction. For instance, participants struggled with effectively sharing insights with leaders and achieving agreement among all stakeholders. To address this, the authors emphasized the importance of specifying and clarifying the goals of the project together with the whole team. This recommendation can also help convince business owners of the value of collecting quality data, which was another challenge respondents mentioned.

Begel and Zimmermann (2014) asked 1500 Microsoft software engineers what questions they would most like data scientists to answer, which were subsequently prioritized by 2500 Microsoft engineers in another survey. While grouping the first survey questions into categories, the authors saw the need to create a category dedicated to collaboration, as participants were interested in practices that could improve the interaction within and between teams. For example, they inquired about methods for sharing knowledge inside the team and defining code ownership. Collaboration also appeared in the results of the second survey. The question “How can we improve collaboration and sharing between teams?” was highly ranked by multiple respondents.

2.3.3 Papers investigating Team Interaction

Papers analyzing collaboration and communication investigate ML teams and projects involving data science in general. Zhang et al. (2020) designed an online survey to understand how data science workers collaborate. This survey was answered by 183 IBM employees, including data scientists and software engineers. The findings reported that data scientists strongly participate in all stages of the projects, which suggests they may be responsible for guiding the team’s activities. On the other hand, software engineers are more involved in core technical activities, such as acquiring data for the model. The paper also presented an overview of code and data documentation practices inside the teams, which the authors highlighted as key for collaborative practices.

Lewis et al. (2021) studied the consequences of ML mismatches between data scientists, software engineers, and operations staff developing ML-enabled systems. The authors defined ML mismatches as problems caused by inaccurate assumptions these actors had about the system that could have been prevented through knowledge sharing. This study revealed that most mismatches were related to incorrect assumptions about the ML model. Participants reported a lack of model specifications, APIs, and test cases for integration testing. They also declared that data scientists trained the ML models and passed them to the software engineers for integration with the other system components. These findings confirm the existence of mismatches in the collaboration between data scientists and software engineers during product-model integration.

In a more recent paper, Mailach and Siegmund (2023) examined sociotechnical challenges for bringing ML-enabled software into production. After analyzing 73 videos in which practitioners shared their experiences developing ML-enabled systems, the authors identified 17 antipatterns for productionizing ML. Most were not related to technical aspects, but instead caused by organizational characteristics. For example, a subset of antipatterns were induced by organizational silos, especially between the data science and software engineering teams. The findings reported that tension, communication issues, and difficulties during model integration and deployment characterized the interaction between these two actors. Because of this, the process of bringing the model to production became delayed, tedious, and error-prone.

Also focusing on these two roles, Nahar et al. (2022) interviewed 45 participants working on software projects with ML components to identify challenges and recommendations for the interaction between these actors. The authors identified three activities that required collaboration: identifying and decomposing requirements, negotiating training data quality and quantity, and integrating data science and software engineering work. During these tasks, participants reported challenges such as data scientists working isolated from software engineers, insufficient system documentation, and problems with responsibility sharing.

2.4 Concluding Remarks

In this chapter, we reviewed several papers illustrating obstacles teams developing ML-enabled systems must confront. While some refer to development and system design, the literature also confirms the existence of considerable issues related to the interaction between team members, especially data scientists and software engineers. When discussing the state of the art, Nahar et al. (2022) mentioned they were unaware of other studies examining challenges between these two actors. With our work, we intend to expand the literature on this topic and provide additional insights and useful recommendations for practitioners.

3
Case Study

3.1
Introduction

This chapter depicts the case study we conducted to comprehend the interaction dynamics between software engineers and data scientists. Choosing a case study strategy allowed us to examine this interaction thoroughly. It also enabled us to qualitatively understand the responsibilities these actors had during the execution of the selected case project. In this work, we followed the guidelines proposed by Runeson et al. (2012) for case study research in software engineering.

The selected case concerns an ML-enabled system for Online Dispute Resolution (ODR) created to help parties settle legal disputes in Rio de Janeiro. Beyond describing the team and system context, we conducted semi-structured interviews with two experienced software engineers and data scientists to understand their interactions and challenges in ML projects. To this end, we asked them about activities covering the development process end-to-end. Our questions ranged from defining requirements to analyzing data and integrating the ML model with the rest of the system. We transcribed and analyzed the interviews using Reflexive Thematic Analysis (RTA), one of the Thematic Analysis (TA) family methods (Braun and Clarke, 2006, 2019). This approach guided us while analyzing the data and finding patterns among the interviewees' points of view.

We detail the goal of this study in Table 3.1. We followed the Goal-Question-Metric (GQM) template for goal definition, a structured approach commonly used in software engineering and related fields (Basili and Rombach, 1988). This approach helps to establish a clear connection between the research's goal, the specific questions that need to be answered, and the metrics used to measure progress. From this goal, we derived the following research questions.

Table 3.1: Case Study Goal

Analyze	the interaction between software engineers and data scientists
for the purpose of	characterization
with respect to	responsibility sharing and collaboration
from the point of view of	experienced software engineers and data scientists
in the context of	a large ML-enabled system project for ODR to help settle legal disputes.

RQ1: How do software engineers and data scientists share responsibilities when developing an ML-enabled system?

This research question investigates what tasks are carried out by software engineers and data scientists during the development of ML-enabled systems. It focuses on how responsibilities are shared between these two pivotal roles, providing insights into the task allocations and synergies that contribute to the successful creation of ML-based solutions. To answer *RQ1*, we evaluated the participation of software engineers and data scientists in multiple stages of the ML-enabled system's creation, such as during the system's design and model development. For each activity, we mapped the actors involved and if any collaboration happened.

RQ2: How do software engineers and data scientists collaborate when developing an ML-enabled system?

This question focuses on the collaborative practices between software engineers and data scientists developing ML-enabled systems. It seeks to uncover the nature of their interactions and communication methods, contributing to a comprehensive understanding of the collaborative processes inherent in this interdisciplinary context. To this end, we asked participants about their perceptions of how this interaction unfolded inside the team. We encouraged them to highlight challenges and improvement possibilities.

This chapter comprises five sections. Section 3.2 details the selected case and how we designed our research methodology. Section 3.3 presents the case study results and the participants' statements. Section 3.4 discusses the participants' perceptions and the answers to our research questions. Section 3.5 depicts our concluding comments.

3.2 Case Study Design

The selected case concerns +Acordo, an ODR system project. It was created to help parties settle legal disputes in Rio de Janeiro. The system uses ML to generate settlement agreements for cases with low legal complexity, therefore avoiding litigation. Since the whole process happens inside the platform, the system can solve its users' problems in an agile and affordable way, minimizing costs and bureaucracy. We chose to focus on this particular project because it is centered around the development of an ML-enabled system, which is aligned with the scope of our investigation. Furthermore, we had easy access to project participants and the complete system documentation.

+Acordo started in 2021 inside PUC-Rio's Tecgraf Institute through a partnership with the Rio de Janeiro State Court. After applying the Lean Inception methodology developed by Caroli (2017), the team defined the product's main functionalities. Given the system's goal, building an ML component to aid in dispute resolution was considered an interesting choice. This led to the incorporation of data scientists into the team, which also began participating in meetings to discuss model characteristics. For the system's first version, the team partnered with an electric power company and established their focus on disputes involving consumer complaints directed to this company. The company representatives then developed external APIs that the system would consume to obtain part of the data required by the model.

The team responsible for developing +Acordo is multidisciplinary. It comprises a product owner, domain experts, UX designers, data scientists, and software engineers. They all attended meetings to understand business rules and discuss product ideas. Customer representatives attended these meetings as well to ensure decisions followed their expectations. Besides specifying requirements, they were also responsible for evaluating the team's deliveries. They could interact with the system through release versions made available by the software engineers every two months. Each release version contained a new major functionality developed and tested by the team.

The team follows the Agile software development methodology, using the Scrum framework to structure their tasks. Work is divided into sprints that last two weeks. Before each sprint, the team meets to plan their activities. Development tasks are thoroughly discussed during this planning stage to make sure the steps needed to complete each task are clear to everyone. After that, the members gather daily to discuss their activities. At the end of each sprint, the team evaluates whether they accomplished their goals. Each team member expresses positive and negative points about the sprint, extracting lessons learned to benefit future sprints. After that, a new development iteration starts with the beginning of a new sprint.

The team building the ML-enabled system comprises six software engineers and two data scientists, considered part of two separate squads. Each squad has its tasks, as well as its own planning and daily meetings. However, the teams share the same product owner. The product owner gets together weekly with the teams' leaders to discuss their activities, evaluate their progress, and align future steps.

Figure 3.1 shows an overview of the system's architecture. Users have access to the system's functionalities through a web application. It allows them to create an account on the platform, register disputes, and view the settlement agreements generated. The application communicates with back-end services through a REST API.

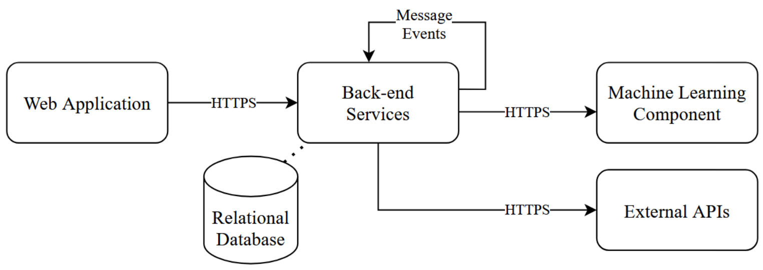

Figure 3.1: Simplified System Architecture

The system's back-end architecture is based on microservices. In this architectural style, services are decoupled, providing more flexibility when deploying and managing the system. Each service has a single responsibility, such as processing disputes, creating agreements, or generating documents. The services communicate both synchronously and asynchronously. Synchronous communication happens through REST APIs, while asynchronous communication occurs via message queues. The developers implemented an event-driven architecture, where the system components can communicate asynchronously through messages to perform their tasks.

A relational database was used to store registered users and disputes. The systems' APIs access the database to display information in the web application and provide the software's functionalities. As disputes may have fields specific to a particular company, the team created different tables in the database to support multiple data types. Besides communicating with its back-end, the system also communicates with external services. This communication occurs for various reasons, such as complementing dispute data through the REST APIs provided by the partner company. The ML model uses the obtained data to determine the creation of a settlement agreement.

One of the system's back-end services communicates with the ML component through a REST API. The model's input consists of data entered by users on the web platform and complementary data obtained from external APIs. As an output, the model returns whether it can generate a settlement agreement. If the result is positive, the model returns all agreement parameters. If not, the model returns why it could not create an agreement. The model consists of a decision tree, which contains a set of fixed rules defined by customer representatives and the partner company that must be validated before the system can generate a settlement agreement. These rules were created to restrain the model's possible outputs and improve transparency.

After verifying all rules, the model then evaluates data from previously resolved disputes. After selecting and analyzing those most similar to the dispute in question, the model defines the ideal value for each settlement agreement parameter, such as the value for compensating moral damages. Other information about this subset of disputes is also extracted and displayed to the user, such as the average time taken to resolve them through litigation. The text classification method behind the model's functionality is described in the work of Coelho et al. (2022).

Figure 3.2 illustrates our steps during the execution of this case study. The first step was to design the interview script, which we discuss in the next section. After that, we conducted the interviews and transcribed all recordings. In the last step, we used RTA to analyze the data. This process is detailed in Section 3.2.2.

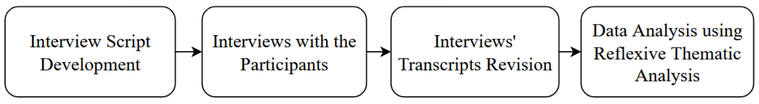
Interview Script Development
Interviews with the Participants
Interviews' Transcripts Revision
Data Analysis using Reflexive Thematic Analysis

Figure 3.2: Research Steps

3.2.1 Data Collection

To elaborate our interview questions, we used the work of Villamizar et al. (2022b) as a reference. The authors developed *PerSpecML*, a perspective-based approach for specifying ML-enabled systems. Through this approach, it is possible to analyze several concerns that must be addressed when planning these systems. *PerSpecML*'s goals include improving decision-making as the project progresses and identifying trade-offs between conflicting requirements and design decisions. In addition, it aims to foster collaboration and communication between the stakeholders involved in the system. To fulfill these goals, this approach evaluates four major components: stakeholders, perspectives, concerns, and tasks. Below, we discuss each one of these elements.

PerSpecML provides a holistic view of the development of ML-enabled systems. For this reason, it involves multiple project stakeholders. In the first version of *PerSpecML*, Villamizar et al. (2022b) defined four actors of interest:

- Business owners, who must align the system's goals with business objectives. They must know what ML can do to set impactful and realistic goals.
- Designers, responsible for creating an engaging user experience for the system. A well-designed interaction with the ML model provides value to end users, whose feedback contributes to the quality of the model over time.
- Data scientists, who analyze the data and develop the ML component. To do this, they must understand the data's context, business needs, and model requirements.
- Software engineers, who implement ML model integration with the rest of the system. Beyond assessing the interaction between system components, they must also monitor performance and scalability.

After evaluating and refining their approach, Villamizar et al. (2024) also included domain experts in the final version of *PerSpecML*. They play a fundamental part in the team by supplying real-world knowledge to ensure the ML-enabled system correctly addresses its challenges. Their collaboration with other stakeholders, such as business owners and data scientists, is crucial for model creation and interpretation.

Perspectives represent particular aspects of a system. The authors separated the specification of an ML-enabled system into five different perspectives, each with its own characteristics. They are presented in Table 3.2.

Table 3.2: Evaluated Perspectives

Perspective	**Description**
ML/System Objectives	Involves understanding the problem to be solved by the ML-enabled system and defining the model's goals.
User Experience	Involves designing an appropriate interaction between the user and the model.
Data	Addresses how data is obtained and analyzed to build the model.
Model	Concerns defining the model's inputs and outputs and evaluating its performance.
Infrastructure	Covers establishing a robust infrastructure for the ML-enabled system to function properly.

A concern is a specific issue that needs to be considered when developing the system. In total, the authors identified 59 concerns, which are covered by all perspectives. The concerns were allocated into 28 groups, each representing a task the team should perform. Each task is associated with one of the perspectives described above and has at least one suggested stakeholder responsible for executing it. Tasks may have more than one stakeholder, which implies that the involved actors will have to work together while analyzing the task's concerns. The authors also mapped relationships between tasks to highlight how one may depend on another. These relationships illustrate the importance of cross-functional collaboration inside the team.

All *PerSpecML* components are integrated into the perspective-based ML task and concern diagram. The diagram presents an extensive view of the activities required for developing successful ML-enabled systems. Moreover, it models the tasks and related concerns faced by different stakeholders in ML projects. We used the initial version of Villamizar et al. (2022b)'s perspective-based ML task and concern diagram to design our interview script. We chose this version since it was the most recent one at the time of the interviews. Using this diagram, we initially mapped tasks associated with the infrastructure perspective involving either a software engineer or a data scientist. After that, we investigated tasks from other perspectives, which gave us a broader view of the responsibilities both of these actors may have inside a project. Figure 3.3 displays the diagram with the tasks we examined highlighted.

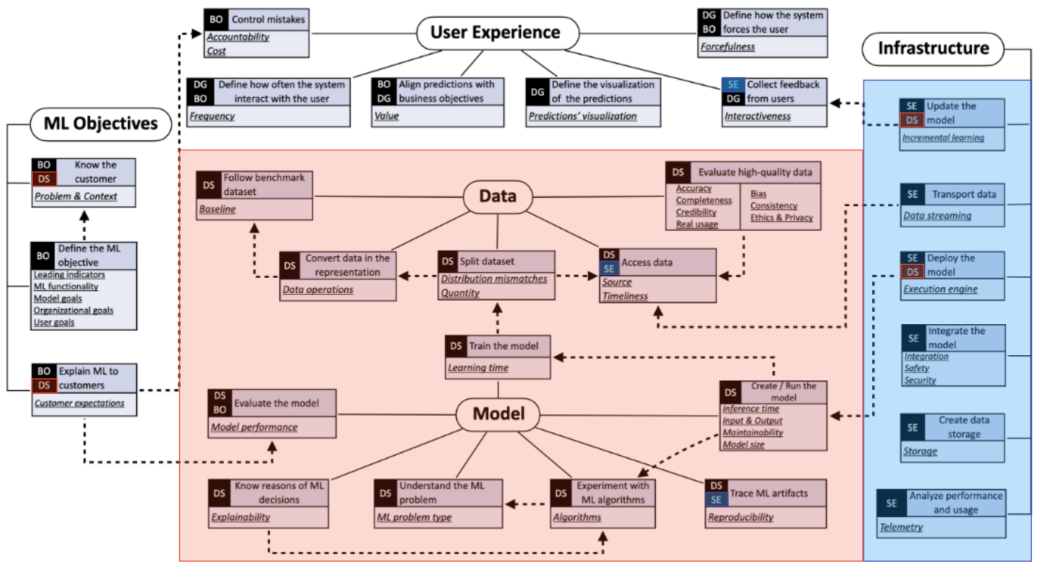

Figure 3.3: The initial version of the perspective-based ML task and concern diagram. The red rectangles indicate tasks performed by data scientists, while the blue ones indicate those performed by software engineers. The dashed arrows indicate relationships between the tasks

This analysis confirmed the participation of either data scientists or software engineers in at least one task from every perspective. Furthermore, we noticed some tasks involve data scientists and software engineers simultaneously, indicating the need for collaboration between them. Hence, we designed our interview script to investigate how participants handled all of these tasks in the context of their project. For each highlighted task in Figure 3.3, we questioned participants about:

– What they did during the task.

– Their interaction with a data scientist, software engineer, or other actors on that task.

– Perceived difficulties or improvement opportunities during task execution.

– If the task originated any documentation.

We used this interview script to guide our discussions with the participants. The interviews were conducted on the Google Meet platform and lasted between thirty and fifty minutes. We asked open-ended questions so participants could provide as much detail as possible. Depending on their answers, we also asked questions outside our script to better comprehend their perspectives. In cases where participants had difficulties understanding the question, the author responsible for the interviews provided examples and context to help clarify the subject. Participants could also address any unmentioned topics related to our research that they deemed relevant. We recorded all interviews and, to transcribe them, we used Google Cloud's Speech-to-Text API[1]. The Google Cloud Platform provides a console where it is possible to upload a WAV audio file and download the corresponding transcript in TXT format.

3.2.2 Analysis Procedure

For analyzing the data, we followed the guidelines for RTA defined by Braun and Clarke (2006, 2019). Although RTA is widely used in psychology research, studies have shown that it can be applied in other fields, such as software engineering (Cruzes and Dyba, 2011) and human-computer interaction (Brown and Stockman, 2013). We decided to use RTA in our research since it allows us to engage analytically with the data. In other types of TA methodologies, such as coding reliability approaches, the analysis summarizes what was said about a particular topic (Braun and Clarke, 2021). In our case, we were interested in finding and interpreting patterns inside the data to fully understand the scenario illustrated by our participants and extract the main

[1]`https://cloud.google.com/speech-to-text`

challenges they reported. Following the recommendations of Braun and Clarke (2021), we did not consider using grounded theory due to the small size of our sample and the fact we do not have the goal of developing a theory.

The first phase of RTA is to become familiarized with the data, which we did while reviewing the transcripts and listening to the recordings. After that, we started the coding process. During this process, we aim to group different data components so that all information covering a given topic is in the same category. To do this, we first read each transcript thoroughly. Then, for each relevant text fragment, we create a code. As we keep reading, we either assign more sentences to one of the codes or create a new one. We followed an inductive approach for coding, where codes are developed using the data itself as a starting point.

With the codes defined, we then grouped them into themes. To develop themes, we must look for similarities between the codes. Themes should be objective and underpinned by a central concept. They must contain useful information about the dataset, directly addressing at least one research question. Following RTA recommendations, we iteratively refined the themes until they met these criteria.

3.2.3 Analysis Execution

The interview transcripts have between 855 and 1020 words. Our first step during analysis was translating them from Portuguese to English and correcting words while listening to the recordings. Some words had been incorrectly transcribed, while others were not transcribed at all. It was also necessary to add punctuation to the sentences and remove direct references to participants' names to guarantee anonymity. The revised transcriptions are illustrated in Appendix B. After applying the coding process, we acquired 47 codes. By examining the codes and looking for patterns across them, we extracted 18 candidate themes. Figure A.1 inside Appendix A illustrates the codes and themes obtained during this analysis.

The next step of RTA is to review and name the themes. In this process, we inspect our list of candidate themes, check if they reflect the coded data, and if they effectively contribute towards answering our research questions. During this step, our first action was to reduce the number of themes. Although Braun and Clarke (2012) state there is no exact formula for defining an ideal number, they warn that having too many themes can harm analysis coherence and decrease the level of detail each one presents. Therefore, we decided to reduce the number of themes to no more than six, an upper-bound value cited by the authors. To promote this reduction, we transformed the labels we had created to categorize our findings (represented by the red rectangles in Figure A.1 inside Appendix A) into themes. With that, our previous list of candidate themes became sub-themes. We also adjusted the names of the newly created themes and sub-themes to make them clearer and more attractive to readers.

Another change we made concerns the themes related to system documentation. Initially, we had created different themes for project definitions' documentation and team interaction while documenting the software. After analysis, we concluded that these themes share the same central concept. For this reason, we transformed these two themes into sub-themes of a new theme, "Documentation Development." Figure A.2 inside Appendix A portrays the new themes and sub-themes.

After restructuring our themes, we developed a thematic map. A thematic map consists of a visual representation of the themes and sub-themes developed throughout the analysis, allowing us to evaluate how representative they are of the collected data. Moreover, by mapping relationships between the themes, we can visualize them in relation to the others. Figure 3.4 illustrates our thematic map for this case study.

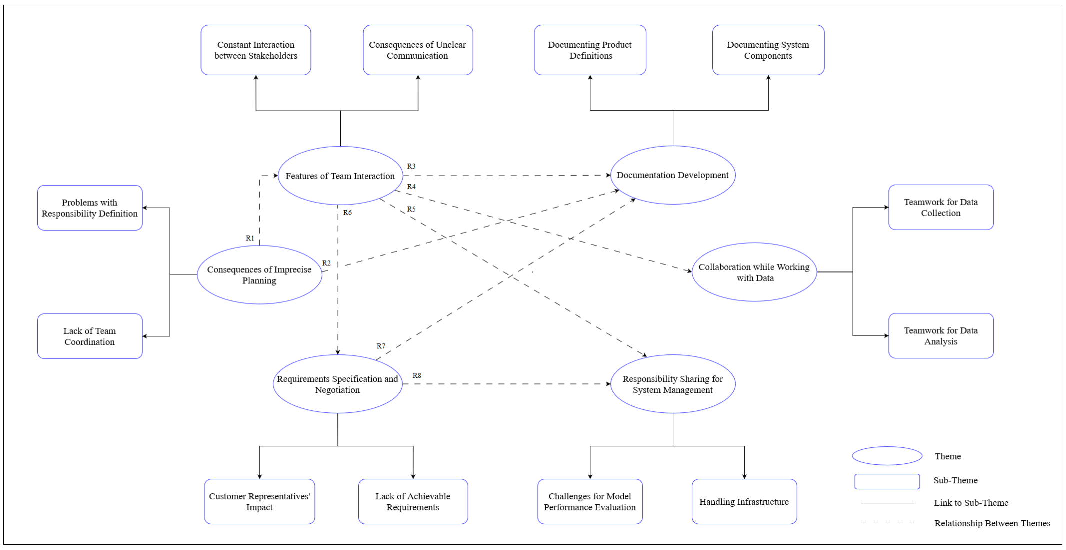
Constant Interaction between Stakeholders
Consequences of Unclear Communication
Documenting Product Definitions
Documenting System Components
Features of Team Interaction
Documentation Development
R3
R4
R5
R6
R1
R2
R7
R8
Problems with Responsibility Definition
Lack of Team Coordination
Consequences of Imprecise Planning
Teamwork for Data Collection
Collaboration while Working with Data
Teamwork for Data Analysis
Requirements Specification and Negotiation
Responsibility Sharing for System Management
Customer Representatives' Impact
Lack of Achievable Requirements
Challenges for Model Performance Evaluation
Handling Infrastructure
Theme
Sub-Theme
Link to Sub-Theme
Relationship Between Themes

Figure 3.4: Thematic Map for the Case Study

The thematic map contains eight relationships between themes, which are explained in Table 3.3. As we describe each one of the themes and sub-themes in the next section, it will become clear how these relationships unfolded inside the team.

Table 3.3: Thematic Map Relationships

Relationship	Description
R1	Participants were unhappy with how the team shared responsibilities during the planning stage.
R2	Data scientists and software engineers agreed to document different system components.
R3	Participants struggled while updating documentation.
R4	Collaboration between our participants was important during data analysis.
R5	Our participants did not communicate much while executing their tasks.
R6	The team had to make some decisions regarding model requirements by themselves.
R7	Participants documented the ML-enabled system's definitions.
R8	The lack of achievable requirements affected how model performance was monitored.

3.3 Case Study Results

All four participants verbally agreed to participate voluntarily in the study and have their interviews recorded. We assigned them an identification number, which we will use in the following sections to ensure their anonymity. All subjects identified as male and hold a master's or a doctorate degree. Table 3.4 shows the roles, education level, and years of work experience for each one.

Table 3.4: Participants' Demographic Data

Participant ID	Role	Education Level	Years of Experience
DS1	Data Scientist	Doctorate	8
DS2	Data Scientist	Doctorate	8
SE1	Software Engineer	Master's degree	11
SE2	Software Engineer	Master's degree	12

As depicted in Figure 3.4, we extracted six themes in total: **Requirements Specification and Negotiation**, **Consequences of Imprecise Planning**, **Collaboration while Working with Data**, **Responsibility Sharing for System Management**, **Features of Team Interaction**, and **Documentation Development**. Below, we describe each one in detail. We included paraphrased statements from the participants to support our analysis and interpretations.

3.3.1 Requirements Specification and Negotiation

Two sub-themes related to requirements emerged from the qualitative analysis: **Lack of Achievable Requirements** and **Customer Representatives' Impact**. Managing the requirements for the ML-enabled system was a challenge, as participants described a lack of achievable requirements. Data scientists mentioned that model requirements were unrealistic and unclear at the beginning of the project. DS2 stated:

> "*The requirements were abstract, like 'the model needs to be fast' or 'the system needs to be easy to use.' There was a misalignment between what was desired and what was possible, which led to many meetings.*"

Customer representatives helped to define the requirements for the model. SE1 gave examples of their participation:

> "*I noticed that customer representatives could actively suggest model parameter adjustments. Another topic they discussed was keeping information about the model's operation private from end users. This was done to prevent them from learning how to manipulate the model in their favor.*"

DS2 also recognized the importance of customer involvement, mentioning that he felt like customer representatives could have participated more:

> "*We had difficulties because we did not include more customer representatives when we defined the product's concepts. They could have helped us by making decisions. Instead, we made decisions internally. We had to revisit some of these decisions later, while we were lucky not to in others.*"

Participants also emphasized that requirements constantly changed after discussions with customer representatives. DS1 provided an example:

> "*In the beginning, we had defined that the model would be as flexible as possible. We realized during later meetings this would not be well accepted, as it would make the model's results less predictable.*"

3.3.2 Consequences of Imprecise Planning

Two sub-themes related to planning emerged from the analysis: **Problems with Responsibility Definition** and **Lack of Team Coordination**. Data scientists performed activities out of their field of expertise, such as eliciting requirements for the system. DS2 explained:

> "*Our team was responsible for understanding the entire business flow and legal procedures so that we could build the model. Someone else could have done this survey and delivered the requirements to us.*"

The data science team also developed the model consumption API. In DS2's view, this should have been done by the software engineers:

> "*We were a research team, not a development team. Still, we needed to develop versions and generate specifications for the model. Our team was responsible for developing and maintaining the model consumption API. This responsibility could have been given to the software engineering team.*"

Software engineers and data scientists struggled when planning their tasks. They tried to plan their activities separately, only communicating when necessary. SE2 explained this process:

> "*We created a REST API to allow the model's integration with the system. We defined a communication interface for the API, and then each team did its part. It was outside the data science team's interest to understand how we stored the data as long as this service existed.*"

Nevertheless, some participants were unhappy with this decision, especially with the coordination between the two teams. Each team had its own goals for each sprint, and dependencies between them were not always correctly mapped. DS2 stated:

> "*There was a misalignment in planning regarding each team's dependencies. For example, software engineers sometimes depended on a change in the model that was not in our backlog. The roles of each team ended up not being clear, which led to problems in the API used to consume the model. We lacked comprehensive planning that involved both teams more.*"

3.3.3 Collaboration while Working with Data

The two sub-themes related to data that emerged from the analysis were **Teamwork for Data Collection** and **Teamwork for Data Analysis**. There was a collaboration effort between software engineers and data scientists to collect data for the model. All participants confirmed the data science team was responsible for analyzing and documenting the data, and no software engineer participated in these activities. Even though software engineers did not directly analyze data, they collaborated with data scientists on other tasks. They were responsible for obtaining the data from customer representatives and making it available to the data science team, as explained by DS1:

> "*Since we worked with legal processes containing sensitive data, we needed a secure way to obtain them. The development team defined how this would be done together with customer representatives. They created a tool to download the data and make it available on our server. This download is done manually whenever new data are available.*"

Participants also reported it took time to obtain the data needed for the model. DS2 described how this situation led to undesired development tasks:

> "*We also had problems with data availability. It took us some months to get all the valid data needed for testing. Therefore, we had to initially use mocked data, which later became different from the real data, leading to rework.*"

Data scientists continuously analyzed data during model creation, and both DS1 and DS2 agreed that this analysis was not trivial. DS2 explained:

> "*Preprocessing the data was complex. We received raw data, so cleaning procedures were necessary, and we also put a lot of effort into annotating the data. It took a lot of effort to analyze and process the data received so that we could work on the model. This situation also affected what algorithms we could use for the model.*"

As requirements for the model constantly changed, so did its input data fields. Data analysis uncovered new fields that needed to be included, as explained by DS1:

> "*Throughout the project, we discussed what data we would take into account to generate the settlement agreement, and it took us a while to figure out what data we needed to request from the customer. We defined some data fields during development, while others were defined during meetings*".

Collaboration between team members enabled data analysis to be less challenging. As raw data was received in files, the software engineers created a tool to help with this process. DS1 explained:

> "*The development team helped us to create a text annotation system and make it available to the domain experts. They [the domain experts] indicated which document parameters were most interesting for extraction and annotated the data we used for model training.*"

Domain experts were considered vital for data understanding and analysis. They helped the data scientists throughout the project, as highlighted by DS1:

"*The domain experts were always by our side to answer questions, which was essential for building the model.*"

3.3.4 Responsibility Sharing for System Management

Two sub-themes related to system management emerged from our analysis: **Handling Infrastructure** and **Challenges for Model Performance Evaluation**. While the data science team was responsible for data analysis and building the model, the software engineers developed the web application and the back-end services that consume the model. They also handled the model's API deployment and infrastructure. SE1 stated:

"*We are responsible for deploying the model consumption API. The deployment of this service is automated through a CI/CD pipeline.*"

SE2 explained why the software engineers had the responsibility of deploying the model:

"*We already had a standard procedure for deployment beforehand, and we knew the data scientists did not specialize in DevOps, so we left this structure ready for them.*"

While software engineers monitored the model's infrastructure, data scientists had to monitor its performance. DS1 and DS2 said it was difficult to define metrics for the model, such as a target accuracy. DS2 mentioned:

"*The time taken to obtain valid data hindered the time to create a better performance evaluation framework*".

To solve this, the team agreed to assess model results with customer representatives. DS2 explained:

"*We presented model studies to the project's stakeholders for them to evaluate if the results were adequate or not.*"

DS1 also mentioned that customer representatives validated the model:

"*A supervisory committee, composed of customer representatives, was responsible for validating the results produced.*"

When asked about implementing incremental learning for the model, participants said this was a future goal. Although customer representatives desired this feature, the team did not prioritize it. The process of retraining the model is currently a manual task, as described by DS1:

> “*We defined that, from time to time, the model would be retrained with new data to update the parameters used.*”

3.3.5 Features of Team Interaction

The sub-themes related to team interaction that emerged from our analysis were **Consequences of Unclear Communication** and **Constant Interaction between Stakeholders**. There were communication issues between software engineers and data scientists, which caused problems in the ML-enabled system. SE1 explained the discovery of errors in the data received by the model:

> “*I did not have the necessary knowledge to analyze if the data were correct and what fields were required or optional. Problems only appeared when we started testing and integrating the system with external APIs. Only then did we notice data either missing or in the wrong format. If the teams had not been so distant, we could have anticipated these problems.*”

SE2, on the other hand, exemplified communication issues by explaining how the team should have discussed how to store the model artifacts:

> “*We provided Git repositories for this storage, but the teams did not discuss how the data scientists would store the artifacts. This eventually caused issues because the model had a lot of artifacts, such as the training scripts, which were not separated from the API code. For this reason, large files were loaded unnecessarily every time a new model release was generated.*”

We noted that software engineers did not know much about the model. Since data scientists and software engineers had different responsibilities, they became unaware of each other’s activities. SE1 explained his view of this situation:

> “*We were very separated, and I did not like that. We did not know much about the model. It was like a ’black box’ [...]. Even with a well-defined API, things that were obvious to the data science team were unclear to us. [...] We only developed the services that consumed the model, so we did not know what was being done.*”

This situation proved to be a problem when defining the model’s input, as SE1 highlighted:

> "*When we met with customer representatives and data scientists to map the data required by the model, I was unsure if the data we requested was correct because I did not know what the data scientists expected for the model's input.*"

Software engineers and data scientists struggled to communicate changes in the model's API. SE2 stated:

> "*Problems in the ML-enabled system were caused by changes in the interface established for the API.*"

DS2 also expressed dissatisfaction with errors in the model's input:

> "*Problems with input data formats when calling the model's API should not have been our responsibility, as this data had to be in the expected form before communication happened. However, we had to build workarounds to correct input data formats, which made the system's integration with the model take time and generate rework.*"

SE1 noticed a misunderstanding about the data between data scientists, software engineers, and customer representatives. He explained:

> "*In meetings, I noticed a mismatch regarding the participants' understanding of the data. For example, I expected the model's input to be in a particular format. Yet, the data scientists expected them in another format, and what the customer representatives understood differed from what the data scientists were expecting. This situation provoked changes in the model consumption API throughout the system's development.*"

SE2 also viewed the teams' separation as a problem, but was wary of harming productivity:

> "*If both teams were closer, we could have avoided these problems. However, we would also have had a higher overhead, as everyone would need to be together in all meetings.*"

The interviews showed how software engineers and data scientists had to communicate with other project stakeholders. For instance, both actors interacted with customer representatives to define requirements. SE2 gave an example:

> "*We had several discussions with customer representatives to understand their product vision and define what was possible. From there, the UX designers started to prototype ideas that we later used to model the system's database*".

Software engineers and data scientists also communicated with each other during development, and we noticed how they had a good relationship. DS1 emphasized this by stating:

> "*We do not have any problems in terms of communication between the teams, as the software engineers are very attentive and available to us. When there is a change, like new data that needs to be included in the API, or when there is an issue, we communicate directly through messages.*"

3.3.6 Documentation Development

Finally, the two sub-themes we extracted related to documentation were **Documenting Product Definitions** and **Documenting System Components**. The team documented the ML-enabled system's definitions and recognized the importance of doing so. DS2 explained:

> "*Each model definition was documented through presentations we did [...] to showcase our team's progress. The architecture of the model was also described in a document.*"

DS1 highlighted the importance of documenting each meeting:

> "*We created a flowchart with all the rules the model considered and documented the meetings through minutes. We even had an episode where it was necessary to resort to these minutes to prove that the team had made certain decisions in a previous encounter.*"

DS1 also mentioned how these documents helped him learn about the project when joining the team:

> "*Several research reports were developed at the beginning of the project. I consulted one about the system, which contained some use cases explaining the problem we had to solve. I also consulted another report on decision trees and other AI techniques that were researched. These documents helped me understand the business faster.*"

On the other hand, SE1 joined the team in the middle of the project and explained that he did not know about documentation that could have helped him during his onboarding:

> "*I became aware of the model's objective and the system's architecture during the project. I used to ask the data science team questions when I had doubts. There was no formal passage of knowledge, but instead explanations on demand.*"

Software engineers and data scientists did not cooperate much when producing documentation, as each team was responsible for documenting different system components. DS1 explained:

> "*The software engineers documented the input data, while we documented the output data.*"

However, changes in the system harmed this process, as explained by SE2:

> "*Our biggest challenge was with changes. The system's initial state was well-documented, but then changes started happening. These changes were not documented properly, which harmed the alignment between the teams. We did not correctly update the documentation throughout the project, and we also did not communicate these changes efficiently. We discovered them as system components stopped working.*"

Likewise, SE1 was not fully satisfied with the system's documentation:

> "*Our system is not documented well enough, even though we currently have the model's output and input data documented. For example, in the middle of this integration, a mapper converts data to the format expected by the model. We could have documented this conversion better.*"

3.4 Discussion

3.4.1 How do software engineers and data scientists share responsibilities when developing an ML-enabled system?

Data scientists and software engineers had specific responsibilities in the project. Data scientists focused on analyzing data and developing the ML model and its consumption API. On the other hand, software engineers were responsible for the model's infrastructure and the back-end services that access it.

Both teams shared responsibilities with other project stakeholders. For example, they participated in meetings with customer representatives to define the ML-enabled system's goals and functionalities. Data scientists worked closely with domain experts to understand data fields and discover new ones subsequently included in the model's input. Software engineers discussed interface layouts with UX designers before implementing them on the system's web application. These findings align with Zhang et al. (2020)'s results, which indicate that software engineers and data scientists are present in different project stages, from developing the system to communicating with business stakeholders.

Participants illustrated multiple interaction points between the software engineering and data science teams. They had several meetings to define model inputs and outputs and to enable model integration with the rest of the system. The same happened during data collection, when software engineers helped data scientists obtain the data for model training. Both teams also interacted during data annotation, as the software engineers created a system to help with this process. The developed tool allowed domain experts to select text areas inside files and associate them with a data field, structuring the data for the model.

The interviews revealed that DS2 was not pleased with all the responsibilities data scientists received. They had to map the business flow behind processing legal disputes, elicit requirements for the model, and present ideas to customer representatives. Besides that, not all decisions regarding model features they had to make could be validated with the customer. DS2 also explained the data science team's participation in developing the model consumption API. Even though they were a research team, they implemented all the API's code, which was a skill they did not have much experience with. For this reason, software engineers helped them during this process. Software engineers also handled the model's infrastructure and deployment pipelines, since this was another skill the data scientists did not possess. Data scientists struggling with ML infrastructure was one of the concerns mentioned by Nahar et al. (2022).

Team members performing activities outside their field of expertise highlights an opportunity to improve planning, which was another topic mentioned during the interviews. Participants revealed that features developed by software engineers could not be deployed because data scientists had to prioritize other functionalities. Although both teams tried to work as independently as possible, having such dependencies effectively mapped and planned could have enhanced the team's delivery speed and avoided problems during the model's integration.

3.4.2 How do software engineers and data scientists collaborate when developing an ML-enabled system?

Participants viewed communication between the software engineering and data science teams positively. Both teams had a good relationship and were always helpful when a member had doubts. They had a group chat where they could interact at any given time. However, their communication could have been more efficient. Both teams worked almost independently, which reduced the frequency of interactions between them and led to software engineers not having much knowledge about the model. Even though the model had a well-defined API, which was discussed by both teams, SE1 and SE2 used the term "black box" to describe the system's ML component. This lack of knowledge became evident during meetings with customer representatives, as there was a mismatch across the participants' understanding of the data. For example, one of the software engineers could not evaluate if the requested data was sufficient for the model, nor if they were in the expected format. This situation caused errors in the ML-enabled system that were discovered only during testing, resulting in avoidable rework.

Constant changes in requirements, also observed in Wan et al. (2019)'s work, worsened the ineffective communication between data scientists and software engineers. As new requirements appeared, both the model and the system had to be updated. The data science team had to implement new data fields for the model's input and business rules for the model's output. At the same time, software engineers needed to change the system to capture such data fields, either by user input or through accessing an external API. These changes provoked errors in the system because they were not communicated properly among the teams. For this reason, data scientists had to develop adaptations in the model consumption API to accommodate different input data formats.

The team strived to document product definitions and the ML-enabled system architecture. Meeting decisions were registered in minutes, and data scientists and software engineers were responsible for documenting different system components. Software engineers documented the model's input data and the back-end services that consumed the model. On the other hand, data scientists documented the business rules behind the model's behavior and its output responses. This separation of responsibilities made maintaining the documentation harder. New features were constantly being developed, and the team struggled to update the documents. The aforementioned inefficient communication of changes in the system was another obstacle when updating documentation. For example, problems with changes in the model's input were fixed by creating mappers that corrected the format of input data fields, and one participant mentioned that these mappers could be better documented.

Communication between the data science and software engineering teams was essential for one participant who joined the team after product development had already started. SE1's understanding of the ML-enabled system's objectives and architecture was acquired through conversations and questions to the team, as no formal documentation was presented to him. The data fields used by the model are very specific to its domain, which makes understanding them difficult for someone unfamiliar with all the business rules related to legal procedures.

3.4.3 Threats to Validity

This section discusses threats to validity, focusing on four types of threats: construct validity, internal validity, external validity, and reliability.

Threats to **construct validity** relate to our applied research methodology not being suited to answer our research questions, as participants' reports may differ from reality. To mitigate this threat, we consulted the project's documentation, such as use case diagrams and system architecture documents. This documentation was used to cross-check the participants' statements. Moreover, two other researchers who collaborated with this study revised the transcriptions, codes, and themes generated during the analysis. At the same time, the coding process in RTA is inherently subjective (Braun and Clarke, 2019), where researchers use their own experiences while interpreting the data.

In terms of **internal validity**, threats include methodological errors that may harm the truthfulness of the results for our population, such as asking incoherent questions to the participants. To mitigate them, we formulated the interview questions based on the findings of Villamizar et al. (2022b), which were acquired through a literature review (Villamizar et al., 2021) and reports of industrial experiences with ML systems (Villamizar et al., 2022a). We also explained the questions in detail when the participants expressed doubts to leave as little room for misunderstandings as possible. We recognize the number of participants, which was limited because of the team's size, may affect the credibility of our results. To mitigate this, we interviewed the team's most experienced software engineers and data scientists.

External validity concerns how our findings can be generalized. We understand that our case study only discussed challenges from a single team working in a specific ML-enabled system. It is possible to have scenarios where, for example, the same team is responsible for all tasks carried out by software engineers and data scientists. In other cases, a project manager might define responsibilities more formally, which can alter the team's collaboration procedures. However, given that some of our results are also present in the current literature, we believe our case study provides additional insights that can be considered when analyzing these two actors' interactions.

Reliability assesses to what extent the study depends on the researchers. To improve reliability, besides the peer-reviewed qualitative procedures, we uploaded the transcription of each interview to an online repository[2]. Making this material available enables auditing our analysis and facilitates the replication of our study.

3.5 Concluding Remarks

In this chapter, we investigated the interaction between data scientists and software engineers through a case study with a team developing a large ML-enabled system project. We interviewed two experienced members of each role about their activities and collaboration practices. We used RTA to inspect the interview transcripts and extract relevant data to answer our research questions. This case study was also illustrated in a paper written by Busquim et al. (2024), presented at the 2024 Software Quality Days (SWQD) conference[3].

The results gave us an overview of how the team organized tasks inside the project and the challenges data scientists and software engineers faced. These include actors being unaware of each other's activities, frequent requirement changes, unsynchronized planning, and outdated documentation of the ML-enabled system.

[2]`https://doi.org/10.5281/zenodo.10035304`
[3]`https://www.software-quality-days.com/en/`

4 Complementary Interviews

4.1 Introduction

In addition to the case study, we conducted semi-structured interviews with two other teams working with ML-enabled systems. We did this to obtain new perspectives on the topic of collaboration and investigate similarities with our case study results. As we had previously examined only a single team working on a specific solution, we could not guarantee our findings could be useful for other teams working with ML-enabled systems. Even though our case study's outcomes are also mentioned in the existing literature, it is important to understand them in the context of other projects for the industry. Interviewing more practitioners allows us to amplify our vision of the challenges behind the interaction between software engineers and data scientists. It also helps us to evaluate the external validity of our case study's findings.

Both teams we interviewed work on projects from ExACTa, an agile experimentation initiative from PUC-Rio. ExACTa's mission is to enable digital transformation through research and industry-academia partnerships, fostering innovation and improving operational efficiency. The projects we analyzed were developed for a customer in the Oil and Gas industry. Even though they are for the same customer, each project has an individual goal, architecture, and team responsible for it.

Since this study aims to enrich the findings of our case study, we defined a similar goal for it, shown in Table 4.1. We also decided to use the same research questions, namely "*How do software engineers and data scientists share responsibilities when developing an ML-enabled system*" and "*How do software engineers and data scientists collaborate when developing an ML-enabled system*". Maintaining the same research questions is vital to our objective, as it allows us to compare this study's findings with our previous results.

Table 4.1: Complementary Interviews Goal

Analyze	the interaction between software engineers and data scientists
for the purpose of	characterization
with respect to	responsibility sharing and collaboration
from the point of view of	software engineers and data scientists
in the context of	two ML-enabled system projects for the Oil and Gas industry.

This chapter contains six sections. Section 4.2 illustrates how we designed our interview script and collected the data for our research. Section 4.3 presents the first project we analyzed, together with participants' perceptions. In Section 4.4, we do the same for the second project we investigated. Section 4.5 contains our discussion of the interviews' results. Section 4.6 comprises our final remarks.

4.2 Data Collection

We followed the same methodology applied in the case study to design our interview script. However, in this study, we used a more recent version of the perspective-based ML task and concern diagram. The process behind developing this new version comprised several validations of the previous version together with academic students and industry representatives, as reported by Villamizar et al. (2024). Once again, we mapped all tasks with a software engineer or data scientist as responsible. Figure 4.1 presents the diagram with both roles highlighted.

Some differences can be perceived between the initial diagram and the improved version. First, the authors removed relationships between tasks, which are now established between concerns. Besides that, the actors responsible for some tasks changed, and new tasks were added. For instance, the User Experience perspective now has a task "Ensure the credibility of predictions," which involves a data scientist. Moreover, by analyzing the Infrastructure perspective, we can see a new task involving data scientists and software engineers, "Automate End-to-End ML workflow."

For each task highlighted in the diagram, we asked the same questions as in the case study, namely:

– What was the subjects' participation in that task, if any.

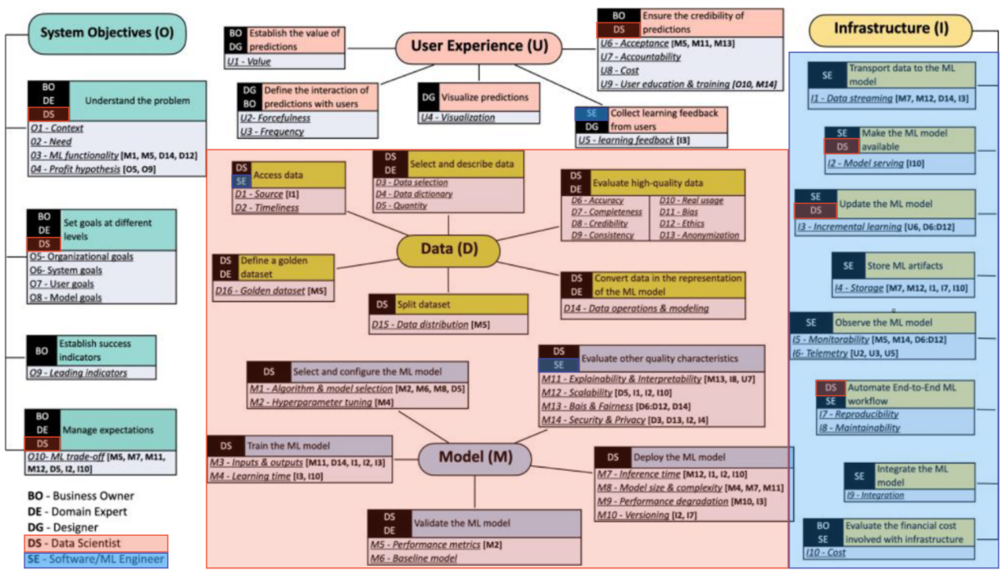
System Objectives (O)
BO DE DS Understand the problem
O1 - Context
O2 - Need
O3 - ML functionality [M1, M5, D14, D12]
O4 - Profit hypothesis [O5, O9]
BO DE DS Set goals at different levels
O5- Organizational goals
O6- System goals
O7 - User goals
O8 - Model goals
BO Establish success indicators
O9 - Leading indicators
BO DE DS Manage expectations
O10- ML trade-off [M5, M7, M11, M12, D5, I2, I10]
BO - Business Owner
DE - Domain Expert
DG - Designer
DS - Data Scientist
SE - Software/ML Engineer
User Experience (U)
BO DG Establish the value of predictions
U1 - Value
DG BO Define the interaction of predictions with users
U2- Forcefulness
U3 - Frequency
DG Visualize predictions
U4 - Visualization
BO DS Ensure the credibility of predictions
U6 - Acceptance [M5, M11, M13]
U7 - Accountability
U8 - Cost
U9 - User education & training [O10, M14]
SE DG Collect learning feedback from users
U5 - learning feedback [I3]
Data (D)
DS SE Access data
D1 - Source [I1]
D2 - Timeliness
DS DE Select and describe data
D3 - Data selection
D4 - Data dictionary
D5 - Quantity
DS DE Evaluate high-quality data
D6 - Accuracy
D7 - Completeness
D8 - Credibility
D9 - Consistency
D10 - Real usage
D11 - Bias
D12 - Ethics
D13 - Anonymization
DS DE Define a golden dataset
D16 - Golden dataset [M5]
DS Split dataset
D15 - Data distribution [M5]
DS DE Convert data in the representation of the ML model
D14 - Data operations & modeling
Model (M)
DS Select and configure the ML model
M1 - Algorithm & model selection [M2, M6, M8, D5]
M2 - Hyperparameter tuning [M4]
DS SE Evaluate other quality characteristics
M11 - Explainability & Interpretability [M13, I8, U7]
M12 - Scalability [D5, I1, I2, I10]
M13 - Bais & Fairness [D6:D12, D14]
M14 - Security & Privacy [D3, D13, I2, I4]
DS Train the ML model
M3 - Inputs & outputs [M11, D14, I1, I2, I3]
M4 - Learning time [I3, I10]
DS Deploy the ML model
M7 - Inference time [M12, I1, I2, I10]
M8 - Model size & complexity [M4, M7, M11]
M9 - Performance degradation [M10, I3]
M10 - Versioning [I2, I7]
DS DE Validate the ML model
M5 - Performance metrics [M2]
M6 - Baseline model
Infrastructure (I)
SE Transport data to the ML model
I1 - Data streaming [M7, M12, D14, I3]
SE DS Make the ML model available
I2 - Model serving [I10]
SE DS Update the ML model
I3 - Incremental learning [U6, D6:D12]
SE Store ML artifacts
I4 - Storage [M7, M12, I1, I7, I10]
SE Observe the ML model
I5 - Monitorability [M5, M14, D6:D12]
I6- Telemetry [U2, U3, U5]
DS SE Automate End-to-End ML workflow
I7 - Reproducibility
I8 - Maintainability
SE Integrate the ML model
I9 - Integration
BO SE Evaluate the financial cost involved with infrastructure
I10 - Cost

Figure 4.1: The most recent perspective-based ML task and concern diagram version. Once again, the red rectangles indicate tasks performed by data scientists, while the blue ones indicate those performed by software engineers. Relationships between concerns are indicated inside the brackets next to each concern's name

- If there was any interaction with a data scientist, software engineer, or other actors during task execution.
- Any challenges or improvement opportunities they perceived.
- If the task led to the creation of documentation.

Although the interview questions remained the same, using a different diagram allowed us to discuss new tasks. Since the improvements made on the diagram are based on feedback from ML practitioners who work on real-world projects, we believe these adjustments can also better capture the collaborative processes behind developing ML-enabled systems.

Once again, we recorded all interviews on the Google Meet platform and transcribed them using Google Cloud's Speech-to-Text API[1]. We presented examples when participants could not precisely understand our questions, and they also had the opportunity to address any unmentioned topics related to our research that they deemed relevant. After acquiring all the transcripts, we listened to the recordings to correct words and revise the sentences' punctuation. After that, we used open coding (Strauss and Corbin, 1998) to extract the most relevant statements based on our research questions. During this process, we coded citations illustrating either a responsibility participants had inside the team or a feature of their collaboration with other stakeholders.

[1]`https://cloud.google.com/speech-to-text`

The following sections describe each project we investigated and present the selected quotes.

4.3 SmartTocha

Our first group of participants are working on a project called SmartTocha. The project started in 2019 and is a few months away from being fully delivered to the customer. It was created to control burning gas quality in the flare systems of the customer's oil refineries. This is vital for minimizing environmental damage and saving resources inside the industrial unit. To do this, the system classifies the flame into one of three states: optimized flame, flame with too much vapor, or flame with soot. The water vapor flow is automatically decreased or increased based on this state. A more detailed description of the system is depicted in the patent application publication written by Kuramoto et al. (2023).

A camera is used to monitor the flames. Through the ML model, the system can quantify the flare flame height and classify it in one of the three states mentioned earlier. Beyond creating and training the model, the team also had to develop components for data processing and for integration with the software responsible for acting on the position of the water vapor valves. Furthermore, the team migrated system components to a cloud provider and created automated pipelines for continuous integration and deployment.

The team behind SmartTocha comprises ten members: a project manager, a scrum master, a UX designer, and seven data scientists. The data scientists are responsible for various tasks, such as image analysis and processing, model development, database administration, and building dashboards for product monitoring. They work with a team of software engineers appointed by the customer, with whom they share the responsibility for the system's infrastructure and component integration. Both teams frequently meet to discuss their activities, showcase results, and align future steps.

4.3.1 Participant Characterization

We interviewed two members working on SmartTocha, who both identified as male. One is a senior data scientist who now acts as project manager. He is responsible for supervising the work and researching academic papers that can help with development. The second participant is also a data scientist, mainly working with the ML model. The participants verbally agreed to participate voluntarily in the study and have their interviews recorded. We assigned each one an identification number, which is used in the following section while reporting the results. Table 4.2 shows their demographic data.

Table 4.2: Participants' Demographic Data

Participant ID	Role	Education Level	Years of Experience
PM1	Project Manager	Doctorate	14
DS1	Data Scientist	Master's degree	5

4.3.2 Results

To determine system definitions, the team used the Lean Inception methodology (Caroli, 2017). All members, including the technical team, participated in meetings to discuss the problem and the system's goals. DS1 said:

"*The whole team, including the data scientists, took part in the meetings. After that, we went on to develop features and start experimenting. I took part in this process to understand the problem we would solve. At first, I found it a bit difficult because it was the first time I had ever taken part in planning meetings in my career, but I overcame that.*"

PM1 explained that the team did not produce any formal requirements documentation after the Lean Inception. In his view, this was not a major problem due to the strong alignment between the team and customer representatives. However, he explained such documentation could have helped later in the project:

"*Towards the end of the project, the customers started to have a vision of the final product that differed from ours. Some project definitions became a little confusing, which led to much discussion about what would be delivered after the end of our contract. I think that having the requirements documented could have helped in this discussion by reminding us of what we had promised to do. It would at least be a starting point with what everyone converged initially.*"

According to PM1, domain experts from the customer company were responsible for data annotation. They also participated in discussions with the data scientists during data analysis and model results validation. One concern PM1 had was regarding the experts' availability:

> "*We depended on professionals who had daily activities inside the refineries, and it was often difficult for them to have time to help us. Furthermore, the annotation task requires time. We worked with different refineries and domain experts, some easier to access than others.*"

DS1, who worked with model training, explained how obtaining images with different characteristics was fundamental for the project's success:

> "*When we only monitored one refinery, our model performed very well. However, it did not work well when we started monitoring multiple refineries, since each one had different characteristics and camera angle positions. To solve this, we changed our initial training set and started using images from different refineries to assemble a generic model. We were careful to have a greater diversity in the set of images to make the model as generic as possible.*"

Team members from ExACTa and the customer's side worked together to develop the system. PM1 highlighted this interaction worked well and that he did not perceive any problems. DS1 explained how his team shared responsibilities:

> "*While some data scientists were responsible for developing the model algorithm, others researched academic papers to investigate new techniques we could use or details we were missing. The team regularly met [...] to present results and verify if they matched the papers' findings.*"

DS1 also illustrated how the model's performance degradation is handled:

> "*We are not responsible for constantly monitoring the model. When there is a problem with the model's response, someone from the refinery reports it to us. We also created dynamic dashboards to visualize the camera frames that were used to make a certain prediction. They are very helpful when analyzing these issues.*"

The architecture behind the model's integration with the refineries' services represented a collaboration point between DS1's team and the team responsible for the customer's cloud infrastructure. He exemplified how communication between the teams happened:

> "*We passed internal team definitions to customer representatives who met with us, and then these representatives would take what was defined to the infrastructure team. This indirect communication took time, and we had to plan our work with this in mind to prevent delays. The infrastructure team served all teams working with the cloud, so it was complicated for us to access them directly. In my opinion, this communication could have been better.*"

Establishing the model's infrastructure in the cloud fostered the creation of an ML workflow. DS1 illustrated how this took place:

> "*In the beginning, customer representatives had to collect videos from the cameras and manually send them to us so we could train the model. After migrating to the cloud, we could access the camera's IP address and fetch the videos in real time. I did not participate much in this activity, as I was mostly dedicated to data analysis and the ML model.*"

When talking about how the team managed the system's documentation, PM1 described why it was difficult to keep it updated:

> "*The system's initial documentation has evolved a lot. When there was a drastic change, we immediately updated the documentation accordingly. However, this was not the case when small changes happened over time. This can also happen while [...] experimenting with different algorithms and techniques, as during this process we are still unsure if the changes made will be definitive. To solve this, we had to plan some time to ensure our documentation was adequate.*"

4.4 Modeck

The second project we analyzed is called Modeck, which started in 2023. Its goal is to measure the extent of unused areas on decks of cargo ships. The team developed an ML model capable of analyzing images from cameras placed on different spots of the ship to calculate the unused area. This information is extremely important for companies who own cargo ships, as they can use it to optimize the amount of transported cargo and avoid the financial costs of underloading. In addition to the ML model, the team also developed a web application with both a front-end and a back-end. The front-end comprises an interface where users can register new cameras and delimit the area of the objects inside each captured image, which the model uses in its calculations. The system's back-end is responsible for storing all data related to the cameras and communicating with the ML model.

Modeck's team comprises a product owner, a UX designer, one data scientist, and two software engineers: one responsible for the front-end and the other for the back-end. The team follows the Agile software development methodology and gathers daily to understand what each member is doing and provide help when needed.

4.4.1 Participant Characterization

We interviewed the three team members who are developing the ML-enabled system: the data scientist responsible for the ML model and the two software engineers working on the back-end (SE1) and the front-end (SE2). They all participated voluntarily in our study and agreed to have their interviews recorded. Table 4.3 shows their demographic data. SE1 and DS1 identified as male, while SE2 identified as female.

Table 4.3: Participants' Demographic Data

Participant ID	**Role**	**Education Level**	**Years of Experience**
DS1	Data Scientist	Bachelor's degree	1
SE1	Software Engineer	Bachelor's degree	1
SE2	Software Engineer	Bachelor's degree	4

4.4.2 Results

All subjects joined the team after the start of the project and did not participate in the product's conception. For this reason, they all received a briefing from other members of ExACTa to comprehend their responsibilities and the project's scope. SE2 described how this process happened:

> "*There was no formal documentation. Instead, there was a meeting to showcase a presentation about the project. From there, I was able to understand its context.*"

She also explained how the lack of requirements documentation affected the team's tasks:

> "*Having a formal requirements documentation would have been beneficial. This is because sometimes requirements were not clear, which affected our planning. For instance, some tasks had to be paused because we were waiting for external data. If we had had precise requirements, we would have asked for the data earlier to prevent this delay.*"

DS1 is responsible for selecting and analyzing the data used to train the model. He mentioned several challenges encountered during this activity:

> "*It is very hard to obtain all the data we need. First, the operations on the deck never cease, and we do not have control over them. This prevents us, for example, from working with images of empty decks. Furthermore, we can not ask the deck operators to change the order of the objects on the deck to test different hypotheses for the algorithms. [...] We also had problems with cameras being moved to different places and images covered by pieces of cloth. We had to remove these images from the dataset manually.*"

The model already has an API that allows its consumption. DS1 is currently trying to deploy it on the customer's cloud environment. He mentioned other actors who are involved in this process:

> "*An infrastructure engineer from ExACTa and a software engineer from the customer's side are helping me with deployment. I also interact with software engineers who represent the customer when a problem needs to be solved. For example, when a field in the model's API is poorly documented.*"

When talking about deployment, DS1 also illustrated how the deployment documentation could be improved:

> "*Our deployment documentation could be better, especially to help the customer with this process. In my view, we need to improve how we describe the infrastructure around each component. For example, each API has error log queues used to register errors during processing, which are not clearly documented.*"

The team has yet to define how model integration with the back-end will take place, as mentioned by SE1:

> "*I am currently only developing the back-end, which was the first task I received after joining the team. I will plan this integration after the model is ready.*"

In the same way, the integration between the system's front-end and back-end has also not started. SE2 illustrated her activities while the back-end is being developed:

> "*We have not started integrating because I am currently waiting for the back-end APIs. In the meantime, I am using mocks to verify the web interface functionalities. As soon as we have a stable environment and the APIs ready, we can start the integration tests.*"

She also detailed her interaction with the team's UX designer so far:

> "*The designer works for multiple teams inside ExACTa. He occasionally participates in our project when we need to develop prototypes for the web interface or adjust them based on customer feedback. Besides that, we do not interact much.*"

On the other hand, SE1 explained how he initially worked together with DS1 to design the back-end:

> "*After analyzing our prototype for the web interface, I sat down with DS1 to define what information we would need to store and what would be displayed to the user. We did this for each web application page, and it became a part of the system's documentation.*"

DS1 acknowledged the team did not interact much outside the daily meetings and presented a reason for that:

> "*Each member is responsible for a different system component. We are not communicating as much at the moment, but I imagine it is because we are still at the beginning of the project and have not come close to any major deadlines.*"

4.5 Discussion

4.5.1 How do software engineers and data scientists share responsibilities when developing an ML-enabled system?

Investigating SmartTocha and Modeck revealed how the teams organized tasks throughout development. In both projects, all team members were introduced to the system's goals, as in the +Acordo project. In the case of Modeck, even though our participants were not part of the team when these definitions occurred, they still received instructions about the project's context and their roles inside the team as soon as they joined it. As we expected, using the most recent version of the perspective-based ML task and concern diagram to design our interview script allowed us to discuss tasks we had not explored during our case study. For example, one participant from SmartTocha talked about the process behind automating the ML workflow using tools from a cloud provider, which no participant from +Acordo had brought up.

In both projects, only the data scientists were responsible for analyzing data and building the model. Some also researched academic papers and created dashboards to help monitor the model's performance. Although the projects have different characteristics and objectives, the two teams faced difficulties obtaining sufficient data for all scenarios the ML model had to consider. SmartTocha's participants also mentioned domain experts aided them with data annotation and model evaluation. Unlike in the +Acordo project, domain experts' availability was another challenge reported during this phase.

On the other hand, software engineers handled the model's infrastructure and developed the components responsible for consuming it. Participants from both teams cited model deployment on the customer's cloud infrastructure as a collaboration point between data scientists and software engineers. In both projects, at least one software engineer was from a separate team responsible for managing the customer's internal systems.

Participants agreed that all team members are responsible for maintaining project documentation. They pointed out that developing clear requirements documentation could have helped better align expectations between the team and the customer. In addition, having requirements precisely defined would have enhanced the team's planning. While discussing system documentation, both teams stated they struggled with updating it during development, which is another difficulty debated in our case study. Participants also explained how some components were not fully documented, which resulted in the team dedicating a separate time to guarantee adequate documentation. This lack of updated documentation harmed collaboration during model deployment.

4.5.2 How do software engineers and data scientists collaborate when developing an ML-enabled system?

Our interviews portrayed how collaboration between these two roles occurred in each project. The team working on SmartTocha does not comprise any software engineers, so they had to collaborate with a team of software engineers who worked for the customer and were responsible for their system's infrastructure. The teams had a lot of interactions, especially during model deployment. However, they could not always communicate directly due to the unavailability of the customer's team. For this reason, communication issues had to be planned to prevent delays in the project's progress. This evidences a challenge related to collaboration between actors from different organizations, a topic we had not analyzed in our previous study.

Modeck's team, on the other hand, comprised a data scientist and two software engineers. The data scientist interacted with the software engineer responsible for the back-end to diagram how data would be stored. After that, they communicated only during daily meetings or when one of the team members needed help. This reduced frequency of interactions can be explained by the fact that the team has not yet started integrating system components. Since the project is still in its early stages, each team member is currently developing the component for which they are responsible separately. However, we could notice interactions between the team's data scientist and the software engineers responsible for managing the customer's infrastructure. This collaboration was once again important for model deployment and when problems in the system appeared. To avoid these problems, both actors worked together to improve the model's API documentation.

4.5.3 Implications for Practitioners

Based on the findings of our case study and complementary interviews, we proposed recommendations to improve collaboration between software engineers and data scientists. We seek to aid teams developing ML-enabled systems to avoid the pitfalls we have depicted.

One of the key challenges that software engineers and data scientists face when interacting and collaborating on ML-enabled systems is the lack of clear requirements specifications. Without well-defined requirements, it can be difficult for these actors to understand each other's needs and expectations, leading to miscommunication and inefficiencies in the development process. This highlights the importance of establishing and maintaining clear requirements specifications that can serve as a shared understanding between software engineers and data scientists, enabling them to work together more efficiently.

Fostering a collaborative culture from the start of the project is fundamental. We believe this can be achieved by establishing a comprehensive planning of the system that involves all project members. While planning, the responsibilities of each actor must be clear to everyone on the team. Moreover, actors should be comfortable with the tasks they will perform or at least be willing to learn how to execute them. If any dependencies between the actors require their interaction, these should be mapped in advance to prevent delays during development.

Despite their background and cultural differences, software engineers and data scientists should avoid working isolated from one another. Even though some tasks can be executed independently, they must communicate frequently. Teams should also encourage knowledge exchange between them, which can be done by pairing a member from each role to work on a task together. Another possibility is to have members of a role present their work to the rest of the team so that other actors can become familiar with their activities.

ML-enabled system architecture and definitions documentation can also enhance the interaction between these two actors. These documents should provide a concise and unambiguous description of what the ML-enabled system and each of its components should do. This facilitates discussion between team members, who can use this documentation as a reference, preventing misconceptions. Our results illustrate that such documentation can also be useful while onboarding new team members.

4.5.4 Threats to Validity

This section discusses threats to the validity of this study. We decided to focus on the same threats portrayed in the case study: construct validity, internal validity, external validity, and reliability.

A threat to **construct validity** consists of participants providing an inaccurate description of their activities inside the team, which would represent a flaw in our research methodology. To mitigate this threat, we made an effort to validate what the participants expressed during the interviews. We obtained a patent application for the SmartTocha project (Kuramoto et al., 2023), which documents how the developed system is supposed to operate. The document is in accordance with the description provided by the participants. Furthermore, the advisor of this work is associated with ExACTa and verified the integrity of the team dynamics reported by the subjects.

Threats to **internal validity** comprise methodological errors such as asking unclear questions to participants. For this reason, we used the work of Villamizar et al. (2024) during the design of our interview script to accurately investigate the tasks carried out during the development of ML-enabled systems. When participants mentioned doubts about a question or activity we referred to, we presented examples for clarification. We recognize the number of subjects we interviewed from each team is not high, as not all team members were available to participate in our research. Nevertheless, we could still obtain perspectives on their activities and how team members collaborated.

To mitigate threats to **external validity**, such as our results not being representative of other ML projects, we investigated two separate teams. Even though both teams worked on ML-enabled systems, they had their own goals, composition, and collaboration procedures. We could identify several similarities between the teams within their challenges and the actors with whom they collaborated. We also noted resemblances with the team examined in our case study, which evidences the generalizability of results.

Finally, to improve **reliability**, we have detailed our research methodology and the questions asked during the interviews. This enables the replication of our study, which we encourage other researchers to perform.

4.6 Concluding Remarks

This chapter discussed two complementary interviews we conducted with data scientists and software engineers of two distinct teams building ML-enabled systems for the industry. We designed our interview script based on the most recent version of PerSpecML (Villamizar et al., 2024). After interviewing participants, we reviewed the transcripts and extracted the most relevant statements for answering our research questions.

The results greatly complemented our case study's findings. We perceived similarities in how data scientists and software engineers share responsibilities inside the project and in the tasks that require collaboration between them and other team members. Furthermore, we noticed resemblances in the challenges faced, such as the difficulty of updating the system's documentation during development. The findings also allowed us to uncover new challenges, such as domain experts and software engineers lacking the availability to interact with data scientists. Based on these and other obstacles, we proposed recommendations for teams developing ML-enabled systems. They aim to reduce the gap between software engineers and data scientists and clarify team definitions for all actors involved. We believe collaboration between the actors can be enhanced by following these recommendations.

5
Focus Group

5.1
Introduction

The results depicted in the previous chapters allowed us to visualize how the interaction between software engineers and data scientists develops in real-world projects. We mapped each actor's responsibilities and several activities that require their collaboration. Investigating how these actors work together also uncovered improvement opportunities for their interaction.

As a final research step, we discussed the findings of our case study and complementary interviews with experienced practitioners during focus group sessions. Adopting this research method enabled practitioners to assess the interaction points between software engineers and data scientists we discussed in the previous chapters. Furthermore, it allowed us to evaluate recommendations that may enhance this collaboration.

Focus groups are a qualitative research method where data is collected through group discussions on a given topic. For this reason, focus groups must be carefully planned so that the discussion stays focused and does not exceed the specified time slot. As explained by Kontio et al. (2008), this method can provide truthful and insightful information, as it depicts the perception of participants who possess knowledge of the discussed topic. It has also been previously used in other software engineering studies for acquiring developers' perspectives (Almeida et al., 2021). For these reasons, we considered conducting focus groups a suitable choice.

Once again, we used the GQM template for goal definition by Basili and Rombach (1988) to define our study's goal, presented in Table 5.1. Based on this research goal, we established two research questions.

Table 5.1: Focus Group Goal

Analyze	the interaction between software engineers and data scientists
for the purpose of	characterization
with respect to	tasks in which they collaborate and recommendations for improving their interaction
from the point of view of	experienced software engineers and data scientists
in the context of	ML-enabled system projects for different industrial customers.

RQ1: What is the perception of software engineers and data scientists on which tasks they most collaborate?

This research question focuses on how software engineers and data scientists evaluate their collaboration while developing ML-enabled systems. It aims to understand how vital they recognize this interaction for several tasks they must perform to build the system. To answer this question, we analyzed our previous findings and selected tasks in which collaboration could be useful for participants to discuss.

RQ2: What is the perception of software engineers and data scientists on how to improve collaboration between them?

This question explores how to enhance the collaboration between software engineers and data scientists working on ML-enabled systems. It aims to uncover valuable recommendations they can use inside their respective projects to improve productivity and team communication. To answer *RQ2*, we elected recommendations based on the literature and our previous studies and asked participants to evaluate their relevance for this collaboration.

This chapter comprises five sections. Section 5.2 covers how we designed the focus group. Section 5.3 reports the participants' demographic data and their statements during the focus groups. Section 5.4 presents the findings we obtained for each research question. Section 5.5 contains our concluding observations.

5.2 Focus Group Design

We designed our study following the guidelines proposed by Kontio et al. (2008) for software engineering empirical research using focus groups. We organized the conception of our focus group into three steps, portrayed in Figure 5.1: Focus Group Planning, Participant Recruitment and Characterization, and Data Collection. The following sections detail each of these steps.

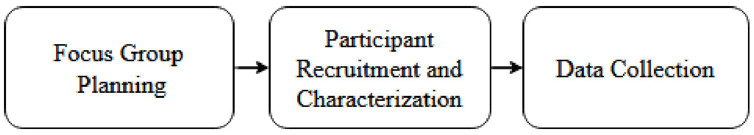

Figure 5.1: Focus Group Design Steps

5.2.1 Focus Group Planning

Our first step towards designing the focus group was to define how it would be conducted. We had to specify what topics the participants would discuss and what would be the best way to foster this discussion. To do this, we used our research questions as a guide. We divided our focus group session into two stages: one to debate tasks related to collaboration between software engineers and data scientists, and one to examine recommendations for improving this collaboration.

To design the first stage, we defined the tasks that would be discussed based on the results of our case study and complementary interviews. We extracted several collaboration points between software engineers and data scientists mentioned in our previous studies and categorized them based on the tasks defined by Villamizar et al. (2024) in the *PerSpecML* approach. This resulted in a total of seven tasks, detailed below:

- *Data Access Definition*: This task concerns defining how the data required for developing and evaluating the ML model will be acquired. This process may represent a collaboration point if the actors interact to collect the data, which we noticed during the case study.
- *Data Selection*: This task involves selecting the data used to build the model and describing what each data component represents. In our previous studies, software engineers reported they did not know the meaning of the data fields used by their respective ML models, which led to problems during system development. For this reason, we decided to investigate if collaboration could be helpful in this context.
- *ML Model Evaluation*: This task addresses evaluating the ML model. Besides measuring its performance through metrics such as accuracy and precision, this task also considers factors like the model's interpretability. Since we had reports of team members lacking knowledge about the

model, treating it as a "black box," we agreed to analyze if collaboration would be considered relevant during this task.

– *ML Artifact Storage*: This task concerns systematically storing and managing all ML artifacts, such as scripts and the model itself. Since our previous studies showed that ML artifacts might be incorrectly stored if data scientists and software engineers do not communicate effectively, we decided to investigate how collaboration between these actors can improve ML artifact storage.

– *ML Model Availability*: This task involves exposing the ML model so other system components can consume it. Our previous works revealed this is an interesting task for exploring the effects of collaboration. For example, one of the case study's participants expressed dissatisfaction with the data scientists being solely responsible for this task. They did not possess the skills required to perform it, so they had to ask the software engineers for help.

– *ML Model Integration*: This task relates to incorporating the ML component into the larger software system. We analyzed several collaboration-related challenges during this task, such as data being in the wrong format during integration tests, communication issues, and component integration not being properly documented.

– *ML Model Deployment*: Finally, this task concerns making the ML model available in a production environment. Multiple data scientists we interviewed depicted difficulties handling model infrastructure issues, which led to software engineers helping them with this task.

The second stage of our focus group aimed to promote the debate of recommendations for collaboration between data scientists and software engineers. To determine the recommendations that would be presented to the participants, we analyzed improvement opportunities portrayed in our previous studies and the current literature on collaboration for ML-enabled systems. Even though there are similarities between these studies' findings, combining them allowed us to obtain a comprehensive set of recommendations. Furthermore, assessing recommendations depicted in past works together with practitioners is currently a gap in the literature.

To contemplate the results of our case study and complementary interviews, we revisited the implications for practitioners discussed in Section 4.5.3 of Chapter 4. We believe they can be useful for teams developing ML-enabled systems, and their debate within the focus groups would allow us to evaluate their relevance. While analyzing the existing literature, we selected two papers to focus on: Nahar et al. (2022) and Mailach and Siegmund (2023).

These two papers provide explicit proposals for improving collaboration in the context of ML-enabled systems and are consistent with the outcomes of our previous studies. Many of the challenges discussed by Nahar et al. (2022) were reported in our works, such as data scientists working isolated from software engineers, insufficient system documentation, and problems with responsibility sharing. Furthermore, our studies portray the three collaboration points the authors presented in their paper. Our findings even reported a new collaboration point, where software engineers developed a system to be used by domain experts to help in data annotation for model training. We perceived several similarities with Mailach and Siegmund (2023)'s paper as well. For instance, they also reported how software engineers may lack knowledge about the model and view it as a black box. Another challenge they discussed was a disconnection between the development team and other project stakeholders. This was evident in our analyses, especially during requirement definitions for the model.

After examining all studies, we extracted the recommendations we judged most relevant for discussion and assembled our final list, described below. We assigned each recommendation a number, which we will use to reference them in the next sections.

1. *Involve data scientists and business owners when eliciting and analyzing requirements*: This recommendation concerns having both data scientists and business owners together with software engineers during requirements definition. Fostering their interaction during this project stage may help clarify requirements and discard unrealistic ones.

2. *Provide ML literacy for all project stakeholders*: This recommendation aims to establish a shared understanding of basic ML concepts and terminologies, such as accuracy and model training, across all team members. This can be done through lectures or workshops and may facilitate communication inside the team.

3. *Develop documentation for product requirements, system architecture, and APIs at collaboration points*: The importance of documentation was mentioned in all studies we reviewed. We grouped three different project documents within this recommendation to evaluate how they can affect collaboration.

4. *Define clear responsibilities and internal processes with clear boundaries for data scientists and software engineers*: This recommendation empha-

sizes that both data scientists and software engineers must know exactly what tasks they are expected to perform inside the team. They should also be able to execute the tasks, or at least be willing to learn how to do so.

5. *Support interdisciplinarity between data scientists and software engineers*: This recommendation encourages knowledge exchange between data scientists and software engineers, where one actor learns how the other performs their activities. This can be done, for example, by pairing both actors to work together on the same task or through workshops.

6. *Organize regular meetings for showcasing team activities*: Finally, this recommendation seeks to improve collaboration through meetings with the whole team. During these meetings, members can present their work and what they have learned from their activities. Meetings can also be useful to synchronize tasks and broadcast messages to all team members.

Our focus group participants discussed each of the tasks and recommendations above. The strategy used to collect their perspectives is described in Section 5.2.3.

5.2.2 Participant Recruitment and Characterization

After defining what topics would be discussed during the focus group, we started recruiting participants for our research. We established two requirements for selecting the participants. First, they needed to be professionals with experience either in data science or software engineering. Moreover, they needed to have worked on at least one ML-enabled system project. With these requirements, we could ensure participants would have enough experience to contribute to the focus group discussions. At the same time, recruiting experienced participants enhances the credibility of our results.

We contacted fourteen Tecgraf and ExACTa professionals working on distinct industry-academia collaboration projects developing ML-enabled systems. We had easy access to them since they are affiliated with PUC-Rio and because some of them had also participated in our previous studies. From the fourteen emails sent, we received seven positive responses. We used the Doodle[1] platform to allow participants to choose the best dates for the focus group sessions.

Each confirmed participant also received a consent and characterization form. In the consent form, we formally stated the goal of the focus group and

[1]`https://doodle.com/en/`

its estimated duration time. We also clarified that all collected data would be anonymized, treated as confidential, and used only for research purposes. The goal of the characterization form was to gather demographic data about the participants. This allowed us to check if our requirements for participant selection had been met. In the form, we asked short and direct questions to prevent harming the respondents' engagement, which led to the estimated time for filling out the form being less than two minutes.

We asked participants about their education levels, years of work experience, and the number of projects they worked on with ML components. We also asked which role they identified with, either as a data scientist or a software engineer, and their proficiency in both areas. To measure proficiency, we used the NIH Proficiency Scale[2], a widely used instrument to evaluate competency in a particular field. Individuals can use this instrument to compare their level of proficiency to other workers. The scale comprises six levels of competency, depicted in Table 5.2. We included each level and its description in the form so that participants could choose the most appropriate one.

Table 5.2: Levels of the NIH Proficiency Scale

Proficiency Level	**Description**
Not Applicable	The respondent is not required to demonstrate this competency on the job and does not have any knowledge.
Fundamental Awareness	The respondent understands basic concepts and techniques.
Novice	The respondent can perform tasks with the help of others.
Intermediate	The respondent can usually perform tasks independently, only needing help from time to time.
Advanced	The respondent can perform tasks without assistance and is capable of coaching others.
Expert	The respondent is recognized as an authority in this competency and is capable of assuming a leadership role.

5.2.3 Data Collection

The focus group sessions were conducted online using Google Meet. We decided to organize two sessions based on the number of confirmed participants. Scheduling two distinct sessions with participants who work on different projects diversified the shared experiences. To present the topics for discussion and enable exchange between participants, we created interactive boards inside Miro[3].

[2]`https://hr.nih.gov/working-nih/competencies/competencies-proficiency-scale`

[3]`https://miro.com/pt/`

Figure 5.2 illustrates the upper part of the board created for the first stage of the focus group. The board contains all tasks depicted in Section 5.2.1. For each one, participants were asked to evaluate how relevant they considered the interaction between data scientists and software engineers for that task. They could either agree, partially agree, partially disagree, disagree, or not express an opinion toward the statement. Post-its with the participants' names were placed next to each task on the board. To vote, participants had to drag their corresponding post-its to the desired answer.

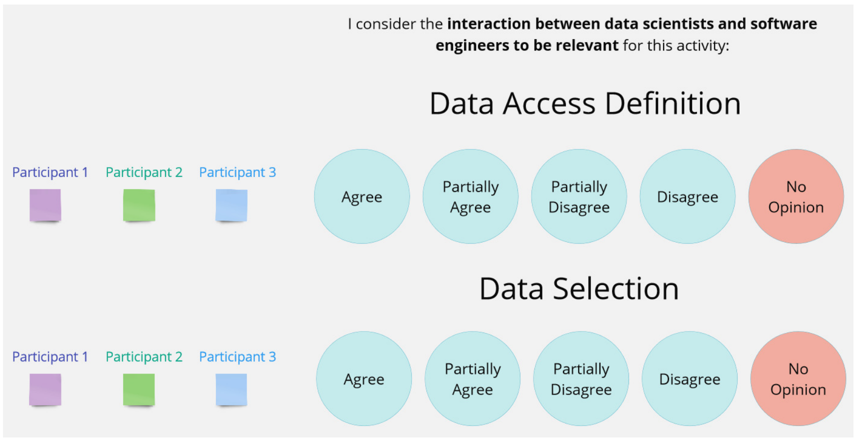

Figure 5.2: Upper Part of the Board for the First Stage of the Focus Group

Two authors participated as moderators in both sessions. In the first stage of the focus group, they described the research goal and read out loud each of the tasks that would be examined. Participants were allowed to ask questions about the tasks if they had doubts, which were answered through examples. After that, they had one minute to cast votes for all tasks. When voting was finished, participants had seven minutes to discuss the thought process behind their answers. During this discussion, they could also change their votes in case they felt persuaded by other participants. Relevant comments made by the participants were registered on the board by one of the moderators. The moderators also asked clarifying questions to the participants when there was a need to better understand the statements made or the context behind the answers.

After the end of the discussion, the moderators explained the second stage of the focus group. In this stage, participants had to examine recommendations for the collaboration between data scientists and software engineers. We created six Miro boards for this stage, one for each recommendation. Figure 5.3 depicts the upper part of the board for the first recommendation. The recommendation is presented at the top, and participants had to assess how relevant they considered it to be for the interaction between data scientists and software engineers. They were asked to do this for each task evaluated in the first stage, which allowed us to grasp the relevance of the recommendations for each task. We used the same voting options employed in the previous stage.

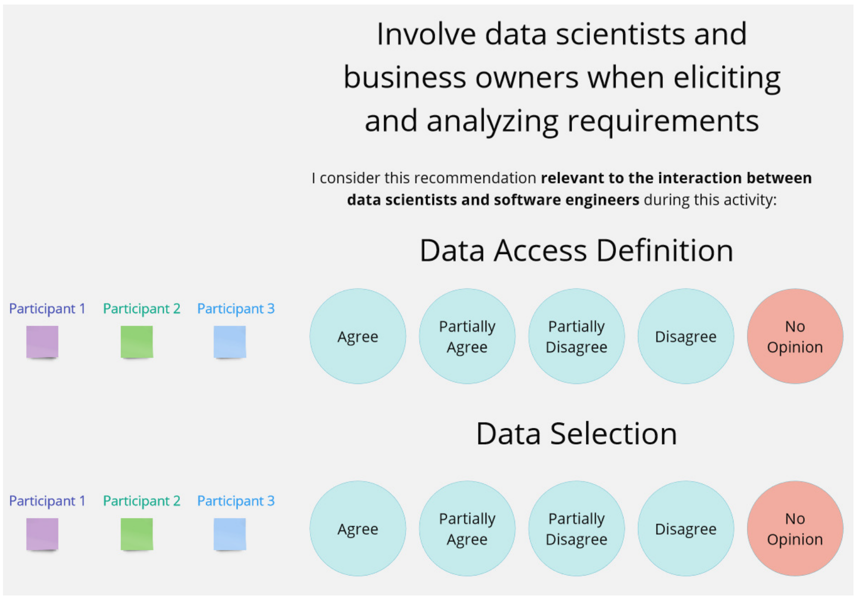

Figure 5.3: Upper Part of the Board for the First Recommendation

One of the moderators explained each recommendation and answered questions in case participants had any. Then, participants had one minute to cast their votes for all tasks. After the last participant's vote, the group had seven minutes to justify their decisions and discuss the relevance of that recommendation for collaboration. Once again, the moderators intervened whenever a comment needed more context to be properly understood. When the discussion was over, the moderators advanced to the next recommendation until all recommendations had been examined.

Both focus group sessions lasted one hour and fifteen minutes. All of them were recorded and subsequently transcribed with Google Cloud's Speech-to-Text API[4]. During data analysis, we first translated the transcripts from Portuguese to English and corrected them while listening to the recordings. The revised transcripts are depicted in Appendix C. We then analyzed the content of the interviews using open coding (Strauss and Corbin, 1998). For each focus group, we looked for statements depicting previous experiences with collaboration shared by the participants, as well as positive and negative features of the recommendations. We also compared the generated codes with the notes registered by the moderators during the sessions to verify if all important topics had been covered. Assigning codes to these statements allowed us to group

[4]`https://cloud.google.com/speech-to-text`

similar perspectives regarding our discussed topics. Moreover, it enabled us to contrast perceptions from participants of different focus group sessions.

5.3 Focus Group Results

We assigned identification numbers to the participants to preserve their anonymity. Table 5.3 displays their demographic data. Participants DS1, DS2, DS3, DS5, SE1, and SE2 identified as male, while DS4 identified as female. Since only two participants identified as software engineers, we allocated them to different focus group sessions, as depicted in Table 5.4. The first session comprised DS1, DS2, and SE1, while the second comprised DS3, DS4, DS5, and SE2.

Table 5.3: Participants' Demographic Data

Participant ID	Role	Education Level	Years of Experience	Number of Projects with ML
DS1	Data Scientist	Doctorate	10	10
DS2	Data Scientist	Doctorate	5	6
DS3	Data Scientist	Doctorate	21	11
DS4	Data Scientist	Master's degree	6	2
DS5	Data Scientist	Doctorate	7	5
SE1	Software Engineer	Master's degree	20	2
SE2	Software Engineer	Master's degree	15	1

Table 5.4: Allocation per Focus Group Session

Participant ID	Focus Group Session
DS1	1
DS2	1
DS3	2
DS4	2
DS5	2
SE1	1
SE2	2

Table 5.5 portrays the participants' responses for their data science and software engineering proficiency levels. As we expected, all participants declared having at least an intermediate level of proficiency in their respective fields, and some participants even reported proficiency in both areas. Furthermore, Table 5.3 shows that all participants have five or more years of work experience and have worked on at least one project with an ML component. These characteristics make them suitable for participating in our study.

In the following subsections, we display the participants' votes and summarize their comments during the sessions. Since the "No Opinion" option was never chosen, we decided to omit it from the tables showcasing the results.

Table 5.5: Participants' Proficiency Data

Participant ID	Data Science Proficiency	Software Engineering Proficiency
DS1	Advanced	Advanced
DS2	Advanced	Novice
DS3	Expert	Advanced
DS4	Intermediate	Not Applicable
DS5	Intermediate	Novice
SE1	Not Applicable	Expert
SE2	Not Applicable	Expert

5.3.1 Relevance of Collaboration for each Task

Table 5.6 depicts how our participants evaluated the relevance of collaboration between software engineers and data scientists for each task.

Table 5.6: Results for the Relevance of Collaboration for each Task

Task	Participants who Agreed	Participants who Partially Agreed	Participants who Partially Disagreed	Participants who Disagreed
Data Access Definition	DS1, DS2, DS3, DS4, DS5	SE1, SE2	-	-
Data Selection	-	-	DS1, DS2, DS3, DS4, DS5	SE1, SE2
ML Model Evaluation	-	-	DS3, DS4, SE2	DS1, SE1, DS2, DS5
ML Artifact Storage	DS1, SE1, DS4, DS5	DS2, DS3	SE2	-
ML Model Availability	DS1, SE1, DS2	DS3, DS4, SE2, DS5	-	-
ML Model Integration	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-
ML Model Deployment	DS1, SE1, DS2, DS3, DS4, DS5	SE2	-	-

All participants either agreed or partially agreed that collaboration during the definition of data access is important. DS1 explained his point of view:

> "*Software engineers usually have a greater knowledge of the company's APIs for capturing data, so they should be able to help the data scientists navigate the available infrastructure better. However, this also depends on the role of the data scientist. For example, there are teams where the data scientist is almost a database administrator as well. In this case, a software engineer may not be needed.*"

DS2 reinforced the importance of a software engineer during this task:

> “*Considering that data may not be in the hands of the data scientist, a software engineer can help understand the best way to access it, when to access it, etc.*”

On the other hand, SE2 explained that an interaction with a software engineer may not always happen:

> “*In the project I am currently working on, we use a collection of documents to train the ML model. These documents were sent to us by customer representatives who are not software engineers, so there was no interaction between these roles. This is why I only partially agreed with the statement for this task.*”

Some participants tended to disagree with the statement when discussing data selection and model evaluation. One of them was SE1, who explained his perception:

> “*In my previous work experiences, the data scientists were responsible for evaluating the model and selecting data. In my current project, I have no idea how our ML model was evaluated or how data was selected.[...] Still, since we have a good relationship with the data scientists, we are always willing to help them if they need it.*”

DS2 agreed with SE1’s opinion, explaining how software engineers may not be interested in participating in these tasks:

> “*I noticed software engineers are usually not very interested in the research process of a data science application. I have made presentations showcasing the algorithms examined to build a model, or the metrics used for its evaluation, and their attention goes away very quickly.*”

DS3 explained that selecting data is more suited to the data scientist’s role:

> “*The role of a data scientist consists of analyzing the data and performing feature engineering to train the model. For these activities, I would not involve a software engineer. On the other hand, during data selection, it is possible to deal with incomplete data that you might have to discard or adjust. You may even discover the existence of more data that you still need to acquire. This can lead to a change in how data is being accessed, which may provoke an interaction with a software engineer.*“

DS1 exemplified how an interaction with a software engineer would not necessarily be useful during model evaluation:

> "*[...] The data scientists know what metrics are relevant. They will know, for example, if the model is overfitted. If you ask software engineers to deploy an overfitted model, they will probably do it, but only a data scientist will realize that the model has a problem and is not ready for production. Hence, I do not know how a software engineer could help in this process.*"

Most participants agreed that collaboration would be important for storing ML model artifacts. DS5 justified his vote:

> "*It is important that both actors define where and how this storage will occur. This interaction with the software engineers allows the data scientists to understand what infrastructure is currently used for storage, as they are usually not involved in this process.*"

SE1 highlighted this interaction is important due to the complexity of the ML component:

> "*Planning model storage is a task which software engineers should participate. The model can have multiple artifacts, some of which may be enormous, so they must be stored accordingly.*"

Finally, DS3 pointed out another advantage brought by collaboration when team members change:

> "*Throughout development, it is possible that team members change, which may harm the improvement of existing models. This can be mitigated by having an infrastructure where artifacts can be properly managed with adequate versioning, which should be created through the actors' collaboration.*"

Making the ML model available and integrating it with the rest of the system did not receive any disagreement votes. DS2 summarized the participants' opinions:

> "*These tasks will define the ML model's output for other system components. This should be discussed between the software engineers and the data scientists, who must evaluate this according to the system's goals.*"

Model deployment was another activity where collaboration was viewed as important. DS1 illustrated how that interaction should work:

"The actors in charge of CI/CD operations are usually software engineers. However, this does not mean they know how to execute the model. For this reason, it is vital that a data scientist gets together with them to explain the model and how to run it. [...] I believe both actors need to be in sync to avoid any problems."

5.3.2 Involve data scientists and business owners when eliciting and analyzing requirements

The first recommendation we discussed was **involving data scientists and business owners when eliciting and analyzing requirements**. Table 5.7 depicts how our participants evaluated the relevance of this recommendation for the collaboration between software engineers and data scientists for each task.

Table 5.7: Agreement on the First Recommendation

Task	Participants who Agreed	Participants who Partially Agreed	Participants who Partially Disagreed	Participants who Disagreed
Data Access Definition	DS1, SE1, DS2, DS3, DS4, DS5	SE2	-	-
Data Selection	DS1, SE1, DS2	-	DS3, DS4, DS5	SE2
ML Model Evaluation	DS1, DS2	SE1, DS3, DS4, DS5	-	SE2
ML Artifact Storage	-	DS1, DS3, DS5	SE2	SE1, DS2, DS4
ML Model Availability	-	DS1	DS3, DS4, SE2, DS5	SE1, DS2
ML Model Integration	-	DS1, DS3, DS5	DS4, SE2	SE1, DS2
ML Model Deployment	-	DS1, DS3, DS5	SE2	SE1, DS2, DS4

Participants agreed that this recommendation is important during data access definition. DS2 explained why based on his previous experiences:

"Involving the business owners can help the actors understand what the data represents. Sometimes, you know the name of a given variable and whether it is numeric or categorical, but you may not understand what it represents. [...] I have worked on projects where the data came from another company. In these cases, the business owners had to explain what each data field represented, and this improved the interaction between us and the software engineers."

SE1 agreed with DS2 due to the knowledge that business owners possess regarding the data:

> “*The business owner knows the data very well. A business owner usually knows how to access the data, whether it is from other systems or from a spreadsheet. I think this interaction is important, especially when defining how data will be acquired and selected.*”

DS3, on the other hand, did not consider this recommendation relevant during data selection:

> “*I do not think the interaction between data scientists and software engineers is important during this task. The presence of other actors, such as business owners, is more important than this interaction.*”

DS3 also defended this recommendation for ML model evaluation:

> “*Everyone must comprehend what exactly will be evaluated. Performance can be evaluated not only in terms of accuracy and other metrics but also in terms of computational performance. There is no point in having a super complex model if it will require a machine with tons of computational power that will not be available. These definitions can sometimes be made together with business owners and software engineers.*”

DS1 also agreed with the importance of business owners during model evaluation:

> “*Business owners understand a lot about the data and can help with model evaluation. For instance, sometimes you may think the model has to avoid false positives, but they might say, ’No, my problem is with false negatives,’ so then you will have to choose another metric.*”

DS1 mentioned this recommendation would be useful when dealing with latency requirements:

> “*Having the business owners close to the software engineers and data scientists during requirements analysis can help enhance their collaboration. For instance, imagine a scenario where the business owner asks for a given accuracy and latency. The data scientist knows how to achieve that accuracy, but may not know if that latency is possible given the available machines. This is something a software engineer can help with.*”

The participants had different opinions regarding ML model integration and deployment. They either partially agreed, partially disagreed, or disagreed entirely. DS4 did not consider this recommendation relevant for these tasks:

> "*I do not think this recommendation would be that interesting for these tasks. [...] I consider model integration more of an implementation task, just like with deployment, so perhaps the business owners might not be needed.*"

DS3, on the other hand, partially agreed with this recommendation's relevance for these tasks:

> "*In my view, any prioritization or decision-making often involves business owners. In some cases, the integration task may involve different teams, and the business owner will be able to facilitate this coordination.*"

5.3.3 Provide ML literacy for all project stakeholders

The second recommendation we discussed was **providing ML literacy for all project stakeholders**. Table 5.8 shows how our participants evaluated the relevance of this recommendation.

Table 5.8: Agreement on the Second Recommendation

Task	**Participants who Agreed**	**Participants who Partially Agreed**	**Participants who Partially Disagreed**	**Participants who Disagreed**
Data Access Definition	DS5	DS3, DS4, SE2	-	DS1, SE1, DS2
Data Selection	-	-	DS1, DS3, DS4, SE2, DS5	SE1, DS2
ML Model Evaluation	-	DS3, DS4, SE2, DS5	-	DS1, SE1, DS2
ML Artifact Storage	DS2	DS1, DS3	SE1, SE2, DS5	DS4
ML Model Availability	DS2	DS1, DS3, SE2, DS5	SE1, DS4	-
ML Model Integration	DS4, SE2, DS5	DS1, DS2, DS3	SE1	-
ML Model Deployment	-	DS1, DS3, DS4, SE2, DS5	SE1, DS2	-

DS1 did not see this recommendation relevant while defining data access:

> "*During data access definition, I do not think knowing the difference between classification and regression would be helpful for acquiring the data. [...] However, if software engineers happen to know a bit more about data science, they may be able to help with data selection. For example, they may discover noise in the data capable of hindering model training, or notice that a data column has many null values.*"

SE1 explained how his work experience influenced his view on the effect of this recommendation during data access definition:

> "*I have been working in an academic environment for a long time. Even though I am not in the AI field, I constantly see lectures and learn about this topic, so I did not require this literacy. Having said that, I do not think this theoretical knowledge is important for data access definition. I think practical instructions, such as how to access a spreadsheet or another system, are more efficient.*"

DS3 assessed this recommendation positively:

> "*I consider literacy relevant, as knowing at least the basics is important for communication. The only exception I can see is during data selection because I think the participation of the software engineer is reduced. I partially agreed on the other tasks because, in the worst case, everyone has to know that a model will be executed, that an output of a certain type will be generated, etc.*"

SE2 also highlighted how this recommendation enhances communication:

> "*This recommendation is important to improve communication between the software engineer and the data scientist. It is vital to establish a common terminology so that communication flows more easily.*"

DS3 mentioned this recommendation could be valuable during the ML artifact storage task:

> "*As I mentioned in the discussion about the tasks, it is a good idea to define the best way to store the model together with the software engineers to make it more easily accessible in the future. [...] They must understand what model training is and what is actually being stored.*"

DS1 exemplified the advantage depicted by DS3:

> "*Comprehending how the models are generated allows you to think about how to persist them more intelligently. For example, suppose there is a high chance of data drifts in the project you are working on. In that case, the software engineer helping with ML artifact storage would know that the model needs to be trained and updated often. This would determine the development of an effective versioning system.*"

When debating ML model deployment and integration, DS1 also considered this recommendation relevant:

> *"ML literacy for software engineers greatly helps in these tasks. They will know, for example, what type of approach to adopt when deploying the model. Depending on the type of model the data scientist has created, the software engineer will have an idea of the computational power required to run it. [...] The software engineer will be able to notice this even if the data scientist forgets to warn the team. This avoids problems in the future, such as having a deployed model with very high latency."*

DS2 emphasized the benefits of this recommendation during model integration:

> *"ML literacy can help with the definition of the model's output and how it will be consumed. It makes communication between the actors easier when specifying the best way to interact with the model."*

5.3.4 Develop documentation for product requirements, system architecture, and APIs at collaboration points

The third recommendation we discussed was **developing documentation for product requirements, system architecture, and APIs at collaboration points**. Table 5.9 presents how our participants evaluated the relevance of this recommendation.

Table 5.9: Agreement on the Third Recommendation

Task	Participants who Agreed	Participants who Partially Agreed	Participants who Partially Disagreed	Participants who Disagreed
Data Access Definition	DS3, DS4, SE2, DS5	-	DS1, SE1, DS2	-
Data Selection	-	DS1, SE1, DS2	DS3, DS4, DS5	SE2
ML Model Evaluation	DS4	DS3, SE2, DS5	-	DS1, SE1, DS2
ML Artifact Storage	DS4	DS1, DS3, SE2, DS5	-	SE1, DS2
ML Model Availability	SE1, DS2, DS4, DS5	DS1, DS3, SE2	-	-
ML Model Integration	SE1, DS2, DS3, DS4, SE2, DS5	DS1	-	-
ML Model Deployment	SE1, DS2, DS3, DS5	DS1, DS4, SE2	-	-

Some participants did not consider this recommendation relevant for data selection, such as DS3:

> "*I do not think this recommendation is relevant for data selection because I do not see the need for much interaction between software engineers and data scientists during this task. For ML model evaluation, I think this interaction exists, but it is not strong, so I partially agreed with the statement.*"

DS4 shared a different opinion regarding model evaluation:

> "*The model will be evaluated based on the documented requirements, so I think this recommendation has a lot of influence on this process. Even if you do not need a lot of interaction between the software engineer and the data scientist in this activity, I believe documentation will be relevant for any interaction that happens.*"

All participants either agreed or partially agreed with the statement for model availability, model integration, and model deployment. DS1 illustrated his point of view:

> "*Having well-produced documentation greatly impacts development. For example, if I have a requirement for extremely low latency, it may affect how the model will be developed and made available. The fact that this is documented explicitly assists in the collaboration between the software engineer and the data scientist. [...] If the data scientist comes up with an extremely slow approach, it will be clear to everyone that the model is not ready for production.*"

DS5, a data scientist, emphasized the importance of this recommendation during the interaction with software engineers:

> "*Especially when defining APIs, this documentation is important for validating what will be done with the software engineers.*"

5.3.5 Define clear responsibilities and internal processes with clear boundaries for data scientists and software engineers

The fourth recommendation we discussed was **defining clear responsibilities and internal processes with clear boundaries for data scientists and software engineers**. Table 5.10 illustrates how our participants evaluated the relevance of this recommendation.

Table 5.10: Agreement on the Fourth Recommendation

Task	**Participants who Agreed**	**Participants who Partially Agreed**	**Participants who Partially Disagreed**	**Participants who Disagreed**
Data Access Definition	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-
Data Selection	DS1, DS2, DS3, DS4, SE2, DS5	SE1	-	-
ML Model Evaluation	DS1, DS2, DS3, DS4, SE2, DS5	-	SE1	-
ML Artifact Storage	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-
ML Model Availability	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-
ML Model Integration	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-
ML Model Deployment	DS1, SE1, DS2, DS3, DS4, SE2, DS5	-	-	-

Analyzing the table, we can see that almost all participants agreed with the importance of this recommendation for all tasks. DS3 explained a reason for this:

"*This recommendation is important regardless of how much interaction occurs during each task. Even if there is an activity where there is not supposed to be any collaboration between a software engineer and a data scientist, this must be clear for everyone to avoid someone doing something that is not their responsibility. This recommendation will help define what interactions will happen during the project.*"

DS5 also highlighted the importance of this recommendation:

"*Even for tasks with reduced interaction, following this recommendation guarantees everyone knows their role and what they must do.*"

While discussing the votes, DS1 exemplified how this recommendation is important when defining the data scientists' responsibilities:

"*The data scientist's role can often get confused with other roles. As I mentioned before, they are sometimes also considered database administrators. In addition, they may be confused with software engineers and expected to handle model deployment solely. When you clearly define each role's responsibility, communication becomes easier.*"

SE1 partially disagreed with the relevance of this recommendation during ML model evaluation:

"*In the teams I have worked in, explicitly defining responsibilities and boundaries was never necessary during this task. Both data scientists and software engineers already knew what was expected from them without a previous formal explanation. I believe this task should be mostly carried out by data scientists, as there is no need for software engineers to be involved.*"

Finally, DS2 mentioned another advantage promoted by this recommendation:

"*Clarifying boundaries helps a lot in communication, especially when evaluating the work we need to do. From the point of view of system architecture, these definitions help us visualize who will be responsible for each system component.*"

5.3.6 Support interdisciplinarity between data scientists and software engineers

The fifth recommendation we discussed was **supporting interdisciplinarity between data scientists and software engineers**. Table 5.11 displays how our participants evaluated the relevance of this recommendation.

Table 5.11: Agreement on the Fifth Recommendation

Task	Participants who Agreed	Participants who Partially Agreed	Participants who Partially Disagreed	Participants who Disagreed
Data Access Definition	SE1, DS2	DS1, DS5	DS3, DS4, SE2	-
Data Selection	SE1	DS1, DS2	DS3, DS4, SE2, DS5	-
ML Model Evaluation	SE1	DS1, DS2	DS3, DS4, SE2, DS5	-
ML Artifact Storage	DS1, SE1, DS2	-	DS3, DS4, SE2, DS5	-
ML Model Availability	DS1, SE1, DS2	-	DS4, SE2	DS3, DS5
ML Model Integration	DS1, SE1, DS2	SE2, DS5	DS3, DS4	-
Deploy the ML model	SE1, DS2	DS1, DS3	DS4, SE2, DS5	-

DS3, DS4, SE2, and DS5 partially disagreed with this recommendation's relevance for most tasks. They shared a similar view, explained by DS4:

"*Knowing about other fields is always beneficial, but I do not consider this essential for any activity.*"

DS5 considered this recommendation relevant for specific tasks:

> "*During data access definition, having both actors working closely may speed up the process. This recommendation can also help during model integration, as this task requires a lot of collaboration between data scientists and software engineers.*"

DS1 did not consider this recommendation vital for most tasks. However, he described how it could help during artifact storage and ML model integration:

> "*For data access definition, data selection, model evaluation, and model deployment, I think interdisciplinarity is interesting but not vital. I believe a lack of knowledge exchange between the actors during these tasks would not threaten the project. On the other hand, I consider this recommendation important for the other tasks. For artifact storage [...], both actors need to know how the model was developed and what should be versioned. The same goes for model availability and integration: how to consume the model and its inputs and outputs must be clear to everyone.*"

According to SE1, this recommendation is valuable for all tasks. He explained:

> "*Having another person beside you while working is always useful, especially for catching something you missed. For instance, during data selection, the software engineer might look at the data selected by the data scientist and notice a field that could be included.*"

DS2 described how acquiring software engineering skills could be interesting for a data scientist:

> "*Data scientists are usually more interested in research and generating insights through data analysis, so they may not know how a solution can actually be operationalized and sustained over time. Before deploying to production, it is important to understand what infrastructure is available, how the model will be updated, and how data will be continuously obtained. I think this kind of knowledge is more related to software engineering. As data scientists become familiarized with these concepts, their communication with software engineers will improve.*"

SE1 emphasized another advantage for data scientists:

> "*If the data scientists know how model deployment works, they can design the ML model with this process in mind. This would make the deployment task less arduous, especially if the project is still in its initial phase.*"

DS2 highlighted this knowledge exchange could also be beneficial for software engineers:

> "*After the system is deployed and operational, the actor responsible for maintaining it is usually a software engineer, not a data scientist. This recommendation motivates the software engineer to know more about model characteristics, such as if it is a separate service or how it is consumed. This knowledge would be very helpful, for example, if the software engineer notices an opportunity to implement incremental learning.*"

5.3.7 Organize regular meetings for showcasing team activities

Finally, the sixth recommendation we discussed was **organizing regular meetings for showcasing team activities**. Table 5.12 illustrates how our participants evaluated the relevance of this recommendation.

Table 5.12: Agreement on the Sixth Recommendation

Task	Participants who Agreed	Participants who Partially Agreed	Participants who Partially Disagreed	Participants who Disagreed
Data Access Definition	SE1, DS2, DS3, DS4, DS5	DS1, SE2	-	-
Data Selection	SE1	DS1, DS2, DS3, DS4, DS5	SE2	-
ML Model Evaluation	DS3, DS4, DS5	DS1, SE1, DS2, SE2	-	-
ML Artifact Storage	DS1, DS2	SE1, DS3, DS4, DS5	SE2	-
ML Model Availability	DS1, DS2	SE1, DS3, DS4, DS5	SE2	-
ML Model Integration	DS1, DS2, DS3, DS4, SE2, DS5	SE1	-	-
ML Model Deployment	DS3, DS4, DS5	DS1, SE1, DS2, SE2	-	-

Once again, almost all participants either agreed or partially agreed with the relevance of this recommendation for all tasks. One of them was DS3:

> "*The interaction between software engineers and data scientists will be greatly facilitated if you know exactly what each person is doing and what is happening in the project. This recommendation is valid for monitoring the team and exchanging knowledge, as you can even schedule technical meetings if needed. At the very least, these regular meetings to find out how things are going and what is being done already help a lot.*"

SE1 stressed this recommendation makes more sense for some tasks than others:

> "*When defining data access and selecting data, I think the meetings would help. For the other tasks, I consider regular meetings useful, but not critical. After the team has agreed on all definitions, an occasional conversation between members should be enough. Even if they are not that frequent, they would be enough to ask questions and answer doubts.*"

DS1 highlighted the relevance of this recommendation will depend on the level of collaboration that data scientists and software engineers have during each task:

> "*I love daily meetings because they bring team members from different areas together to witness the whole project's evolution. For tasks that require a greater synergy between software engineers and data scientists, I agree that there should be regular meetings to keep everyone up to date. However, for tasks where I do not see such a strong synergy, I believe these meetings are nice to have so that everyone knows what is happening, but they are not crucial. In the case of data selection [...], a meeting to discuss this task would be good for the software engineers to be more informed about what is happening, but it is not essential for their work. The same is true for data access definition: showcasing how data is obtained might be interesting for data scientists, but they do not have to know where the data comes from as long as they have access to it.*"

5.4 Discussion

5.4.1 What is the perception of software engineers and data scientists on which tasks they most collaborate?

All participants expressed opinions on each of the tasks we had defined, and we could identify similar arguments in both sessions. Based on their responses, we discuss our study's findings for each task in the next paragraphs.

Collaboration plays an important part in data access definition. Software engineers can help data scientists acquire the data needed to develop and validate the ML model, as they usually know how to access the available APIs and databases. However, there may also be cases where the data scientists obtain the data independently. This is the case when data is received manually by another actor, such as a customer representative, or when data scientists are also database administrators. In these scenarios, an interaction with a software engineer may not be required.

Data selection is usually a data scientist's job, given this is their field of expertise (Kim et al., 2017). This makes collaboration with software engineers less frequent during this task. Moreover, participating in this process may sometimes seem uninteresting for the software engineers, which contributes to less interaction. Yet, during this task, a data scientist may come across incomplete data for training the model. If this leads to any modifications in data access definitions, then a software engineer may be involved. Collaboration between software engineers and data scientists may also not be needed for ML model evaluation, as data scientists should be able to perform this task independently. Once again, software engineers might lack interest in understanding the metrics used for evaluation. Since they usually do not possess this knowledge, it might not be valuable to interact with them during this task.

Interaction between the actors can benefit the team when specifying ML artifact storage. Creating and maintaining an adequate storage infrastructure for the model is a challenge, given the large size of some ML artifacts and the need to manage their different versions. This difficulty can be mitigated by guaranteeing that software engineers and data scientists are involved when defining how and where each artifact will be stored. This helps familiarize the data scientists with the available storage infrastructure, and they can use this information and their knowledge of the ML model to point out the most suitable storage possibilities.

Collaboration is fundamental for assessing ML model availability and integration. These two tasks relate to the interaction of the ML model with other system components. They include defining how the model will be consumed and what output it will provide (Villamizar et al., 2024). For this reason, performing these tasks implies several discussions between data scientists and software engineers. These discussions are also relevant during model deployment. Even though software engineers usually handle deployment tasks, data scientists should also be present to explain how the model works and how it should be executed. Having them next to the software engineers can also help resolve latency issues caused by the model.

Our results illustrated how collaboration between software engineers and data scientists is fundamental for several tasks, as their interaction can greatly help face the challenges involved in developing ML-enabled systems. Tasks that can benefit from this interaction include, for example, specifying data access and integrating the ML model. For tasks such as data selection and ML model evaluation, however, the team must examine if assigning a single actor as responsible would be more advantageous than fostering collaboration.

5.4.2 What is the perception of software engineers and data scientists on how to improve collaboration between them?

Participants raised important features of each recommendation we proposed. The fact they were analyzed in the context of specific tasks allowed participants to give practical examples to support their points of view. In the following paragraphs, we will detail the findings from each recommendation's assessment.

Involving data scientists and business owners when eliciting and analyzing requirements can be very useful when defining how data will be accessed. Business owners are usually familiar with the data and can explain what they represent and how to access them correctly. This, in turn, will improve the communication between software engineers and data scientists when discussing this topic. Model evaluation can also benefit from this recommendation. Business owners can use their understanding of the data to define the most suitable performance metrics, and evaluating metrics such as the system's latency may stimulate the actors' collaboration. For the other tasks, the influence of this recommendation will depend on the project's context. For example, when different teams are involved in ML model integration, having a business owner coordinating them during requirements analysis may be helpful.

Providing ML literacy for all project stakeholders is important for collaboration because it aids in establishing a common terminology inside the team, making communication more efficient. It also helps team members become familiar with the model being developed, as it is vital that they clearly understand the model's goal and the type of output it generates (Piorkowski et al., 2021). Participants argued that this recommendation is valuable for collaboration during model deployment, model integration, and ML artifact storage. However, it might not be as relevant for other tasks. For example, ML literacy may be less useful for data access definition than simply providing practical data acquisition instructions, such as what commands should be used to obtain data or what database should be accessed.

Documentation constitutes an important tool for enhancing collaboration (Nahar et al., 2022). Having all requirements and definitions explicitly stated on a document enables team members to understand what is expected from the system. This allows data scientists and software engineers to evaluate each other's work despite their different fields of expertise. Documentation can also be beneficial when the actors have to plan their tasks. For model integration and availability, for example, documenting API contracts and model inputs and outputs can help clearly define what each actor must do, making the development process more efficient. These advantages reinforce the importance of developing well-written documents and making them available for all team members to consult.

Clearly defining responsibilities and boundaries between data scientists and software engineers is critical, even for tasks that may not require much collaboration. This recommendation raises awareness about the tasks each actor is supposed to perform, enhancing team alignment. For example, it must be clear from the beginning to everyone in the team whether data scientists should be responsible for handling ML model deployment. Doing this will avoid problems later in the project and improve their communication with software engineers during this task if needed. This recommendation can also be useful when specifying the system's architecture, as each actor's responsibilities might be directly related to the system components they will have to develop.

Supporting interdisciplinarity between data scientists and software engineers may not be as vital as some of the other recommendations we assessed. If responsibilities are correctly assigned, a lack of knowledge exchange between the actors during the tasks should not compromise the team's performance. Nevertheless, following this recommendation can still be useful since it enables team members to work more closely, facilitating communication. For tasks such as model deployment and integration, where collaboration may be required, interdisciplinarity allows actors to learn more about each other's work. Besides improving communication, this can also benefit the team in the long term. For instance, data scientists may use this knowledge to develop ML models in such a way that makes their deployment easier in the future. Similarly, software engineers can use what they have learned about the model to assess the implementation of incremental learning.

Organizing regular meetings provides several advantages for team alignment and collaboration overall. They allow team members to know what is being done and discuss the project's current state. Moreover, these meetings can foster knowledge exchange and help resolve issues during development. However, to improve collaboration, the team must organize these meetings properly. For example, it may not be interesting to schedule meetings with both software engineers and data scientists to discuss tasks that do not require their interaction. For this reason, it is important that teams thoroughly evaluate the importance of each meeting based on the project's current state and the team's characteristics. Defining a clear goal for each meeting can help with this evaluation.

Our findings demonstrate how the impact of the recommendations for each task depends on project characteristics and the level of interaction between software engineers and data scientists during task execution. Most recommendations are relevant for tasks that require strong collaboration, such as ML model integration. Still, recommendations like defining clear responsibilities and organizing regular meetings can be valuable regardless of the level of collaboration. Teams working on ML-enabled systems need to analyze the recommendations we have discussed to identify the most appropriate ones for enhancing their performance.

5.4.3 Threats to Validity

This section describes how we handled threats to our study's validity. Once again, we discuss threats to construct validity, internal validity, external validity, and reliability.

A potential threat to **construct validity** is related to our focus group design not being appropriate to investigate collaboration and recommendations in the context of ML-enabled systems. To avoid this threat, we carefully established the tasks and recommendations discussed during the focus groups based on our previous studies and the current literature on collaboration for these systems. To improve the credibility and representativeness of our results, we selected papers published in prestigious venues with findings that were acquired through perceptions from professionals working on industry ML projects.

Threats to **internal validity** include participants not understanding the tasks and recommendations we defined for discussion, as well as applying a different methodology in each focus group. To mitigate these threats, we followed a standardized procedure during the two sessions we conducted. Tasks and recommendations were discussed in the same order, and we timed the discussions to ensure all topics were debated equally. Before each voting phase, we allowed participants to ask questions to better understand the task or recommendation they had to assess. On the other hand, participants could see each other's votes while voting, which may have induced groupthink. We tried to mitigate this by reinforcing that there were no right or wrong answers, as the votes reflected each participant's previous experiences. We also encouraged participants to explain the motivation behind their votes.

A threat to **external validity** concerns our results not being valuable for other teams working on ML-enabled systems. We recognize the limited number of participants in our focus groups, especially software engineers, may be considered a threat to our findings' validity. To mitigate this threat and increase generalizability, we invited experienced professionals who work on different ML-enabled systems to participate in our research. However, the experiences they shared were exclusively based on the context of the projects they participated in. Practitioners should examine our results to project how they can be applied in their respective teams. For this reason, we consider our findings useful for any team developing an ML-enabled system looking to enhance collaboration between software engineers and data scientists.

Finally, to strengthen the **reliability** of our results, we created an online repository[5] with all artifacts developed during this study. This allows other practitioners to validate our results and replicate our research design in future studies.

5.5 Concluding Remarks

In this chapter, we reported two focus group sessions with seven experienced data scientists and software engineers. Our study aimed to comprehend how participants viewed collaboration between these roles during multiple tasks important for building ML-enabled systems. Furthermore, we instigated them about the relevance of several recommendations for improving this collaboration. To do this, we defined seven tasks and six recommendations we considered interesting for evaluation. They were selected based on our previous results and findings from the literature researching this collaboration in-depth.

Our findings illustrated how collaboration between software engineers and data scientists is crucial for most tasks, as knowledge exchange between the actors can improve the team's performance while developing ML-enabled systems. For some tasks, such as data selection and ML model evaluation, the team must analyze if establishing collaboration is more beneficial than assigning only one role as responsible. Regarding the recommendations we proposed, they were considered valuable for different tasks of ML projects. Teams must examine their advantages and prioritize implementing the ones considered most beneficial.

[5]`https://doi.org/10.5281/zenodo.10884480`

6
Conclusion

6.1
Contributions

This dissertation described three studies investigating the interaction between software engineers and data scientists when building ML-enabled systems. They provide factual examples of challenges based on real ML-enabled system projects for different customers. The challenges we described are also mentioned in the current literature, reinforcing their relevance. In addition to explaining each work's results, the chapters also detail the research methodology employed and how the study was planned. This information can be valuable for other researchers who want to conduct new investigations on our research topic. Furthermore, our findings can be helpful for organizations seeking to leverage the collaboration and performance of their teams responsible for developing ML-enabled systems.

In our first study, we defined our research goal as characterizing the interaction between software engineers and data scientists to comprehend how they share responsibilities and collaborate. To do this, we performed a case study following the guidelines by Runeson et al. (2012) for using this research methodology in software engineering. We interviewed four professionals working on a large ML-enabled system and examined the collected data using RTA to identify response patterns. The study's findings revealed how our selected case handled tasks during system development and how data scientists and software engineers interacted. We believe this knowledge can be useful for other practitioners creating ML-enabled systems. By understanding the challenges participants faced in multiple stages of the project, which are thoroughly discussed in this chapter, other teams can improve their planning and task management to avoid them.

Our second study detailed two complementary interviews we performed with professionals working on two different ML-enabled systems. Our intent with this work was to enrich the outcomes of our case study. Furthermore, investigating collaboration in two additional projects allowed us to evaluate our previous findings' generalizability. For this reason, we established the same research questions as before. After conducting the interviews, we reviewed the transcripts and extracted the participants' most relevant comments. The results showcased how the teams had different collaboration practices and responsibility division approaches, which were influenced by each team's configuration and the current stage of their respective projects. This investigation of different scenarios allowed us to compare them with the discoveries of our prior study. In addition, the obstacles discussed in this study and the previous one enabled us to propose recommendations for improving the interaction between data scientists and software engineers. We believe they can guide team leaders and managers who are interested in fostering this interaction.

Our third and final study involved two focus group sessions with seven experienced software engineers and data scientists working on ML-enabled systems. The sessions were planned following the guidelines by Kontio et al. (2008) for employing this method in a software engineering context. Our study aimed to acquire the participants' perceptions regarding the relevance of collaboration for their tasks, and how this interaction could be improved. To do this, we first reviewed our previous results and the literature on collaboration for ML-enabled systems. After that, we selected a group of tasks and recommendations that we considered interesting for discussion. We transcribed all focus group sessions and analyzed each participant's comments to answer our research questions. Our discoveries can effectively contribute to practitioners interested in how collaboration between these actors unfolds during tasks crucial for developing ML-enabled systems, such as ML model integration and deployment. Moreover, the study enabled the identification of practical examples of the effects of each assessed recommendation. We believe this can help other teams working on ML-enabled systems identify the most appropriate recommendations for enhancing their performance.

6.2 Limitations

This section details some limitations of the studies we conducted. First, we acknowledge that the number of subjects in each study is relatively small. We interacted with four participants in our first study, five in our second, and seven in our third. Due to the qualitative research methods we employed, we had to analyze several transcripts for each participant, which limited our ability to increase the number of subjects. To mitigate that, we strived to recruit the most experienced members of each team. In some situations, this was not possible due to the unavailability of these members. We recognize that having a larger number of participants would have increased the reliability of our results.

In our three studies, we selected professionals who work on industry-academia research projects inside PUC-Rio as participants. We did this because we had easier access to them, and because they voluntarily agreed to participate in our research. Even though industrial partners sponsor the projects they work on, we recognize that teams within an academic context tend to have characteristics specific to this environment. For example, it is common to have software engineers or data scientists who are also researchers with a master's degree or a doctorate, which may not be true for teams working inside companies. This, among other differences, may affect collaboration between the actors on the team. Even though we believe our findings can be useful for any team working with ML-enabled systems, we admit that collaboration within teams inside an industrial context may have features that were not addressed in our results.

Some limitations of our third study are also worth discussing. We chose the topics debated in the focus group sessions based on our assessment of the current literature and our prior studies. During the sessions, we did not address other tasks and recommendations beyond the ones we had previously selected. We decided to do this so that sessions would have a predetermined duration. Still, allowing participants to propose additional tasks and recommendations they judged worth discussing would have probably enhanced our findings. Furthermore, given the time we allocated to discussing each task and recommendation, we could not address every single vote cast by each participant. Although the debates captured most of the participants' opinions, we recognize that we could not instigate everyone's participation in all discussions.

6.3 Future Work

By assessing the literature and our research results, we could identify multiple opportunities for future work in the field of collaboration for ML-enabled systems. One possibility includes conducting additional studies with industry practitioners, which would address one of the limitations of our work. Understanding how collaboration between software engineers and data scientists unfolds inside teams with other compositions and companies with different organizational structures can enhance our findings and verify the occurrence of the challenges we reported. The supplementary material available in our online repositories can be used to replicate our studies or serve as a reference for new investigations.

Our research focused on the interaction between software engineers and data scientists. However, future work can also consider expanding this focus to collaboration with other roles, such as business owners and domain experts. Our participants cited these actors multiple times throughout our three studies, which evidences their importance during several tasks of ML-enabled system development. Business owners, for example, are essential for defining requirements, while domain experts play an important part in explaining the data. Hence, investigating collaboration with these actors in-depth may uncover new challenges worth addressing.

Finally, another opportunity for future work relies on exploring the recommendations we evaluated in our focus group sessions. Even though we discussed their relevance for collaboration during multiple tasks, how to apply them and monitor their effects on the team is a subject that was not thoroughly debated. Given how positively our participants evaluated most of the recommendations, we believe this additional investigation can be very beneficial. Besides investigating our proposed recommendations, researchers can also look into new recommendations and artifacts to improve communication and knowledge sharing between data scientists and software engineers building ML-enabled systems.

Bibliography

Aho, T., Sievi-Korte, O., Kilamo, T., Yaman, S., and Mikkonen, T. (2020). Demystifying data science projects: A look on the people and process of data science today. In *Product-Focused Software Process Improvement: 21st International Conference, PROFES 2020, Turin, Italy, November 25–27, 2020, Proceedings 21*, pages 153–167. Springer.

Almeida, C., Kalinowski, M., and Feijó, B. (2021). A systematic mapping of negative effects of gamification in education/learning systems. In *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 17–24. IEEE.

Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., and Zimmermann, T. (2019). Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE.

Basili, V. R. and Rombach, H. D. (1988). The tame project: Towards improvement-oriented software environments. *IEEE Transactions on software engineering*, 14(6):758–773.

Begel, A. and Zimmermann, T. (2014). Analyze this! 145 questions for data scientists in software engineering. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 12–23, New York, NY, USA. Association for Computing Machinery.

Braun, V. and Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative research in psychology*, 3(2):77–101.

Braun, V. and Clarke, V. (2012). *Thematic analysis.* American Psychological Association.

Braun, V. and Clarke, V. (2019). Reflecting on reflexive thematic analysis. *Qualitative research in sport, exercise and health*, 11(4):589–597.

Braun, V. and Clarke, V. (2021). Can i use ta? should i use ta? should i not use ta? comparing reflexive thematic analysis and other pattern-based qualitative analytic approaches. *Counselling and psychotherapy research*, 21(1):37–47.

Brown, N. and Stockman, T. (2013). Examining the use of thematic analysis as a tool for informing design of new family communication technologies. In *27th International BCS Human Computer Interaction Conference (HCI 2013) 27*, pages 1–6.

Busquim, G., Villamizar, H., Lima, M. J., and Kalinowski, M. (2024). On the interaction between software engineers and data scientists when building machine learning-enabled systems. In *International Conference on Software Quality*, pages 55–75. Springer.

Caroli, P. (2017). Lean inception. *São Paulo, BR: Caroli. org.*

Coelho, G. M., Ramos, A. C., de Sousa, J., Cavaliere, M., de Lima, M. J., Mangeth, A., Frajhof, I. Z., Cury, C., and Casanova, M. A. (2022). Text classification in the brazilian legal domain. In *ICEIS (1)*, pages 355–363.

Cruzes, D. S. and Dyba, T. (2011). Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement*, pages 275–284. IEEE.

Ishikawa, F. and Yoshioka, N. (2019). How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey. In *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, pages 2–9. IEEE.

Kim, M., Zimmermann, T., DeLine, R., and Begel, A. (2017). Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, 44(11):1024–1038.

Kontio, J., Bragge, J., and Lehtola, L. (2008). The focus group method as an empirical tool in software engineering. In *Guide to advanced empirical software engineering*, pages 93–116. Springer.

Kuramoto, A. S. R., De Oliveira, A. D. C. M., Torres, P. H. L., Fernandes, W. P. D., Lopes, H. C. V., Schwaner, W. K., Paravidino, B. I., Pereira, C. S., Happ, P. N., Comandulli, S., et al. (2023). Method of assessment of the quality of the burn of the gases in the flare and adjustment to the vapor flow rate in a continuous and constant way. US Patent App. 17/890,539.

Lewis, G. A., Bellomo, S., and Ozkaya, I. (2021). Characterizing and detecting mismatch in machine-learning-enabled systems. In *2021 IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, pages 133–140. IEEE.

Mailach, A. and Siegmund, N. (2023). Socio-technical anti-patterns in building ML-enabled software: Insights from leaders on the forefront. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pages 690–702. IEEE.

Nahar, N., Zhou, S., Lewis, G., and Kästner, C. (2022). Collaboration challenges in building ML-enabled systems: communication, documentation, engineering, and process. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, pages 413–425, New York, NY, USA. Association for Computing Machinery.

Nazir, R., Bucaioni, A., and Pelliccione, P. (2023). Architecting ML-enabled systems: Challenges, best practices, and design decisions. *Journal of Systems and Software*, page 111860.

Piorkowski, D., Park, S., Wang, A. Y., Wang, D., Muller, M., and Portnoy, F. (2021). How AI Developers Overcome Communication Challenges in a Multidisciplinary Team: A Case Study. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):131:1–131:25.

Rahman, M. S., Khomh, F., Rivera, E., Guéhéneuc, Y.-G., and Lehnert, B. (2022). Challenges in Machine Learning Application Development: An Industrial Experience Report. In *2022 IEEE/ACM 1st International Workshop on Software Engineering for Responsible Artificial Intelligence (SE4RAI)*, pages 21–28.

Runeson, P., Host, M., Rainer, A., and Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples.* John Wiley & Sons.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28.

Strauss, A. and Corbin, J. (1998). Basics of qualitative research techniques.

Villamizar, H., Escovedo, T., and Kalinowski, M. (2021). Requirements engineering for machine learning: A systematic mapping study. In *2021 47th*

Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 29–36. IEEE.

Villamizar, H., Kalinowski, M., et al. (2022a). A catalogue of concerns for specifying machine learning-enabled systems. In *Workshop on Requirements Engineering (WER)*, page 14.

Villamizar, H., Kalinowski, M., and Lopes, H. (2022b). Towards perspective-based specification of machine learning-enabled systems. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 112–115. IEEE.

Villamizar, H., Kalinowski, M., Lopes, H., and Mendez, D. (2024). Identifying concerns when specifying machine learning-enabled systems: A perspective-based approach. *Journal of Systems and Software*, page 112053.

Wan, Z., Xia, X., Lo, D., and Murphy, G. C. (2019). How does machine learning change software development practices? *IEEE Transactions on Software Engineering*, 47(9):1857–1871.

Yang, Q., Scuito, A., Zimmerman, J., Forlizzi, J., and Steinfeld, A. (2018). Investigating how experienced UX designers effectively work with machine learning. pages 585–596.

Zdanowska, S. and Taylor, A. S. (2022). A study of UX practitioners roles in designing real-world, enterprise ML systems. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, pages 1–15, New York, NY, USA. Association for Computing Machinery.

Zhang, A. X., Muller, M., and Wang, D. (2020). How do Data Science Workers Collaborate? Roles, Workflows, and Tools. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):22:1–22:23.

A
Thematic Analysis

This appendix presents all codes, themes, and sub-themes generated during the case study's data analysis with RTA. Figure A.1 illustrates the first version, while Figure A.2 portrays the final version after refinement iterations.

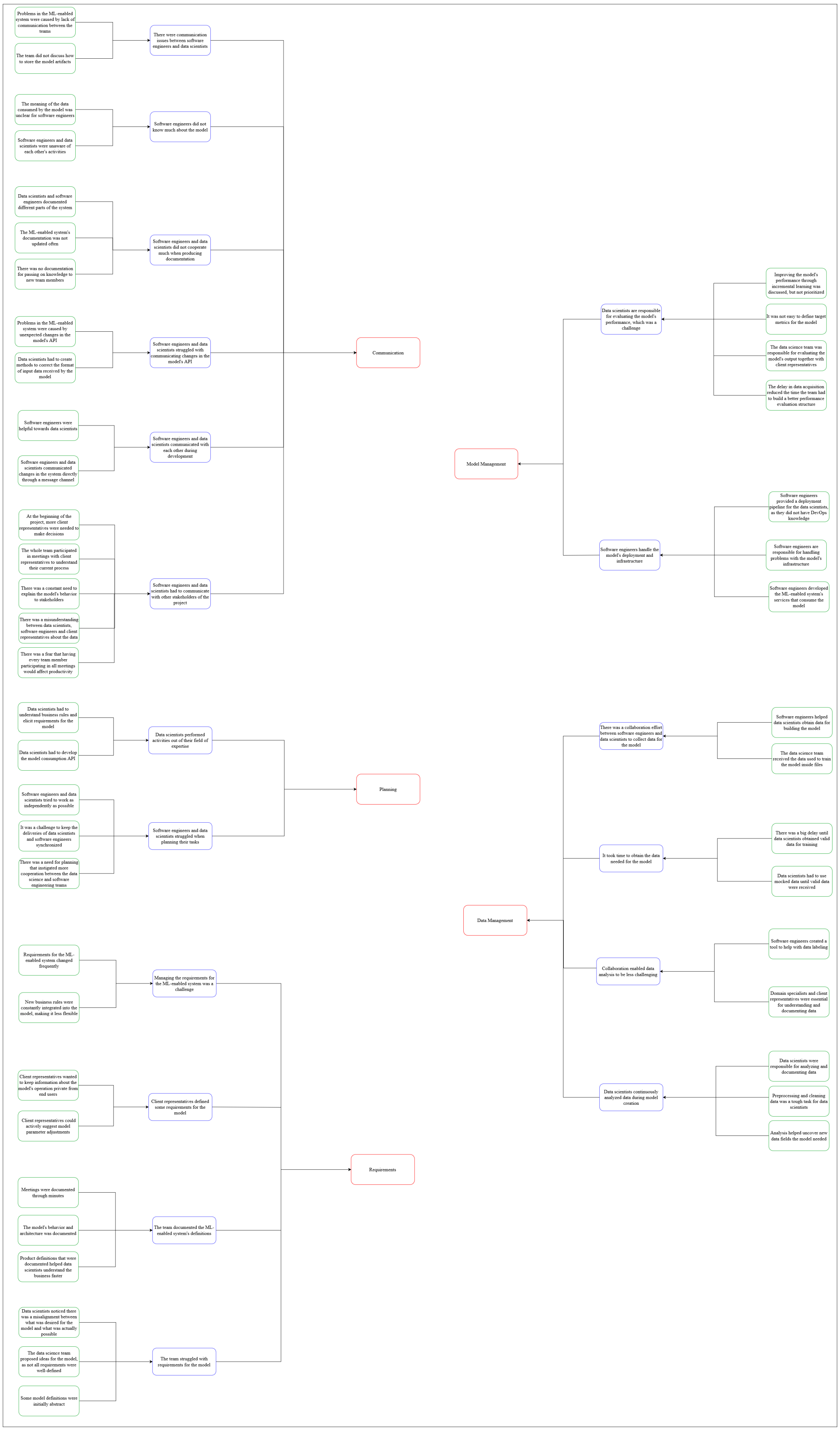
Problems in the ML-enabled system were caused by lack of communication between the teams
The team did not discuss how to store the model artifacts
There were communication issues between software engineers and data scientists
The meaning of the data consumed by the model was unclear for software engineers
Software engineers and data scientists were unaware of each other's activities
Software engineers did not know much about the model
Data scientists and software engineers documented different parts of the system
The ML-enabled system's documentation was not updated often
There was no documentation for passing on knowledge to new team members
Software engineers and data scientists did not cooperate much when producing documentation
Problems in the ML-enabled system were caused by unexpected changes in the model's API
Data scientists had to create methods to correct the format of input data received by the model
Software engineers and data scientists struggled with communicating changes in the model's API
Communication
Software engineers were helpful towards data scientists
Software engineers and data scientists communicated changes in the system directly through a message channel
Software engineers and data scientists communicated with each other during development
At the beginning of the project, more client representatives were needed to make decisions
The whole team participated in meetings with client representatives to understand their current process
There was a constant need to explain the model's behavior to stakeholders
There was a misunderstanding between data scientists, software engineers and client representatives about the data
There was a fear that having every team member participating in all meetings would affect productivity
Software engineers and data scientists had to communicate with other stakeholders of the project
Data scientists had to understand business rules and elicit requirements for the model
Data scientists had to develop the model consumption API
Data scientists performed activities out of their field of expertise
Planning
Software engineers and data scientists tried to work as independently as possible
It was a challenge to keep the deliveries of data scientists and software engineers synchronized
There was a need for planning that instigated more cooperation between the data science and software engineering teams
Software engineers and data scientists struggled when planning their tasks
Requirements for the ML-enabled system changed frequently
New business rules were constantly integrated into the model, making it less flexible
Managing the requirements for the ML-enabled system was a challenge
Client representatives wanted to keep information about the model's operation private from end users
Client representatives could actively suggest model parameter adjustments
Client representatives defined some requirements for the model
Requirements
Meetings were documented through minutes
The model's behavior and architecture was documented
Product definitions that were documented helped data scientists understand the business faster
The team documented the ML-enabled system's definitions
Data scientists noticed there was a misalignment between what was desired for the model and what was actually possible
The data science team proposed ideas for the model, as not all requirements were well-defined
Some model definitions were initially abstract
The team struggled with requirements for the model
Improving the model's performance through incremental learning was discussed, but not prioritized
It was not easy to define target metrics for the model
The data science team was responsible for evaluating the model's output together with client representatives
The delay in data acquisition reduced the time the team had to build a better performance evaluation structure
Data scientists are responsible for evaluating the model's performance, which was a challenge
Model Management
Software engineers provided a deployment pipeline for the data scientists, as they did not have DevOps knowledge
Software engineers are responsible for handling problems with the model's infrastructure
Software engineers developed the ML-enabled system's services that consume the model
Software engineers handle the model's deployment and infrastructure
Software engineers helped data scientists obtain data for building the model
The data science team received the data used to train the model inside files
There was a collaboration effort between software engineers and data scientists to collect data for the model
There was a big delay until data scientists obtained valid data for training
Data scientists had to use mocked data until valid data were received
It took time to obtain the data needed for the model
Data Management
Software engineers created a tool to help with data labeling
Domain specialists and client representatives were essential for understanding and documenting data
Collaboration enabled data analysis to be less challenging
Data scientists were responsible for analyzing and documenting data
Preprocessing and cleaning data was a tough task for data scientists
Analysis helped uncover new data fields the model needed
Data scientists continuously analyzed data during model creation

Figure A.1: Codes (in green) and the corresponding themes (in blue). To better organize our findings, we labeled the themes according to the aspects they relate to (in red)

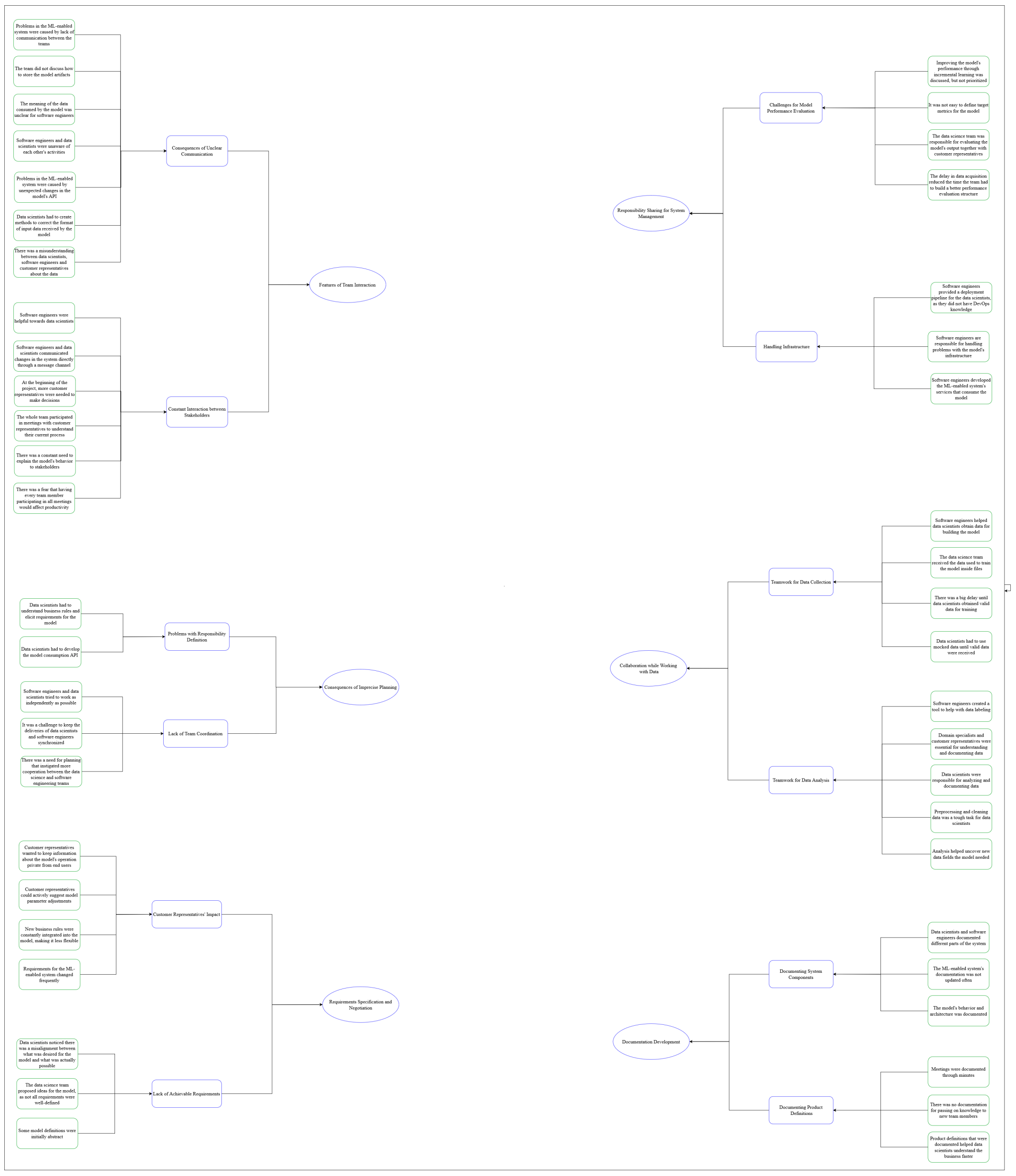
Problems in the ML-enabled system were caused by lack of communication between the teams
The team did not discuss how to store the model artifacts
The meaning of the data consumed by the model was unclear for software engineers
Software engineers and data scientists were unaware of each other's activities
Problems in the ML-enabled system were caused by unexpected changes in the model's API
Data scientists had to create methods to correct the format of input data received by the model
There was a misunderstanding between data scientists, software engineers and customer representatives about the data
Consequences of Unclear Communication
Features of Team Interaction
Software engineers were helpful towards data scientists
Software engineers and data scientists communicated changes in the system directly through a message channel
At the beginning of the project, more customer representatives were needed to make decisions
The whole team participated in meetings with customer representatives to understand their current process
There was a constant need to explain the model's behavior to stakeholders
There was a fear that having every team member participating in all meetings would affect productivity
Constant Interaction between Stakeholders
Data scientists had to understand business rules and elicit requirements for the model
Data scientists had to develop the model consumption API
Problems with Responsibility Definition
Software engineers and data scientists tried to work as independently as possible
It was a challenge to keep the deliveries of data scientists and software engineers synchronized
There was a need for planning that instigated more cooperation between the data science and software engineering teams
Lack of Team Coordination
Consequences of Imprecise Planning
Customer representatives wanted to keep information about the model's operation private from end users
Customer representatives could actively suggest model parameter adjustments
New business rules were constantly integrated into the model, making it less flexible
Requirements for the ML-enabled system changed frequently
Customer Representatives' Impact
Requirements Specification and Negotiation
Data scientists noticed there was a misalignment between what was desired for the model and what was actually possible
The data science team proposed ideas for the model, as not all requirements were well-defined
Some model definitions were initially abstract
Lack of Achievable Requirements
Improving the model's performance through incremental learning was discussed, but not prioritized
It was not easy to define target metrics for the model
The data science team was responsible for evaluating the model's output together with customer representatives
The delay in data acquisition reduced the time the team had to build a better performance evaluation structure
Challenges for Model Performance Evaluation
Responsibility Sharing for System Management
Software engineers provided a deployment pipeline for the data scientists, as they did not have DevOps knowledge
Software engineers are responsible for handling problems with the model's infrastructure
Software engineers developed the ML-enabled system's services that consume the model
Handling Infrastructure
Software engineers helped data scientists obtain data for building the model
The data science team received the data used to train the model inside files
There was a big delay until data scientists obtained valid data for training
Data scientists had to use mocked data until valid data were received
Teamwork for Data Collection
Collaboration while Working with Data
Software engineers created a tool to help with data labeling
Domain specialists and customer representatives were essential for understanding and documenting data
Data scientists were responsible for analyzing and documenting data
Preprocessing and cleaning data was a tough task for data scientists
Analysis helped uncover new data fields the model needed
Teamwork for Data Analysis
Data scientists and software engineers documented different parts of the system
The ML-enabled system's documentation was not updated often
The model's behavior and architecture was documented
Documenting System Components
Documentation Development
Meetings were documented through minutes
There was no documentation for passing on knowledge to new team members
Product definitions that were documented helped data scientists understand the business faster
Documenting Product Definitions

Figure A.2: Codes (in green) and the corresponding themes and sub-themes (in blue)

B Interview Transcripts - Case Study

This appendix comprises the revised interview transcripts we analyzed during the case study. All interviews were translated from Portuguese to English.

B.1 Transcription of DS1's interview

Q: Did you participate in discussions about the model's objectives?

A: We had meetings to understand the problem the model would solve and define a strategy to achieve its objective. In my first month of work, I read documents to understand more about the business and legal terms.

Q: What documents did you consult?

A: Several research reports were developed at the beginning of the project. I consulted one about the system, which contained some use cases explaining the problem we had to solve. I also consulted another report on decision trees and other AI techniques researched. These documents helped me to understand the business faster.

Q: And do you remember who attended these definition meetings?

A: All the team participated, including data scientists, domain experts, UX designers, and software engineers. Client representatives also participated in some meetings. Even with the help of the reports, many project definitions constantly changed, so I was always learning during the project. The domain experts were always by our side to answer questions, which was essential for building the model. We created a flowchart with all the rules the model considered and documented the meetings through minutes. We even had an episode where it was necessary to resort to these minutes to prove that the team had made certain decisions in a previous encounter.

Q: Were the ML-enabled system's objectives clear to you?

A: These definitions changed throughout the project. In the beginning, for example, we had defined that the model would be as flexible as possible. We realized during later meetings this would not be well accepted, as it would make the model's results less predictable. In our case, estimating a target accuracy for the model was also difficult. We then defined that a supervisory committee, composed of client representatives, would be responsible for validating the results produced.

Q: How was the functionality of the model within the system defined?

A: This was a long process. We had to generate results and explain the business rules behind them and how the model worked to the client. There was an expectation that the model would learn automatically without these fixed rules. However, the clients voted in favor of them to prevent the model from creating deals that involved a lot of money and to prevent them from becoming biased towards some consumer or company. Establishing these well-defined rules contributed to the transparency of the model.

Q: How were the data used to train the model obtained?

A: Since we worked with legal processes containing sensitive data, we needed a secure way to obtain them. The development team defined how this would be done together with client representatives. They created a tool to download the data and make it available on our server. This download is done manually whenever new data is available. The data consisted of PDFs of different processes. During data analysis, we also requested more data from the client.

Q: Were there any difficulties during data analysis?

A: Yes, we needed to annotate the text in the PDFs, which is a complicated task. We devised several methods to extract the data. We had to discuss the amount of annotated data needed to train and test the model, how long this annotation would take, and the best way to perform this annotation. The development team helped us to create a text annotation system and make it available to the domain experts. They indicated which document parameters were most interesting for extraction and annotated the data for us, which we used for model training.

Q: Did any other actors participate in data activities besides data scientists?

A: No, it was only our team with the domain experts. The results, however, were presented to everyone.

Q: Has an effort been made to document the data?

A: We created a dictionary for the extracted data, which we stored in spreadsheets. The first action we took after obtaining the data was to analyze it and understand its meaning, which we did with the help of the domain experts. We also documented the models' accuracy during the tests and our client presentations.

Q: What difficulties appeared during the construction of the model?

A: Throughout the project, we discussed what data we would take into account to generate the settlement agreement, and it took us a while to figure out what data we needed to request from the client. We defined some data fields during development, while others were defined during meetings.

Q: How was the consumption mode of the model defined?

A: Among the data scientists, I had the most experience outside the research field, so I was responsible for the API that the rest of the system would use to consume the model. Other developers helped me, showing me if I was developing the API correctly.

Q: Was there any discussion about model updates and implementing incremental learning?

A: The model's behavior is dictated by a decision tree, whose configuration we defined in a JSON file. Therefore, to update the model or add a new rule, it is only necessary to modify this file. However, it is important to know about the structure of the tree nodes to change the tree correctly.

We defined that, from time to time, the model would be retrained with new data to update the parameters used. The client wanted the training to be done automatically, but this process is still ongoing.

Q: How was the relationship with the software engineers during the integration of the model with the rest of the system?

A: After the development of the API, what changed the most was the model's input and output data. The software engineers documented the input data, while we documented the output data in our repository. When there is a change, like new data that needs to be included in the API, or when there is an issue, we communicate directly through a channel in Slack. We then update the documentation afterward, if needed. We do not have any problems in terms of communication between the teams, as the software engineers are very attentive and available to us.

B.2 Transcription of DS2's interview

Q: Did you participate in discussions about the model's objectives?

A: No, the project's scope and our team's objectives were already predefined when I joined. However, they changed during the project.

Q: How did you come into contact with these objectives?

A: Initially, the objectives were abstract. We did not have all the project requirements elicited. We had several meetings to define what the product would look like, using tools like Lean Inception. Other team members, like the software engineers, also participated in these meetings. On the other hand, client representatives did not participate in these meetings, and it would have been interesting to have them together with us at that moment to make the companies feel comfortable with the product.

Q: After that, did the ML-enabled system objectives become clear?

A: From the model's point of view, what we would do could have been more specific. There was a misalignment between what was desired and what was possible, which led to many meetings. No one came to us with a model requirement. In fact, we were the ones proposing ideas. Initially, we did not anticipate the model being a key system part.

Each model definition was documented through presentations we did in meetings to showcase what our team was proposing. The architecture of the model was also described in a formal document.

Q: You mentioned the absence of client representatives in meetings. Did they prioritize any features for the model?

A: The requirements were abstract, like "the model needs to be fast" or "the system needs to be easy to use". We had difficulties because we did not include more client representatives when we defined the product's concepts. They could have helped us by making decisions. Instead, we made decisions internally. We had to revisit some of these decisions later, while we were lucky not to in others.

Q: How were the data used to train the model obtained?

A: The data access method was not defined from the beginning, and no database was provided for the model. Instead, data provided by the client was downloaded with the help of the software engineering team. The retraining of the model is done manually.

Q: Were there any difficulties during data analysis?

A: Pre-processing the data was complex. We received raw data, so cleaning procedures were necessary, and we also put a lot of effort into annotating the data. It took a lot of effort to analyze and process the data received so that we could work on the model. This situation also affected what algorithms we could use for the model.

We also had problems with data availability. It took us some months to get all the valid data needed for testing. Therefore, we had to initially use mocked data, which later became different from the real data, leading to rework.

Q: Did any other actors participate in data activities besides data scientists?

A: The software engineers made it possible to download the data, but our team did all the analysis. They were also present in the definition of the model consumption API and the format of each data. We presented our model studies to the project's stakeholders for them to evaluate if the results were adequate or not.

Q: Has an effort been made to document the data?

A: We have some documentation, but the data could be better explained.

Q: What difficulties appeared during the construction of the model?

A: The time taken to obtain valid data hindered the time to create a better performance evaluation framework. In addition, the client did not define any model metrics, like a minimum accuracy, as they probably could not determine such value. We also struggled with new requirements that emerged throughout the project, and also with model requirements related to legislation and transparency. These requirements generated doubts and impacted the model construction. For example, the model was supposed to be "auditable", and we did not know what that implied.

Q: How was the consumption mode of the model defined?

A: An API was created to be consumed by the rest of the system. There was a discussion with the whole technical team to define where the model would be hosted, in which the software engineers also participated.

Q: Was there any discussion about model updates and implementing incremental learning?

A: The model update is currently done manually, as incremental learning was not prioritized.

Q: How was the relationship with the software engineers during the integration of the model with the rest of the system?

A: In the beginning, it was difficult. We were a research team, not a development team. Still, we needed to develop versions and generate specifications for the model. Our team was responsible for understanding the entire business flow and legal procedures so that we could build the model. Someone else could have done this survey and delivered the requirements to us.

Our team was responsible for developing and maintaining the model consumption API. This responsibility could have been given to the software engineering team. Problems with input data formats when calling the model's API should not have been our responsibility either, as this data had to be in the expected form before communication happened. However, we had to build workarounds to correct some input data formats, which made the system's integration with the model take time and generate rework.

Each unanticipated requirement that arose implied an adaptation in the system, causing errors and undermining the project's planning. There was also a misalignment in planning regarding each team's dependencies. For example, software engineers sometimes depended on a change in the model that was not in our backlog. The roles of each team ended up not being clear, which led to problems in the API. We lacked comprehensive planning that involved both teams more.

B.3 Transcription of SE1's interview

Q: Did you participate in any discussions about user interactions with the model?

A: I joined the team in the middle of the project, so I did not participate much in the initial discussions. I noticed that client representatives could actively suggest model parameter adjustments. Another topic they discussed was keeping information about the model's operation private from end users. This was done to prevent them from learning how to manipulate the model in their favor.

Q: When you joined the project, were the definitions related to the model's objectives and the system architecture passed on to you?

A: No, I became aware of them during the project. I would ask questions to the data science team when I had doubts. There was no formal passage of knowledge but instead explanations on demand.

Q: Do you perceive that as a difficulty?

A: Yes. When we met with client representatives and data scientists to map the data required by the model, I was unsure if the data we requested was correct, since I did not know what the data scientists expected for the model input.

In these meetings, I noticed a mismatch regarding the participants' understanding of the data. For example, I was expecting them to be in a particular format. Yet, the data scientists were expecting them in another form, and what the client representatives understood differed from what the data scientists were expecting. This situation provoked changes in the model consumption API throughout the system's development.

Q: Did you participate in discussions about obtaining the data for the model?

A: I was not involved in this discussion. As far as I know, data were not obtained through the cloud or online access. The data scientists received files from the client representatives for training and validating the model. Still, the data were placed in a secure area since it was sensitive.

Q: Did you participate in any data-related activity?

A: No, everything was in the hands of the data science team.

Q: Did you participate in the definition of how the system would consume the model?

A: I developed the services that use the model consumption API. I do not know if it is easy to change the model.

Q: How was the discussion about the storage of the model artifacts?

A: I did not participate in that conversation. I think all artifacts are in a Git repository.

Q: Was any data scientist involved in the publication of the model?

A: No, we are responsible for deploying the model consumption API. The deployment of this service, as well as the other system services, is automated through a CI/CD pipeline.

Q: Was there any discussion about model updates and implementing incremental learning?

A: We discussed this, it was a feature raised by client representatives, but the discussion did not go very far. For now, we initially defined that retraining the model with new data would be a manual task. The data scientists also participated in these discussions.

Q: How was the discussion regarding model monitoring?

A: I know the data science team will send model results to client representatives to assess if they are as expected, which is also done manually. We are not in charge of that. However, if there is a problem with the infrastructure of any service, the responsibility falls to our team.

Q: Were security and privacy issues considered when building the system?

A: We tried to protect communication with external APIs and services as much as possible, using JWT tokens that expire after some time. Private data consumed by internal services are persisted in databases and are not exposed to other users.

Q: How was the relationship with the data scientists during the integration of the model with the rest of the system?

A: We were very separated, and I did not like that. We did not know much about the model. It was like a "black box" that we did not get involved much. That's how it happened, and I do not know if it was supposed to be like that. Even with a well-defined API, things that were obvious to the data science team were unclear to us. We did not participate in creating the model. We only developed the services that consumed it, so we did not know what was being done. This proved to be a problem when we met with the client to define the data we needed from their external APIs.

This separation got in the way of integrating the model with the rest of the system. I did not have the necessary knowledge to analyze if the data was correct and what fields were required or optional. Problems only appeared when we started testing and integrating the system with the client's external APIs. Only then did we notice data either missing or in the wrong format. If the teams were not so distant, we could have anticipated these problems.

Q: Is the integration of the model with the rest of the system well-documented?

A: No, it is not documented well enough. We currently have the model's output and input data documented. But, for example, in the middle of this integration, there is a mapper that converts data to the format expected by the model. We could have documented this conversion better.

B.4 Transcription of SE2's interview

Q: Did you participate in any discussion about user interactions with the model?

A: Initially, we discussed what data we would capture from the user before discussing the model. There was no discussion with the whole team but partial discussions with business stakeholders, data scientists, among other members. At that stage, we were still defining product concepts, whose meanings differed for each actor.

We had several discussions with client representatives to understand their product vision and define what was and was not possible. From there, the UX designers started to prototype ideas we used to model the system database. Not all software engineers participated in these meetings, as having all of them participating could have affected our productivity.

Q: Did you notice any difficulties during the discussions with the data science team?

A: I did not experience any difficulties. Since the data scientists were part of a separate team, their activities were like a "black box" that we did not need to care about. We only asked them what data they needed, and then we included a new field in the UI so the user could fill it in. I know that internally they did a more detailed analysis of the data. But I was not a part of those discussions, as I was from another team.

Q: Did you participate in discussions about obtaining the data for the model?

A: We only had a few discussions about this because we mainly depended on how the client representatives could provide us with the data. So we just obtained the data following the methodology they defined and adapted it to how the data scientists desired. I do not know if this was the ideal way to get the data, but it was how we handled it in our scenario. Our documentation of this process consists of emails we could retrieve in case of an audit.

The data scientists defined what data was needed, but our team took full responsibility for obtaining the data. We discussed this subject with them to a certain extent, but since this process required skills they did not have, it was easier for us to assume this responsibility.

Q: Did you participate in any data-related activity?

A: We only captured the data needed to train the model.

Q: Did you participate in the definition of how the system would consume the model?

A: At the beginning of the project, we discussed this with the data science team. We created a REST API to allow the model integration with the system. We defined a communication interface for the API, and then each team did its part. It was outside the data science team's interest to understand how we stored the data as long as this service existed. So this responsibility was left to us, and we discussed it only among ourselves.

Q: How was the discussion about the storage of the model artifacts?

A: We did not have that discussion, which led to problems. We provided Git repositories for this storage, but the teams did not discuss how the data scientists would store the artifacts. This eventually caused issues because the model had a lot of artifacts, such as the training scripts, which were not separated from the API code. For this reason, large files were loaded unnecessarily every time a new model release was generated.

Q: Was any data scientist involved in the publication of the model?

A: No. We already had a pattern for deployment beforehand, and we knew the data scientists did not specialize in DevOps, so we left this structure ready for them. All the infrastructure concerning the model is our responsibility.

I also participated in the selection of machines for model training. The training infrastructure is heavy and demands good machines. I did not participate in the definitions of how the training would work.

Q: Was there any discussion about model updates and implementing incremental learning?

A: I did not participate in any discussion on this subject. I do not know what was decided.

Q: How was the discussion regarding model monitoring?

A: The responsibility for model performance lies with the data science team. The services created to consume the model are our responsibility.

Q: Were security and privacy issues considered when building the system?

A: Yes. The system's backend is the one that communicates with the model, which improves security. We had several discussions about this topic when designing the product, as it is available online.

Q: How was the relationship with the data scientists during the integration of the model with the rest of the system?

A: It was natural. We defined the data needed in the discussions between the teams. We then defined an API from there, and each team followed its side.

Still, we had some problems with changes in the communication interface established for the API, and also with the timing of these changes. But I think these issues are more related to managing changes in requirements and tasks than the teams' separation. If both teams were closer, we could have avoided these problems. However, we would also have had a higher overhead, as everyone would need to be together in all meetings.

Q: Do you think actors linked to requirements management could have participated more?

A: Yes, this participation is very important, especially when you are delivering a product to a client. System definitions are expected to change throughout the project, so it is important to manage these changes effectively. The client representatives' expectations and what was passed on to the team must be clear.

Q: Is the integration of the model with the rest of the system well-documented?

A: Yes, I think so. Our biggest challenge was regarding the changes. The system's initial state was well-documented, but then changes started happening. These changes were not documented properly, which harmed the alignment between the teams. We did not correctly update the documentation throughout the project, and we also did not communicate these changes efficiently. We discovered them as system components stopped working.

C
Interview Transcripts - Focus Group

This appendix comprises the revised transcripts we analyzed during the focus group sessions. All transcripts were translated from Portuguese to English.

C.1
Transcription of the First Focus Group Session

C.1.1
First Stage: Task Assessment

SE1: I will briefly describe my experience with AI systems and models. As a software engineer, I am interested, for example, in how I will store the model. Planning model storage is a task in which software engineers should participate. The model can have multiple artifacts, some of which may be enormous, so they must be stored accordingly. Evaluating and selecting data is not much the software engineer's job, it is much more the data scientist's job. We start to participate more when storing the model and making it available. At least, that was my experience. In some of the activities, we can help the data scientist. For example, I imagine we could provide a spreadsheet with the data. So we are not necessarily working on the system, we are simply helping the data scientists organize themselves. Sometimes we end up helping them because we are friends.

DS1: I think that data access definition should involve the software engineers. Software engineers usually have a greater knowledge of the company's APIs for capturing data, so they should be able to help the data scientists navigate the available infrastructure better. However, this also depends on the role of the data scientist. For example, there are teams where the data scientist is almost a database administrator as well. In this case, a software engineer may not be needed.

DS2: Considering that data may not be in the hands of the data scientist, a software engineer can help understand the best way to access it, when to access it, etc.

DS1: When it comes to ML model evaluation, the data scientists know what metrics are relevant. They will know, for example, if the model is overfitted. If you ask software engineers to deploy an overfitted model, they will probably do it, but only a data scientist will realize that the model has a problem and is not ready for production. Hence, I do not know how a software engineer could help in this process.

SE1: I agree. In my previous work experiences, the data scientists were responsible for evaluating the model and selecting data. In my current project, I have no idea how our ML model was evaluated or how data was selected. In fact, I was never even curious to know. Still, since we have a good relationship with the data scientists, we are always willing to help them if they need it.

DS2: I noticed software engineers are usually not very interested in the research process of a data science application. I have made presentations showcasing the algorithms examined to build a model, or the metrics used for its evaluation, and their attention goes away very quickly. This may be because they do not know how to collaborate much at this stage of research.

DS1: For infrastructure-related tasks, the data scientist will probably not know how to deploy the model, but the software engineer will. The actors in charge of CI/CD operations are usually software engineers. However, this does not mean they know how to execute the model. For this reason, it is vital that a data scientist gets together with them to explain the model and how to run it. Regarding deployment, I believe both actors need to be in sync to avoid any problems.

DS2: The same goes for ML model integration tasks. These tasks will define the ML model's output for other system components. This should be discussed between the software engineers and the data scientists, who must evaluate this according to the system's goals.

C.1.2
Second Stage: Recommendation Assessment

Involve data scientists and business owners when eliciting and analyzing requirements

SE1: The business owner knows the data very well. A business owner usually knows how to access the data, whether it is from other systems or from a spreadsheet. I think this interaction is important, especially when defining how data will be acquired and selected. I think the business owner has a lot of knowledge. Sometimes they will know who possesses that hidden spreadsheet that you will need.

DS1: Business owners understand a lot about the data and can help with model evaluation. For instance, sometimes you may think the model has to avoid false positives, but they might say, 'No, my problem is with false negatives,' so then you will have to choose another metric.

DS2: Involving the business owners can help the actors understand what the data represents. Sometimes, you know the name of a given variable and whether it is numeric or categorical, but you may not understand what it represents. For example, I have worked on projects where the data came from another company. In these cases, the business owners had to explain what each data field represented, and this improved the interaction between us and the software engineers.

DS1: Having the business owners close to the software engineers and data scientists during requirements analysis can help enhance their collaboration. For instance, imagine a scenario where the business owner asks for a given accuracy and latency. The data scientist knows how to achieve that accuracy, but may not know if that latency is possible given the available machines. This is something a software engineer can help with.

Provide ML literacy for all project stakeholders

DS1: During data access definition, I do not think knowing the difference between classification and regression would be helpful for acquiring the data.

SE1: This is what I thought. I have been working in an academic environment for a long time. Even though I am not in the AI field, I constantly see lectures and learn about this topic, so I did not require this literacy. Having said that, I do not think this theoretical knowledge is important for data access definition. I think practical instructions, such as how to access a spreadsheet or another system, are more efficient.

DS1: I thought about disagreeing with the following task as well. However, if software engineers happen to know a bit more about data science, they may be able to help with data selection. For example, they may discover noise in the data capable of hindering model training, or notice that a data column has many null values. Nevertheless, I disagree with the statement for defining data access or evaluating the model. These tasks should not induce collaboration. Moreover, I think this recommendation could be interesting for model storage. Comprehending how the models are generated allows you to think about how to persist them more intelligently. For example, suppose there is a high chance of data drifts in the project you are working on. In that case, the software engineer helping with ML artifact storage would know that the model needs to be trained and updated often. This would determine the development of an effective versioning system.

DS2: Yes, depending on the model's characteristics, a software engineer's knowledge could be interesting. Especially when updating the model, I think it would help.

DS1: I feel the same way for the subsequent tasks. ML literacy for software engineers greatly helps in these tasks. They will know, for example, what type of approach to adopt when deploying the model. Depending on the type of model the data scientist has created, the software engineer will have an idea of the computational power required to run it.

SE1: When I voted, I thought this could be the data scientist's responsibility. For example, the data scientist would come and say what the infrastructure requirements are. Then, the software engineer would not need to know anything about AI.

DS1: That is true, their interaction is also important. However, with ML literacy, the software engineer will be able to notice this even if the data scientist forgets to warn the team. This avoids problems in the future, such as having a deployed model with very high latency.

DS2: ML literacy can help with the definition of the model's output and how it will be consumed. It makes communication between the actors easier when specifying the best way to interact with the model.

Develop documentation for product requirements, system architecture, and APIs at collaboration points

SE1: I think documentation always helps, especially for ML model availability. When it comes to data selection, I almost completely disagreed, but I think a requirements documentation may be helpful. However, I have the vision of a software engineer, not a data scientist.

DS2: This recommendation is very relevant for ML model integration and deployment. It is also very useful when defining the model's API.

DS1: I agreed with the statement for most of the tasks. Well-defined product requirements documentation can greatly help with model availability and integration. It can even help with model storage.

SE1: The system architecture documentation mentioned in the recommendation is essential for these tasks. Having that there influenced my votes.

DS1: Yes, exactly. Having well-produced documentation greatly impacts development. For example, if I have a requirement for extremely low latency, it may affect how the model will be developed and made available. The fact that this is documented explicitly assists in the collaboration between the software engineer and the data scientist. For instance, if the data scientist comes up with an extremely slow approach, it will be clear to everyone that the model is not ready for production.

Define clear responsibilities and internal processes with clear boundaries for data scientists and software engineers

SE1: I partially disagreed with the statement for ML model evaluation because it is very clear to me that this is a data scientist's responsibility.

DS1: I agreed because, in my opinion, this recommendation facilitates task division. The data scientist's role can often get confused with other roles. As I mentioned before, they are sometimes also considered database administrators. In addition, they may be confused with software engineers and expected to handle model deployment solely. When you clearly define each role's responsibility, communication becomes easier. This is why I agreed with the statement for all tasks.

DS2: Even if a data scientist will need to perform tasks usually carried out by a software engineer, this needs to be clear for everyone. Clarifying boundaries helps a lot in communication, especially when evaluating the work we need to do. From the point of view of system architecture, these definitions help us visualize who will be responsible for each system component.

SE1: I understand. Still, I maintain my opinion regarding model evaluation. In the teams I have worked in, explicitly defining responsibilities and boundaries was never necessary during this task. Both data scientists and software engineers already knew what was expected from them without the need for a formal explanation. I believe this task should be mostly carried out by data scientists, as there is no need for software engineers to be involved This may be because I have worked majorly in academic environments.

Moderator: What are the differences related to working in an academic environment?

DS1: In a research environment, roles can be mixed up. For example, I might be in a team where I am the data scientist, and there is someone else who is the software engineer. Because we are in a research environment, the software engineer may also be conducting data science research. This allows them to contribute to the work that I am doing as a data scientist. This is uncommon in industry teams, where roles are more strictly defined. In a research environment, there is a greater knowledge exchange.

Support interdisciplinarity between data scientists and software

engineers

SE1: Having another person beside you while working is always useful, especially for catching something you missed. For instance, during data selection, the software engineer might look at the data selected by the data scientist and notice a field that could be included.

DS1: For data access definition, data selection, model evaluation, and model deployment, I think interdisciplinarity is interesting but not vital. I believe a lack of knowledge exchange between the actors during these tasks would not threaten the project. On the other hand, I consider this recommendation important for the other tasks. For artifact storage, for example, both actors need to know how the model was developed and what should be versioned. The same goes for model availability and integration: how to consume the model and its inputs and outputs must be clear to everyone.

DS2: Data scientists are usually more interested in research and generating insights through data analysis, so they may not know how a solution can actually be operationalized and sustained over time. Before deploying to production, it is important to understand what infrastructure is available, how the model will be updated, and how data will be continuously obtained. I think this kind of knowledge is more related to software engineering. As data scientists become familiarized with these concepts, their communication with software engineers will improve.

SE1: I agree. If the data scientists know how model deployment works, they can design the ML model with this process in mind. This would make the deployment task less arduous, especially if the project is still in its initial phase.

DS2: After the system is deployed and operational, the actor responsible for maintaining it is usually a software engineer, not a data scientist. This recommendation motivates the software engineer to know more about model characteristics, such as if it is a separate service or how it is consumed. This knowledge would be very helpful, for example, if the software engineer notices an opportunity to implement incremental learning.

Organize regular meetings for showcasing team activities

SE1: When defining data access and selecting data, I think the meetings would help. For the other tasks, I consider regular meetings useful, but not critical. After the team has agreed on all definitions, an occasional conversation between members should be enough. Even if they are not that frequent, they would be enough to ask questions and answer doubts.

DS1: I love daily meetings because they bring team members from different areas together to witness the whole project's evolution. For tasks that require a greater synergy between software engineers and data scientists, I agree that there should be regular meetings to keep everyone up to date. However, for tasks where I do not see such a strong synergy, I believe these meetings are nice to have so that everyone knows what is happening, but they are not crucial. In the case of data selection, for example, having a meeting to discuss this task would be good for the software engineers to be more informed about what is happening, but it is not essential for their work. The same is true for data access definition: showcasing how data is obtained might be interesting for data scientists, but they do not have to know where the data comes from as long as they have access to it. In these scenarios, I feel the team's progress would not be compromised if these meetings did not happen.

DS2: I agreed with the statement because of the team alignment generated by this recommendation. The team must know what is being done and how this impacts the tasks.

C.2 Transcription of the Second Focus Group Session

C.2.1 First Stage: Task Assessment

DS3: In the case of defining data access, I think this connection with the software engineer is going to be essential. After the data scientist has the data, it will be possible to work on it, train the model, perform feature engineering, etc.

DS5: I agree, I think the scientist should have this conversation with the software engineer to find out how to access the data.

DS3: I partially disagreed with the statement for data selection. The role of a data scientist consists of analyzing the data and performing feature engineering to train the model. For these activities, I would not involve a software engineer. On the other hand, during data selection, it is possible to deal with incomplete data that you might have to discard or adjust. You may even discover the existence of more data that you still need to acquire. This can lead to a change in how data is being accessed, which may provoke an interaction with a software engineer. However, I believe the data selection process itself falls directly to the data scientist.

DS4: That is exactly why I partially disagreed. Sometimes you have to go back to the data access definition to get data not initially considered.

DS5: I think this is a very exceptional case.

SE2: In the project I am currently working on, we use a collection of documents to train the ML model. These documents were sent to us by customer representatives who are not software engineers, so there was no interaction between these roles. This is why I only partially agreed with the statement for this task.

DS3: This makes total sense. In some cases, data scientists may receive data through spreadsheets. At some point, someone had to export or make them available. It may not have been specifically a software engineer, but someone who knows the subject. I have seen several projects where the data scientist is also the software engineer.

SE2: I do not consider the interaction between software engineers and data scientists very relevant for ML model storage.

DS5: I have a different opinion. It is important that both actors define where and how this storage will occur. This interaction with the software engineers allows the data scientists to understand what infrastructure is currently used for storage, as they are usually not involved in this process.

DS3: I partially agreed in this case. Throughout development, it is possible that team members change, which may harm the improvement of existing models. This can be mitigated by having an infrastructure where artifacts can be properly managed with adequate versioning, which should be created through the actors' collaboration.

C.2.2 Second Stage: Recommendation Assessment

Involve data scientists and business owners when eliciting and analyzing requirements

DS5: It is important to have the business owners during data access definition so that they can explain the data.

DS3: I do not consider this recommendation relevant for collaboration during data selection. I do not think the interaction between data scientists and software engineers is important during this task. The presence of other actors, such as business owners, is more important than this interaction.

DS4: I did not agree with the tasks related to deployment and integration I do not think this recommendation would be that interesting for these tasks. For example, I consider model integration more of an implementation task, just like with deployment, so perhaps the business owners might not be needed. It depends on whether there will be a testing phase for their acceptance of what was developed.

DS3: In my view, any prioritization or decision-making often involves business owners. In some cases, the integration task may involve different teams, and the business owner will be able to facilitate this coordination. I had doubts concerning this recommendation for ML model evaluation, but I decided to partially agree. Everyone must comprehend what exactly will be evaluated. Performance can be evaluated not only in terms of accuracy and other metrics but also in terms of computational performance. There is no point in having a super complex model if it will require a machine with tons of computational power that will not be available. These definitions can sometimes be made together with business owners and software engineers.

Provide ML literacy for all project stakeholders

DS3: I consider literacy relevant, as knowing at least the basics is important for communication. The only exception I can see is during data selection because I think the participation of the software engineer is reduced. I partially agreed on the other tasks because, in the worst case, everyone has to know that a model will be executed, that an output of a certain type will be generated, etc.

SE2: This recommendation is important to improve communication between the software engineer and the data scientist. It is vital to establish a common terminology so that communication flows more easily.

DS3: As I mentioned in the discussion about the tasks, it is a good idea to define the best way to store the model together with the software engineers to make it more easily accessible in the future. For this to be possible, they must understand what model training is and what is actually being stored.

DS4: What you said made me think, but I personally do not see much relevance in ML literacy for improving collaboration during ML artifact storage.

Develop documentation for product requirements, system architecture, and APIs at collaboration points

DS3: I do not think this recommendation is relevant for data selection because I do not see the need for much interaction between software engineers and data scientists during this task. For ML model evaluation, I think this interaction exists, but it is not strong, so I partially agreed with the statement.

DS4: The model will be evaluated based on the documented requirements, so I think this recommendation has a lot of influence on this process. Even if you do not need a lot of interaction between the software engineer and the data scientist in this activity, I believe documentation will be relevant for any interaction that happens.

SE2: Particularly for ML model integration, I find this recommendation very beneficial for collaboration.

DS5: I agree. Especially when defining APIs, this documentation is important for validating what will be done with the software engineers.

Define clear responsibilities and internal processes with clear boundaries for data scientists and software engineers

DS3: This recommendation is important regardless of how much interaction occurs during each task. Even if there is an activity where there is not supposed to be any collaboration between a software engineer and a data scientist, this must be clear for everyone to avoid someone doing something that is not their responsibility. This recommendation will help define what interactions will happen during the project.

DS5: Even for tasks with reduced interaction, following this recommendation guarantees everyone knows their role and what they must do.

Support interdisciplinarity between data scientists and software engineers

DS4: Knowing about other fields is always beneficial, but I do not consider this essential for any activity.

DS5: During data access definition, having both actors working closely may speed up the process. This recommendation can also help during model integration, as this task requires a lot of collaboration between data scientists and software engineers.

DS4: I think sharing knowledge is always valuable, but I do not consider this recommendation relevant for collaboration during the tasks.

Organize regular meetings for showcasing team activities

DS3: The interaction between software engineers and data scientists will be greatly facilitated if you know exactly what each person is doing and what is happening in the project.

Moderator: Do you see any other advantages brought by this recommendation?

DS3: This recommendation is valid for monitoring the team and exchanging knowledge, as you can even schedule technical meetings if needed. At the very least, these regular meetings to find out how things are going and what is being done already help a lot.