

Thesis Proposal

Facilitating Collaboration in Building Machine Learning Products

Nadia Nahar

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University

Thesis Committee:

Christian Kästner, Chair
James D. Herbsleb
Claire Le Gouse
Kenneth Holstein
Samir Passi

*Submitted in partial fulfillment of the requirements for the degree of Doctor
of Philosophy in Software Engineering*

May 2024

Abstract

In this proposal, I investigate the collaboration challenges between software engineers and data scientists in building machine learning (ML) products, and propose interventions to facilitate their collaboration by bridging the identified *knowledge boundaries*.

Despite significant advancements in ML algorithms and model development, integrating ML models into operational products remains challenging, with collaboration issues frequently cited as one of the major challenges. I identify collaboration challenges, and triangulate them with existing domain knowledge through a qualitative interview study with industry practitioners and a comprehensive meta-summary study of academic literature. I demonstrate principles or ideas of how those collaboration problems can be solved, illustrated with three interventions: (a) a novel approach for supporting data scientists and software engineers in deriving actionable model requirements, which aims to bridge gaps during the requirements elicitation process, (b) an innovative method to engage practitioners in responsible AI practices, fostering a culture of ethical awareness and compliance, and (c) a policy for guiding the development of explainable AI, ensuring transparency and understandability of ML models within products. These interventions are designed to address the *syntactic*, *semantic*, and *pragmatic knowledge boundaries* that hinder effective teamwork in ML product development. Lastly, I compile a comprehensive dataset of ML products from GitHub to further support research and education in the domain. The methodological approach combines various research techniques tailored to address the specific research questions in each study.

By systematically identifying and addressing collaboration challenges among practitioners, this proposal aims to support the successful development and deployment of ML products in real-world settings.

Table of contents

List of figures	iv
List of tables	v
1 Introduction	1
1.1 Thesis Statement	3
1.2 Organization and Contributions	3
2 Background	7
2.1 Machine Learning (ML) Products	7
2.1.1 ML Models vs ML Products	7
2.1.2 How ML Challenges Traditional Software Development	9
2.1.3 Qualities of Concern	10
2.2 Collaboration	11
2.2.1 Knowledge Boundaries	11
2.2.2 Past Collaboration Success Stories: DevOps and MLOps	12
2.2.3 Collaboration with ML	13
3 Identifying Challenges	15
3.1 Identification A: Interview Study of Collaboration Challenges in Building ML Products	15
3.1.1 Completed Work: Research Design	16
3.1.2 Findings for the Requirements Collaboration Point	18
3.1.3 Discussion	21
3.2 Identification B: Meta-Summary of Challenges in Building ML Products . .	23
3.2.1 Completed Work: Research Design	24
3.2.2 Findings	28
3.2.3 Discussion	32
3.3 Summary	33

4	Designing Interventions	35
4.1	Intervention A: Supporting Elicitation of Model Requirements	38
4.1.1	Problem Scoping	39
4.1.2	Proposed Work: Research Design	39
4.2	Intervention B: Encouraging Engagement in Responsible AI (RAI)	43
4.2.1	Problem Scoping and Related Work	44
4.2.2	Proposed Work: Research Design	45
4.3	Intervention C: Guiding to Satisfy Explainable AI (XAI) Requirements	51
4.3.1	Related Work	52
4.3.2	Ongoing Work: Research Design	52
4.4	Summary	57
5	Setting Foundation for Future Research and Education	59
5.1	Completed Work: Research Design for Curating the Dataset	61
5.1.1	Search Space and Scope	62
5.1.2	Search Pipeline	62
5.1.3	Limitations and Threats to Validity	64
5.2	The Open-Source ML Product Dataset	64
5.3	Findings from a Qualitative and Quantitative Analysis	65
6	Conclusion and Proposed Timeline	66
	References	67

List of figures

1.1	Overview of Thesis Contributions	2
2.1	Example of an ML Component within an ML Product	8
3.1	Structure of Two Interviewed Organizations	16
3.2	Identification A: Research Design	17
3.3	Identification B: Research Design	24
4.1	Targetted Problems Across Different Knowledge Boundaries	36
4.2	Interventions Facilitating Collaboration to Overcome Knowledge Boundaries	37
4.3	Intervention A: Proposed Approach	40
4.4	MLTE User Interface	41
4.5	Prompt Engineering Pipeline for Story Generation	46
4.6	Story Generation Matrix and an Example Story	48
4.7	Iterative and Collaborative Policy Design Process	53
5.1	Overall Process of ML Product Mining in GitHub	62
6.1	Thesis Timeline	66

List of tables

3.1	Overview of Identified Challenges	25
4.1	Summary of the Observations on Policy Design Exercise	55
4.2	Policy Draft from Week Seven and Notes Highlighting Improvements over Prior Drafts	56
5.1	Sample ML Products for Analysis, from the Curated Dataset of 262 ML Products: Mobile (P1-P10), Desktop (P11-P20), and Web Applications (P21- P30)	60
5.2	Number of Retrieved Projects after Each Step	62

Chapter 1

Introduction

In this proposal, I study the collaboration challenges between software engineers and data scientists in building machine learning (ML) products, and I propose interventions to facilitate collaboration between them using the concepts of boundary objects. Despite significant advancements in machine learning and model development, building products with ML components is still identified as challenging, with collaboration issues as one of the major obstacles. Through qualitative interviews with practitioners, I uncover the specific collaboration challenges they face. Additionally, I conduct a qualitative meta-summary study of the academic literature to collate the collective knowledge in this area. To support further research and education, I also compile a dataset of ML products from GitHub and conduct a preliminary analysis. Finally, I propose three interventions to facilitate collaboration: a novel approach to support data scientists and software engineers in deriving model requirements, a novel approach to engage practitioners in responsible AI, and a policy for guiding practitioners to build ML products with explainable AI¹. Throughout this work, I deliberately select and combine various research methods tailored to the research questions in each study.

The challenges of building products with ML components, which I refer to as *ML products* in this proposal, are well-known and broadly discussed. Even though, in recent years, we have made remarkable progress in both developing models (e.g., object detection and content generation) and building products with such models (e.g., autonomous delivery robots and cancer prognosis tools), practitioners still report struggling with the integration of models to real-world products [119, 4, 98]. In fact, according to Gartner, an estimated 85% of machine learning projects fail to transition from prototype to production [39, 154, 153, 37]. All these anecdotal claims probe the researchers to systematically study the challenges in this domain as well as design interventions to support the development of ML products.

¹This proposal is specifically targeted towards machine learning (ML), a subfield of AI, but I also use the term “AI” in case of established terminologies such as Responsible AI and Explainable AI.



Fig. 1.1 Overview of Thesis Contributions

Thinking about the product beyond the model. A machine learning model, on its own, cannot serve users, without being incorporated into a software product. While model development is already challenging in itself, building an operational product with the model is a much larger and complex process. This process involves many crucial steps, such as collecting the right kind of data, designing and developing the non-ML components of the software product that seamlessly integrates the ML model, implementing safeguards to mitigate inevitable model mistakes, deploying and scaling the model to meet real-world demands, continuously monitoring and updating the model and product to keep them fit for purpose, securing user data against breaches, ensuring regulatory compliance, and considering ethical and safety implications of model decisions to prevent adverse effects in real situations. The complexity and diversity of these tasks go beyond the responsibilities and expertise of data scientists and necessitate a collaborative effort of many other stakeholders such as software engineers, operators, and project managers. Software engineers, in particular, play a critical role in building the infrastructure that supports the model, integrating the model within the product, adding safety features to counter any model errors, and scaling and maintaining the model and the product under real-world conditions. This calls for a close collaboration between data scientists, software engineers, and other relevant stakeholders to effectively build and deploy appropriate machine learning solutions in real-world settings.

Collaboration is hard in ML product development. Collaboration in general is challenging, especially across disciplines among team members with different educational backgrounds and experiences [28, 34], which is true for building ML products as well. Historically, researchers have made significant progress in enhancing collaboration in traditional software development, with practices such as DevOps [29, 60] and the Security Development Lifecycle (SDL) [138, 35] to improve teamwork and establish a culture of collaboration between developers, operations teams, and security experts. Yet, the development of ML products introduces additional hurdles for teamwork, due to the nature and novelty of ML and the processes involved in its integration, deployment, and maintenance [149, 101], as

I will explore. The iterative, experimental, and data-driven approach to ML development requires continuous communication and adaptability. It also raises unique syntactic, semantic and pragmatic knowledge boundaries [22, 23] and related challenges, particularly between software engineers, who are used to more predictable workflows, and data scientists, who are engaged in the exploratory development of models. These differences may result in difficulties in understanding each other’s perspectives, and work processes, and in integrating their efforts, as I will show. Furthermore, ML heavily depends on the quality and quantity of data, necessitating close collaboration between data scientists, who train models with the data, domain experts, who understand the data, and data engineers, who manage and preprocess the data. Additionally, ML models are not simply set up once and left alone; they need to be continually updated to accommodate new data or to handle model drift, requiring ongoing collaboration between ML engineers and operations teams. While some progress has been made in addressing these issues, such as the introduction of MLOps [58] to facilitate effective teamwork between the data scientists and operations teams, the collaboration between data scientists and software engineers, who are the main stakeholders in developing the ML product, still needs attention.

This leads to my proposal, where I am particularly interested in facilitating the collaboration between data scientists and software engineers in building machine learning products.

1.1 Thesis Statement


Building a **machine-learning product** that integrates one or more machine-learning models and serves the needs of the end-users and other stakeholders, relies heavily on effective **collaboration** among team members with different backgrounds.


In this thesis, (a) I *identify collaboration points* and corresponding *challenges* primarily between *software engineers, data scientists*, and other members of the development team. Then, (b) I *propose interventions* to promote effective collaboration among these development team members.

1.2 Organization and Contributions


My research contributions are organized into three primary thrusts, as depicted in Figure 1.1. In the first thrust, I identify the collaboration challenges in building ML products. Then based on the identified challenges, in the second thrust, I design interventions to facilitate

better collaboration. In the last thrust, I work on developing infrastructure to support future research and education.




I have conducted two studies in the first thrust (Chapter 3, denoted by ) to identify the collaboration points and the challenges reported by the industry practitioners and examined within the academic literature. The findings from these studies highlight that many challenges arise at the interface or boundary between teams or distinct roles. We report the three major collaboration points that practitioners consistently report as challenging. We also triangulate the challenges with findings from existing literature.

The identified challenges demonstrate problems at different *knowledge boundaries*, namely, *syntactic*, *semantic*, and *pragmatic* [23], and thus, to demonstrate how collaboration can be improved at these boundaries by *transferring*, *translating*, and *transforming* knowledge, we develop three interventions, as outlined in the second thrust (Chapter 4, indicated by ). Each of these interventions addresses fundamental gaps in communication and collaboration, as mentioned below:

- My first intervention facilitates the elicitation of actionable model requirements and filling out negotiation cards, which is a boundary object [159, 22] for model requirements agreement among the development team members. This intervention primarily aims to overcome *semantic boundaries* and the tensions in requirements elicitation [143, 105] for ML products, by bridging the gap between the model team, primarily composed of data scientists, and the product team, which typically includes software engineers and project managers. The proposed solution aims to facilitate effective negotiations between the two parties by assisting each group in identifying viable and actionable model requirements as well as avoiding unrealistic and vague requirements.
- My second intervention is tailored to encourage data scientists to engage with the principles of *responsible AI*. It addresses the *pragmatic boundary* by resolving tensions that arise when data scientists feel resistant or indifferent to the importance of ethical considerations in developing ML products. By promoting awareness and acceptance, this initiative may help align the values and practices of data scientists with broader organizational goals.
- Finally, aimed at bridging *syntactic*, *semantic*, and *pragmatic boundaries*, my third intervention involves establishing a policy for *explainable AI*. This policy serves as a boundary object that fosters common ground among data scientists, software engineers, governance people, and other relevant team members. By clarifying expectations and responsibilities, it may support all parties to have a unified approach toward ensuring explainability in ML products.

In the third and final thrust of my research, I focus on infrastructure development (Chapter 5, )², intending to address the challenge of lack of access to ML products for academic research. To provide a resource that can be leveraged for both research and educational purposes, I offer a dataset of 262 open-source ML products and report initial findings and implications.

The summary of these contributions is as follows:

-  **A list of key collaboration challenges from industry practitioners (Section 3.1).** Through a qualitative interview study with 45 practitioners from 28 organizations, we² identify key collaboration challenges that teams face when building and deploying ML products. We report on common collaboration points in ML product development for requirements, data, and model-product integration, as well as corresponding challenges based on the different team structures and workflow.
-  **A catalog of the challenges of developing ML products from academic literature (Section 3.2).** We conduct a meta-summary study by reviewing 50 academic papers to understand different challenges in developing ML products, based on interviews and surveys with over 4,758 industry practitioners. By categorizing over 500 challenges mentioned, this study highlights the key obstacles faced, serving as a useful resource for guiding research and education in this field.
-  **A novel approach and a corresponding tool to support model requirements elicitation (Section 4.1).** We aim to advance the process of negotiating model requirements, with a focus on assessing model qualities, by leveraging the existing Machine Learning Test and Evaluation (MLTE) framework [78], and one of its components—negotiation cards. The MLTE framework serves as a roadmap for organizations, guiding them through a comprehensive evaluation of model qualities to ascertain their alignment with defined product and model requirements. Within this process, the negotiation cards act as a boundary object to document and share the agreed-upon requirements among the team members. Our objective is to assist the development team members such as data scientists and software engineers in identifying actionable and feasible model requirements specific to the product use cases. For this, we plan to integrate a generative AI pipeline as a component within the MLTE to support the data scientists in identifying relevant model requirements tailored to the specific context of the product, as well as ask the right questions to software engineers and project managers, which would help the data scientists to negotiate and reach consensus on model specifications. The pipeline will also assist software engineers in formulating realistic model requirements that align with the expected model capabilities.

²I use “we” instead of “I” in the rest of the proposal

- **🔗 A novel approach and a corresponding tool for responsible AI engagement and collaboration (Section 4.2).** To encourage data scientists and software engineers to engage with responsible AI practices and comprehend the potential harm ML products can inflict on end-users, we have devised a strategy that involves generating stories of harm using large language models (LLMs). This approach is underpinned by a long pipeline, designed to ensure the stories produced are concrete, severe, surprising, and diverse. Our next steps include an experimental evaluation of the pipeline’s effectiveness and efficiency, alongside conducting a user study to gauge its impact and usefulness.
- **🔗 Policy design insights and recommendations for explainability in ML products (Section 4.3).** To develop practical guidance on explainability for data scientists and software engineers, we, first, conduct an experimental study alongside an interdisciplinary team of AI and policy researchers to design a policy for AI explainability that is clearer, more actionable, and enforceable than existing guidelines. Moving forward, we aim to assess this policy through a large-scale controlled experiment within an educational setting. For this purpose, we have randomly assigned one of six different policy document combinations to 140 students as part of an 8-hour homework assignment. Following this, we plan to perform qualitative content analysis to evaluate the explanations and evidence submitted by the students.
- **⚙️ A dataset of open-source ML products and development insights (Section 5).** To address the limited access problem of academics in accessing commercial ML products for research purposes, we identify 262 open-source ML products on GitHub and conduct qualitative and quantitative analyses on 30 of those. Findings related to collaboration include rare explicit assignment of team responsibilities and unusually low modularity between ML and non-ML code.

Chapter 2

Background

This chapter aims to provide the foundational knowledge necessary for understanding the subsequent discussions in this proposal. We will delve into the key concepts surrounding machine learning (ML) products, including their development process and various qualities of concern. Additionally, given that our proposal concentrates on collaboration challenges, we will explore various facets of collaborative efforts within this context.

2.1 Machine Learning (ML) Products

Researchers and practitioners have gradually recognized that integrating ML into products extends beyond merely developing a model. It involves substantial effort to integrate the model into a product, while considering aspects such as system architecture, requirements, UX design, safety and security, system testing, and operations. As this proposal focuses on the development of ML products rather than just ML models or components, in this section, first, we define ML products, and then, delve into the nuances of developing ML products in comparison to traditional software products.

2.1.1 ML Models vs ML Products

While an ML model is a standalone algorithm trained on data to make predictions or decisions, an ML product encompasses the entire application that leverages this model as a component. For instance, consider an object detection model integrated into a photo gallery application (as depicted in Fig. 2.1). The development of this ML product involves not just creating the object detection model in a Jupyter notebook but also integrating it with various other components such as user interfaces for tagging objects within photos, designing the system

architecture to support real-time detection via cloud processing, and setting up data pipelines for continuous model updates.

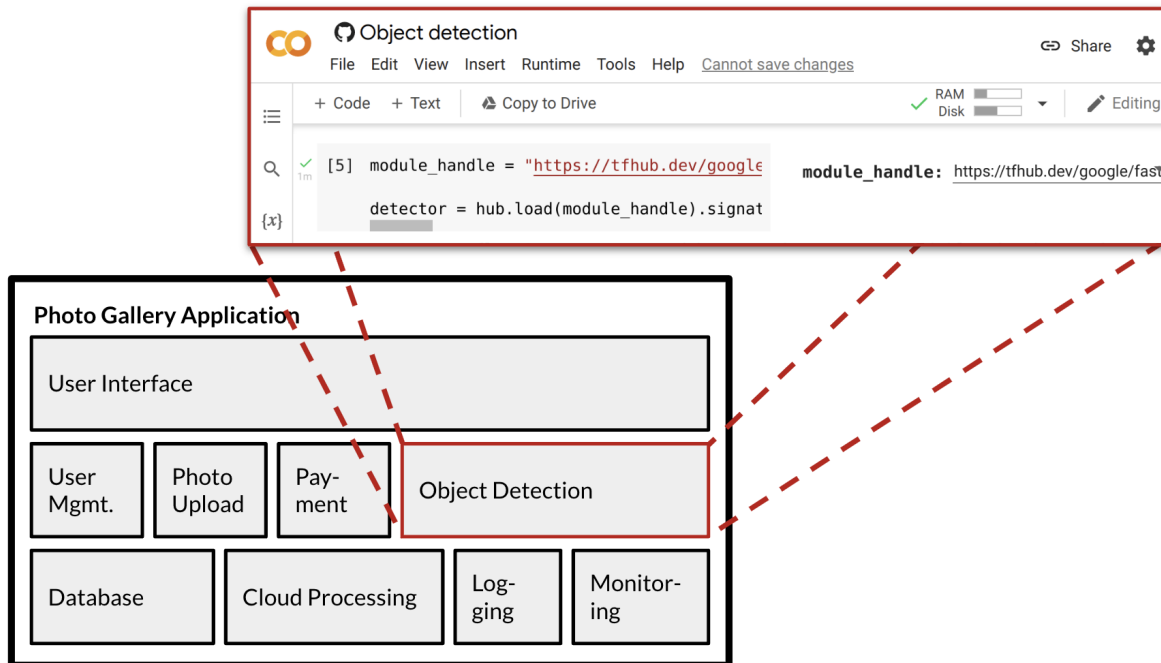


Fig. 2.1 Example of an ML Component within an ML Product

However, we observe that the terminology in this field is inconsistent, with researchers and practitioners using terms such as ML systems, ML projects, or ML applications interchangeably to refer to the libraries that train models (e.g., Tensorflow), the code to train models (e.g., in a notebook), the deployed models (e.g., GPT-3), or the products around those models (e.g., FaceSwap). Our concept of an *ML product* encompasses the entire software system, including both ML and non-ML components, in line with past research that used terms like ML-enabled systems [142, 88], or ML systems [128, 68]. Thus, to eliminate any ambiguity in this proposal, we define the term *ML product* in this section.

A key way to distinguish an ML product from other ML projects (e.g., experimental notebooks, toy projects, and libraries) is its focus on an end-user base. When an ML prototype is converted into a released product, it may be designed to be more professional and user-friendly to attract and retain end-users. Such products often feature a polished interface and comprehensive user manuals, enabling non-technical users to install and use the software without needing to execute terminal commands or library installations (e.g., `pip install ...`). Considering these essential characteristics of a software product with ML components, we define *ML product* as follows:

A machine-learning product is a software project (a) **for end-users** that (b) contains one or more **machine-learning components**.

To be considered *for end-users*, the project must have a *clear purpose* and a *clear target audience*. The purpose can be fun and the audience can be “everybody.” The software must be *complete, usable, polished, and documented* (e.g., install and usage instructions) to the level typically expected by the target audience. The product needs to use at least one machine-learned model that is used for major or minor functionality of the software. The model can be developed from scratch or called using an existing library or API.

For contrasting, we define ML library and ML project:

ML Library: Libraries, frameworks, or APIs that are used to perform ML tasks, such as TensorFlow and Scikit-learn.

ML Project: ML Project represents any software project that integrates some form of ML functionality or code. Examples include notebooks, research artifacts associated with a paper, and course homework. All ML products are ML projects, but most ML projects are not ML products.


2.1.2 How ML Challenges Traditional Software Development

Many studies have shown that incorporating ML models in software products impacts traditional software development in many different ways [149, 31]. The requirements and specifications get impacted because of the uncertainties associated with ML development, lack of AI literacy in stakeholders, difficulties in identifying business goals and metrics, accuracy vs other qualities of the product, and so on [143, 105]. Similarly, ML impacts the ways systems are architected and designed, challenging the concept of modularity, introducing additional complexity for designing data, models, and pipelines, and adding new design qualities like monitorability, and so on [125, 150, 68]. The quality assurance of ML products is also more complicated than traditional software – the concept of correctness is challenged by the accuracy metrics of the ML models, defining the model adequacy goal is difficult, online monitoring is a necessity, and so on [19, 128, 94]. Apart from mere model performance or accuracy, ML also calls for evaluating different system/product qualities such as fairness, explainability, safety, security, and so on [47, 108, 120, 81, 116, 14]. Additionally, unlike traditional software products, data is an integral part of ML components, leading

to a lot of challenges related to data quality, data accessibility, data understanding, data management, and so on [117, 102, 16, 55].

Along with all the additional technical challenges in each of the development stages, ML also raises many organizational and cultural issues. Team collaboration becomes problematic with data scientists in the teams due to differences in working style, vocabulary, code quality, and so on [101, 62, 88]. Organizational resource constraints restrict ML practitioners from achieving the model qualities they plan to provide [48, 6, 9]. The need for diverse skills in development teams requires additional training and education for both software engineers and data scientists [62, 146, 88]. The challenges and need for documentation of the model, data, and system are other aspects that many recent papers have pointed out and worked on [16, 8, 84], but we are still lacking standardization [24]. On top of all, we still do not have a good process to integrate the data science pipeline with the software development lifecycle [42, 80, 135].

To systematically study these challenges that occur during the development of ML products, as mentioned in the literature, we conduct a meta-summary study as part of our challenge identification chapter.

 Cf. §3.2
for the literature
findings

2.1.3 Qualities of Concern

For the successful development of ML products, practitioners must consider two distinct types of quality attributes: model quality and product quality. While model quality focuses on the technical performance of the ML model or algorithm such as accuracy and robustness, product quality addresses the end-to-end experience and functionality of the complete software product such as usability and interface design. The incorporation of ML in a product urges us to reconsider how we view various product qualities, such as privacy, due to its reliance on large datasets, and safety, considering the unpredictability of model behavior. These qualities are vital and demand collaborative efforts from various stakeholders. In two of the interventions in my proposal, I focus on facilitating collaboration for such product qualities: one is broadly for responsible AI (RAI), and the other is specifically for explainable AI (XAI). For readers less familiar with these areas, I offer a brief introduction to both concepts in this section.

Responsible AI (RAI). Responsible AI is a broad term, which refers to the practice of designing, developing, and deploying artificial intelligence (AI) with ethical considerations in mind [30, 148]. It encompasses ensuring that AI and ML products are fair, transparent, and accountable. This means that the ML products should be free from biases, respect privacy, demonstrate reliability, and have safeguards in place to prevent misuse. RAI is a

product quality, which involves considering the societal impacts and ensuring that the product encompassing the ML model contributes positively to the human experience rather than causing harm. In short, it is about taking a holistic approach to building AI/ML products that prioritizes ethical standards and the well-being of stakeholders affected by it.

Explainable AI (XAI). Explainable AI is a specific quality within the broader Responsible AI (RAI) framework that aims to make decisions and operations of AI/ML products transparent and comprehensible to humans [12, 1]. Its objective is to develop AI/ML products that provide clear and understandable explanations for the model predictions, decisions, or actions. This becomes particularly crucial for complex and opaque models, such as deep learning models, where it is not always evident how the model arrived at a particular conclusion.

XAI enables users and stakeholders to understand the reasoning behind the product's decisions, which can be essential in sensitive applications such as healthcare, finance, and law enforcement. Furthermore, it assists in validating that the ML product functions correctly and supports the identification and rectification of model errors. Overall, explainable AI seeks to bridge the gap between human decision-making and ML outcomes, ensuring that we can leverage the model's power while maintaining control, understanding, and ethical oversight.

2.2 Collaboration

Given that our proposal focuses on collaboration challenges, this section will explore various aspects of collaboration. We start with a brief discussion of knowledge boundaries—the limits of understanding between different individuals or groups within a collaborative setting. We will also discuss the past successes of DevOps and MLOps as models of effective collaboration from which we can draw inspiration. Finally, we will briefly contextualize collaboration specifically for the development of ML products.

2.2.1 Knowledge Boundaries

Knowledge boundaries [22, 23] refer to the division that exists when different groups or individuals have distinct bases of knowledge, which can lead to challenges in communication and collaboration. These boundaries often emerge from differences in expertise, professional background, or disciplinary focus, and can manifest as gaps in understanding or misinterpretations when exchanging information. Addressing knowledge boundaries involves developing

shared languages, creating boundary objects, and fostering environments that promote mutual learning and understanding across different knowledge domains.

There are three categories of knowledge boundary. Syntactic boundary is the most basic level of knowledge boundary that refers to the communication barriers among team members due to differences in technical language and terminologies. For instance, the term “*performance*” means *prediction accuracy* to the data scientists, but traditionally refers to *response time* for software engineers. Overcoming this boundary requires *transferring knowledge* by creating common vocabularies or standardized codes that all team members can understand.

Semantic boundary involves deeper levels of misunderstanding where team members interpret the same information differently due to varying backgrounds or expertise. For example, a developer and a designer might have different interpretations of what “user-friendly” means in the context of the software. Bridging this boundary may involve *translating knowledge* using a boundary object such as a UX mockup, which serves both as a guide for designers in visualizing user interactions and as a blueprint for developers to understand and implement the necessary functionalities.

Pragmatic boundary is the most complex boundary, which focuses on the differences in interests and values that influence decision-making or practices. At this boundary, each party might recognize the other’s differences but still not value it sufficiently to motivate change in their practices or beliefs. For instance, while developers may push for rapid delivery of new features, operators might prioritize product stability, leading to conflicting priorities. Overcoming this boundary requires *transforming knowledge* by establishing new shared practices or frameworks that respect and integrate diverse perspectives and goals. An example of such an integration is DevOps, which merges development and operations disciplines into a cohesive process with common goals and practices.

2.2.2 Past Collaboration Success Stories: DevOps and MLOps

Conflicts between teams with varying roles and goals are a commonplace issue in software development. Within the software engineering community, approaches such as DevOps and MLOps have been extensively explored to enhance teamwork between developers and operators, and similarly between data scientists and operators. These methodologies not only address the needs of tooling and operational designs but also provide a robust framework for fostering interdisciplinary collaboration, which we may take some inspiration from.

DevOps and MLOps promote a culture of collaboration where traditional conflicts between developers—who typically focus on rapid feature development—and operators—who aim for stability and cost-effectiveness—are transformed into cooperative partnerships.

By sharing tools, vocabulary, and responsibilities, these approaches successfully merge distinct workflows into a cohesive operation. For instance, development and operations teams jointly use tools such as containers for software deployment, which allows both immediate application in production environments and real-time monitoring, thus aligning their objectives.

Furthermore, these methodologies stress the significance of a shared understanding and mutual benefits, breaking down silos and encouraging a culture that values joint accountability and continuous feedback. This reformed workflow enables developers to seamlessly integrate testing, deploy faster, and adjust based on real user feedback, while operators gain efficiencies in managing infrastructure more reliably and dynamically.

However, adopting such transformative approaches is not only about adopting new tools and methods. It requires both cultural change within a company and education about these principles. Looking beyond DevOps and MLOps, these principles offer valuable lessons for interdisciplinary collaborations. Whether between data scientists and software engineers or other diverse teams, the emphasis remains on fostering joint goals and leveraging shared tools to facilitate collaboration. The narratives within DevOps and MLOps serve as a testament to what can be achieved when collaboration and mutual understanding take precedence over traditional role-defined boundaries. Such success stories provide compelling examples for other sectors and fields aiming to nurture a collaborative culture and achieve collective goals efficiently.

2.2.3 Collaboration with ML

Collaboration in ML is crucial due to the inherently interdisciplinary nature of the field, requiring expertise in data science, software engineering, and domain-specific knowledge to provide contextual understanding. In contrast to traditional software development, where projects typically revolve around well-defined requirements, project plans, and deterministic outputs, ML products often operate in environments with inherent uncertainties and probabilistic outcomes, where data quality, model selection, and hyperparameter tuning can significantly affect the results, and influence the project's progress, timeline, and overall success.

In traditional software development, teams usually follow a structured process with clear milestones and a predictable path, with well-defined stages such as planning, design, development, testing, and deployment, allowing relatively straightforward progress and timelines easier to estimate. However, in ML product development, the collaborative process is inherently more dynamic and iterative. Data scientists, software engineers, and subject matter experts must work closely together to continually adapt and refine their models based

on ongoing feedback and evolving data. This iterative experimentation process necessitates frequent adjustments and reiterations, often requiring quick pivots and problem-solving across multiple disciplines. Effective collaboration, therefore, becomes essential to integrate these diverse skills and navigate the complexities of ML products.

Chapter 3

Identifying Challenges

In this chapter, I present two key studies that lay the foundation for my proposal. The objective behind these studies was to gain a deep understanding of the obstacles to effective collaboration, which is essential to support stakeholders throughout this process of ML product development—this marks the initial phase ~~(1.1)~~ of my thesis work: “*In this thesis, (a) I identify collaboration points and corresponding challenges primarily between software engineers, data scientists, and other members of the development team*”.

In the first study, I delve into the collaboration challenges faced by industry practitioners through a qualitative interview study. This approach allowed me to gather rich, firsthand insights into the complex dynamics and challenges that professionals encounter when collaborating on ML products.

For the second study, my focus shifts to synthesizing existing knowledge in the academic field by conducting a qualitative meta-summary study. This involves analyzing existing literature to consolidate and interpret the collective knowledge regarding ML product development challenges in the field, offering a broader perspective on the issue.

3.1 Identification A: Interview Study of Collaboration Challenges in Building ML Products

This section is a strongly shortened version of the ICSE’22 paper [88]: “*Collaboration Challenges in Building ML-enabled Systems: Communication, Documentation, Engineering, and Process*” [Nahar et al., 2022]

To better understand collaboration challenges and avenues toward better practices, we conducted interviews with 45 participants contributing to the development of ML products (i.e., not pure data analytics/early prototypes). Our research question is: **What are the**

collaboration points and corresponding challenges between data scientists and software engineers? Participants come from 28 organizations, from small startups to large big tech companies, and have diverse roles in these projects, including data scientists, software engineers, and managers. During our interviews, we explored organizational structures (e.g., see Fig. 3.1), interactions of project members with different technical backgrounds, and where conflicts arise between teams.

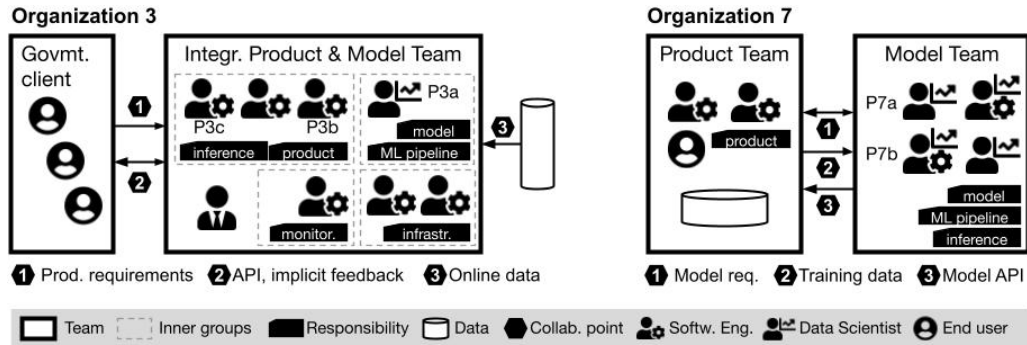


Fig. 3.1 Structure of Two Interviewed Organizations

Three *collaboration points* seemed particularly challenging: (a) identifying and decomposing requirements, (b) negotiating training data quality and quantity, and (c) integrating data science and software engineering work. To keep the proposal concise and focused, we only discuss results from the first collaboration point, particularly given that the challenges from the other collaboration point are not central to the subsequent chapters. For a more detailed understanding, we encourage interested readers to consult the full paper.

3.1.1 Completed Work: Research Design

Due to limited research on collaboration in building ML products, we adopt a qualitative research strategy to explore collaboration points and corresponding challenges, primarily with stakeholder interviews. We proceeded in three steps, as depicted in Fig. 3.2: (1) Prepared interviews based on an initial literature review, (2) conducted interviews, and (3) triangulated results with literature findings. We base our research design on Straussian Grounded Theory [133, 132], which derives research questions from literature, analyzes interviews with open and axial coding, and consults literature throughout the process.

Step 1: Scoping and interview guide. To scope our research and prepare for interviews, we looked for collaboration problems mentioned in the existing literature on software engineering for ML products. In this phase, we selected 15 papers opportunistically through keyword

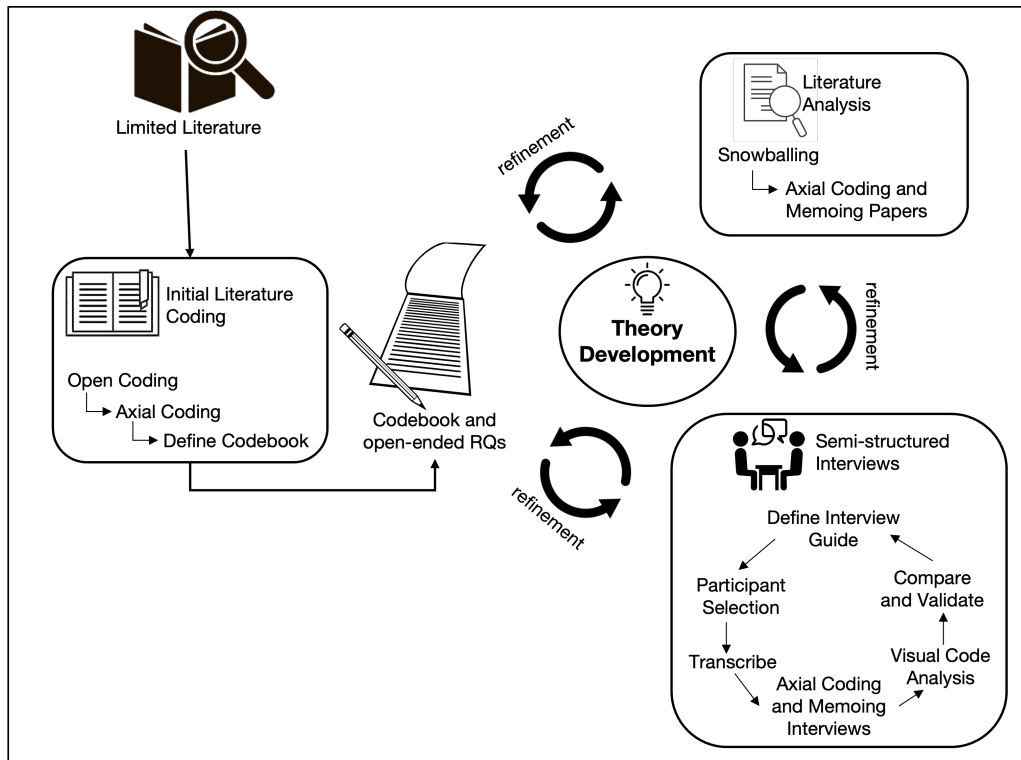


Fig. 3.2 Identification A: Research Design

search and our knowledge of the field. We marked all sections in those papers that potentially relate to collaboration challenges between team members with different skills or educational backgrounds, following a standard open coding process [133]. Even though most papers did not talk about problems in terms of collaboration, we marked discussions that may plausibly relate to collaboration, such as data quality issues between teams. We then analyzed and condensed these codes into nine initial collaboration areas and developed an initial codebook and interview guide.

Step 2: Interviews. We conducted semi-structured interviews with 45 participants from 28 organizations, usually 30 to 60 minutes long. All participants are involved in professional software projects that use ML and that are either already deployed in production or have a concrete plan for deployment.

We tried to sample participants purposefully (maximum variation sampling [44]) to cover participants in different roles, types of companies, and countries. We intentionally recruited most participants from organizations outside of big tech companies, as they represent the vast majority of projects that have recently adopted ML and often face substantially different challenges [49] (21.4% big tech, 39.3% mid-size tech, 17.9% startups, 14.3% non-IT, and

7.1% consulting). Among the 45 participants, their roles related to machine learning (51%), software engineering (20%), management (11%), and others. Where possible, we tried to interview multiple participants in different roles within the same organization to get different perspectives. For confidentiality, we refer to organizations by number and to participants by PXy where X refers to the organization number and y distinguishes participants in the same organization.

We transcribed and analyzed all interviews. Then, to map challenges to collaboration points, we created visualizations of organizational structure and responsibilities in each organization (we show two examples in Fig. 3.1) and mapped collaboration problems mentioned in the interviews to collaboration points within these visualizations. We used these visualizations to further organize our data; in particular, we explored whether collaboration problems are associated with certain types of organizational structures.

Step 3: Triangulation with literature. As we gained insights from interviews, we returned to the literature to identify related discussions and possible solutions (even if not originally framed in terms of collaboration) to triangulate our interview results. We pursued a best-effort approach that relied on keyword search for topics that surfaced in the interviews, as well as backward and forward snowballing. Out of over 300 papers read, we identified 61 as possibly relevant and coded them with the same evolving codebook.

Threats to validity and credibility. Our work exhibits the typical threats common and expected for this kind of qualitative research. Generalizations beyond the sampled participant distribution should be made with care; for example, we interviewed few managers, no dedicated data experts, and no clients. In several organizations, we were only able to interview a single person, giving us a one-sided perspective. Observations may be different in organizations in specific domains or geographic regions not well represented in our data. Self-selection of participants may influence results; for example, developers in government-related projects more frequently declined interview requests.

3.1.2 Findings for the Requirements Collaboration Point

Through our interviews we identified three central collaboration points where organizations building ML products face substantial challenges: (1) requirements and project planning, (2) training data, and (3) product-model integration. In this section, we delve into the results from the requirements collaboration point, as they are essential for understanding the content of the subsequent chapters.

In an idealized top-down process, one would first solicit product requirements and then plan and design the product by dividing work into components (ML and non-ML), deriving each component's requirements/specifications from the product requirements. In this process, the product team needs to negotiate product requirements with clients and other stakeholders, and plan and design product decomposition, negotiating with component teams the requirements for individual components (e.g., specific model requirements).

Common Development Trajectories There are distinct patterns relating to how organizations elicit requirements and decompose their systems. Most importantly, we see differences in terms of the order in which teams identify product and model requirements:

Model-first trajectory: 14 of the 28 organizations (3, 5, 10, 14–17, 19, 20, 22, 23, 25–27) focus on building the model first, and build a product around the model later. In these organizations, product requirements are usually shaped by model capabilities after the (initial) model has been created, rather than being defined upfront. In organizations with separate model and product teams, the model team typically starts the project and the product team joins later to build a product around the model.


Product-first trajectory: In 12 organizations (1, 4, 7–9, 11–13, 18, 21, 24, 28), models are built later to support an existing product. In these cases, a product often already exists and product requirements are collected for how to extend the product with new ML-supported functionality. Here, the model requirements are derived from the product requirements and often include constraints on model qualities, such as latency, memory, and explainability.

Parallel trajectory: Two organizations (2, 6) follow no clear temporal order; model and product teams work in parallel in close collaboration.


Challenges of Eliciting Product and Model Requirements We found a constant tension between product and model requirements in our interviews.

Product requirements require input from the model team (👥, 📄). While more organizations follow a model-first trajectory, organizations that follow a product-first trajectory also report requirements challenges regarding ML capabilities. A common theme in the interviews is that it is difficult to elicit product requirements without a good understanding of ML capabilities, which almost always requires involving the model team and potentially performing some initial modeling when eliciting product requirements. Regardless of whether


product requirements or model requirements are elicited first (product-first trajectories) or a model team focuses on a model first without a full understanding of the product (model-first trajectory), data scientists often mentioned unrealistic expectations of model capabilities of the model from both the client and the product development team. For example, P26a shared “*For this particular project, [the project manager] wanted to claim that we have no false positives and I was like, well, that’s not gonna work.*” Usually, the product team cannot identify product requirements alone, instead, product and model teams need to interact to explore what is achievable.

 We aim to overcome this challenge of unrealistic expectations with intervention A, §4.1

In organizations with a model-first trajectory, members of the model team sometimes engage directly with clients (and also report having to educate them about ML capabilities). However, when requirements elicitation is left to the model team, members tend to focus on requirements relevant for the model, but neglect (or do not document) requirements for the product, such as expectations for usability. O’Leary and Uchida raise similar concerns about model-centric development where product requirements are not obvious at modeling time [95].

 We also target this challenge in §4.1, intervention A

Model development without clear model requirements is common (📄). Participants from model teams frequently explain how they are expected to work independently, but are given sparse model requirements. They try to infer intentions behind them, but are constrained by having a limited understanding of the product that the model will eventually support. Especially in organizations following the model-first trajectory, model teams may receive some data and a goal to predict something with high accuracy, but no further context, e.g., P3a shared “*there isn’t always an actual spec of exactly what data they have, what data they think they’re going to have and what they want the model to do.*” Several papers similarly report projects starting with vague model goals [116, 160, 102].


 An example of semantic knowledge boundary, Cf. §2.2.1



Even in organizations following a product-first trajectory, product requirements are often not translated into clear model requirements. For example, participant P17b reports how the model team was not clear about the model’s intended target domain, and thus could not decide what data was considered in scope. The difficulty of providing clear requirements for an ML model has also been raised in the literature [62, 105, 75, 160, 145, 127]. Ashmore et al. report mapping product requirements to model requirements as an open challenge [10].


Provided model requirements rarely go beyond accuracy and data security (⚙️, 📄). Requirements given to model teams primarily relate to some notion of accuracy. Beyond accuracy, requirements for data security and privacy are common, typically imposed by


the data owner or by legal requirements. Literature also frequently discusses how privacy requirements impact and restrict ML work [14, 103, 75, 52, 76, 53].

We rarely heard of any qualities other than accuracy considered upfront. When prompted, very few of our interviewees report considerations for fairness either at the product or the model level. Similarly, no participant brought up requirements for the explainability of models. This is in stark contrast to the emphasis that fairness and explainability receive in the literature, e.g., [162, 47, 77, 14, 127, 122, 157, 21, 5, 49].

 We target this challenge in intervention B (§4.2) and C (§4.3)

Recommendations Our observations suggest that involving data scientists early when soliciting product requirements is important () and that pursuing a model-first trajectory entirely without considering product requirements is problematic (). Conversely, model requirements are rarely specific enough to allow data scientists to work in isolation without knowing the broader context of the product and interaction with the product team should likely be planned as part of the process. Requirements form a key collaboration point between product and model teams, which should be emphasized even in more distant collaboration styles (e.g., outsourced model development). Vogelsang and Borg also provide similar recommendations to consult data scientists from the beginning to help elicit requirements [144].

ML literacy for customers and product teams appears to be important (). Participants suggested conducting technical ML training sessions to educate clients; similar training is also useful for members of product teams. Several papers argue for similar training for non-technical users of ML products [56, 122, 144].

Most organizations elicit requirements only rather informally and rarely have good documentation, especially, but not only, when it comes to model requirements. It seems beneficial to adopt more formal requirements documentation for product and model (), as several participants reported that it fosters shared understanding at this collaboration point. Checklists could help to cover a broader range of model quality requirements, such as model latency, fairness, and explainability, in such requirements. Formalisms such as model cards [85] could be extended to cover more of these model requirements.

3.1.3 Discussion

Data scientists and software engineers are certainly not the first to realize that interdisciplinary collaborations are challenging and fraught with communication and cultural problems [17], yet it seems that many organizations building ML products pay little attention to fostering better interdisciplinary collaboration. Organizations differ widely in their structures and practices, and some organizations have found strategies that work for them. Yet, we find

that most organizations do not deliberately plan their structures and practices and have little insight into available choices and their tradeoffs. Based on the challenges identified in this study, we see four broad themes that benefit from more attention both in engineering practice and in research:

👥 Communication issues: Many issues are rooted in miscommunication between participants with different backgrounds. To facilitate interdisciplinary collaboration, education is key, including AI literacy for software engineers and managers (and even customers) but also training software engineers to understand data science concerns. The idea of T-shaped professionals [139] (deep expertise in one area, broad knowledge of others) can provide guidance for training and hiring.

👥 Communication and education is a key theme for our interventions in §4

📄 Lacking documentation: Clearly documenting expectations between teams is important. Traditional interface documentation familiar to software engineers may be a starting point, but practices for documenting *model requirements*, and assured *model qualities* are not well established. Recent suggestions like model cards [85] are a good starting point for encouraging better, more standardized documentation of ML components, but capture only select qualities. Given the interdisciplinary nature at these collaboration points, such documentation must be understood by all involved – theories of boundary objects [3] may help to develop better interface description mechanisms.

⚙️ Underinvesting in engineering: With attention focused on ML innovations, many organizations seem to underestimate the engineering effort required to turn a model into a product to be operated and maintained reliably in production. Arguably ML increases software complexity [59, 119, 96] and makes engineering-heavy practices such as data quality checks, deployment automation, and testing in production even more important. Project managers should ensure that both the ML and the non-ML parts of the project have sufficient engineering capabilities and foster product and operations thinking from the start.

📅 Lack of planning and process: Finally, ML with its more science-like process challenges traditional software process life cycles. It seems clear that *product requirements* cannot be established without involving data scientists in model prototyping, and often it may be advisable to adopt a model-first trajectory to reduce risk. But while a focus on the product and overall process may be delayed, neglecting it entirely invites the kind of problems reported by our participants. Whether it may look more like the spiral model

or agile [18], more research into integrated process life cycles for ML products (covering software engineering and data science steps) in all their different forms is needed.

3.2 Identification B: Meta-Summary of Challenges in Building ML Products

This section is a strongly shortened version of the CAIN'23 paper [90]: "*A Meta-Summary of Challenges in Building Products with ML Components – Collecting Experiences from 4758+ Practitioners*" [Nahar et al., 2023]

While there is no prior work on identifying *collaboration challenges* specifically, we find studies on identifying challenges faced by practitioners on different aspects of building ML products. Many researchers have *interviewed* or *surveyed* practitioners to identify what has really changed for them with the introduction of machine learning, often with the goal of identifying challenges, research opportunities, and best practices in a rapidly changing field. While some studies focus on specific aspects, such as challenges regarding requirements [143, 105], or fairness [47, 108], many others explore challenges more broadly. Many of these studies have identified similar challenges. We believe that we have reached a point where practices have settled and research on challenges approaches saturation – we think that now is a good time to step back and survey the collective findings of the research community. This collected challenge catalog will also enable us to triangulate the collaboration challenges identified in section 3.1 and discern the implicit alignment between them.

Thus, in this study, we aim to consolidate knowledge about challenges in the practice of building ML products, with a systematic literature survey of existing studies that *interviewed* or *surveyed* industry practitioners across multiple projects. We identified 50 studies of which, 30 conducted interviews, 11 conducted surveys, and nine did both, with a total of over 4758 identified participants (seven studies did not report the number of participants; some participants may have participated in multiple studies). Using the *meta-summary research method* [118, 110, 36], we analyze, organize, and synthesize findings across all these studies (as shown in Figure 3.3), answering the overall research question: **What are the challenges experienced by industry practitioners in building ML products?**

We group the challenges found in the meta-summary into categories. In a nutshell, we find practitioners struggle in different product development stages: (1) requirements engineering, (2) architecture, design, and implementation, and (3) quality assurance. We also find several engineering challenges in ML-specific stages, in particular (4) model development, and (5) data engineering. Other issues relate to cross-cutting concerns related to (6) process, and (7)

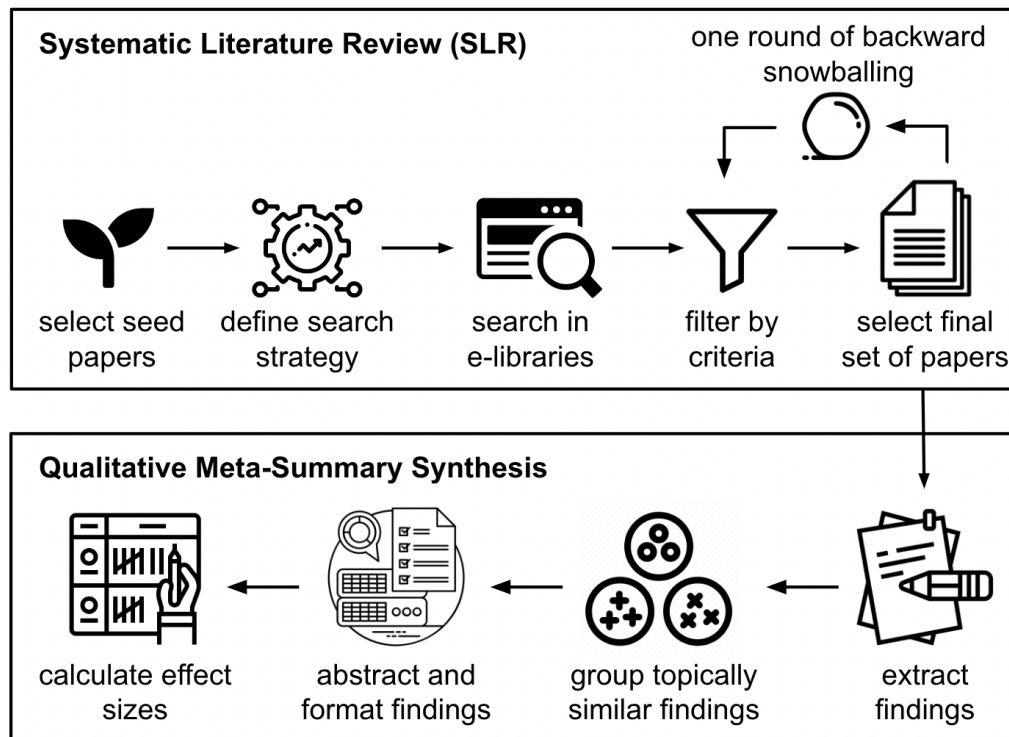


Fig. 3.3 Identification B: Research Design

organization and teams. Table 3.1 contains a summary of the findings. In this proposal, we only elaborate on the challenges from category (1) and (2); readers interested in the other challenges can refer to the full paper.

3.2.1 Completed Work: Research Design

The goal of this study is to summarize challenges in building ML products, accumulated from industry practitioners in prior research. To achieve this goal and answer our research question, we first define the appropriate search strategy and study selection criteria to find the relevant literature that identifies challenges through communicating with industry practitioners. We follow the guidelines for systematic literature review (SLR) for this paper collection step [61]. Then, we extract the data from the selected papers and analyze the data to complete the synthesis process. Several approaches have been explored for synthesizing qualitative research in software engineering, such as thematic synthesis, meta-ethnography, and meta-summary [50]. As we aim to discover patterns or themes of challenges in building ML products, as well as get a sense of the priority of the challenges based on the frequency of reports by industry practitioners, the meta-summary method is best suited for this research problem [50, 110, 118]. The meta-summary method provides a well-balanced

Requirements Engineering: Lack of AI literacy causes unrealistic expectations from customers, managers, and even other team members • Vagueness in ML problem specifications makes it difficult to map business goals to performance metrics • Regulatory constraints specific to data and ML introduce additional requirements that restrict development

Architecture, Design, and Implementation: Transitioning from a model-centric to a pipeline-driven or system-wide view is considered important for moving into production, but a difficult paradigm shift for many teams • ML adds substantial design complexity with many, often implicit, data and tooling dependencies, and entanglements due to a lack of modularity • Difficulty in scaling model training and deployment on diverse hardware • While monitorability and planning for change are often considered important, they are mostly considered only late after launching

Model Development: Model development benefits from engineering infrastructure and tooling but provided infrastructure and technical support are limited in many teams • Code quality is not standardized in model development tools, leading to conflicts about code quality

Data Engineering: Data quality is considered important, but difficult for practitioners and not well supported by tools • Internal data security and privacy policies restrict data access and use • Although training-serving skew is common, many teams lack support for its required detection and monitoring • Data versioning and provenance tracking are often seen as elusive, with not enough tool support

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

Process: Development of products with ML component(s) is often ad-hoc, lacking well-defined processes • The uncertainty in ML development makes it hard to plan and estimate effort and time

Organization and Teams: Building products with ML components requires diverse skill sets, which is often missing in development teams • Many teams are not well prepared for the extensive interdisciplinary collaboration and communication needed in ML products • ML development can be costly and resource limits can substantially curb/limit efforts • Lack of organizational incentives, resources, and education hampers achieving all system-level qualities

Table 3.1 Overview of Identified Challenges

synthesis mechanism, which is deeper than mapping studies, and not as exhaustive as meta-ethnography, which requires significant expertise and experience with the methodology and its philosophical stance [110]. Thus, we apply the meta-summary method [118, 36, 110] to perform the quantitative aggregation of the qualitative evidence that we present as findings. Figure 3.3 shows the overview of the research process followed in this study.

Paper Selection

To increase the reliability, reproducibility, and objectivity of the process for paper selection, we follow the established procedure of conducting systematic literature reviews [61]. Defining the right scope and corresponding search query required some iteration. We started by

assembling an initial set of 21 papers as a seed set (a common practice [38, 76]). The seed set was composed of papers that we knew well from our past work in this field. We then analyzed the seed set to define the keywords needed to retrieve those and similar papers. Based on them, we developed a search query, which we use to search papers in digital libraries and databases commonly used by software engineering review papers, e.g., [57, 25, 72]. This provided us with a total of 341 papers. Next, we selected 86 relevant papers by reading the title and abstract, and evaluating them against our pre-defined inclusion and exclusion criteria. To capture relevant papers that did not match our keywords in their abstract, we performed one iteration of backward snowballing [158].

Overall, our process resulted in a final set of 50 papers. Most of the papers were published after 2019. This sudden explosion of interview and survey studies with practitioners in recent years justifies our motivation for this study to aggregate all the findings of these papers. Most of the papers, 30 out of 50, were published in software engineering venues (including five at WAIN/CAIN), 11 papers in HCI venues, two papers in AI Ethics venues, and the seven remaining ones are scattered over other communities. A total of 947 interviews and 3811 survey responses were reported in 43 papers, and the seven remaining papers did not report specific counts of the interviewed or surveyed practitioners. Of the 50 papers, 31 papers explicitly list research questions or the aim of their research as the identification of challenges (or issues, problems, difficulties) in different aspects of building ML products. The other papers do not explicitly set a goal of identifying challenges but more broadly study the process of building ML products, yet they also report practitioner challenges in their findings.

Qualitative Meta-Summary Process

As stated earlier, we used the meta-summary research method [118, 110, 36] to synthesize the findings from the collected papers. This method is used to perform quantitative aggregation of qualitative findings, which are necessarily the thematic summaries of the underlying data from different studies. We conduct the following steps to perform the synthesis, as per the guidelines.

Extracting findings. Along with the standard metadata (title, source, venue, year, etc.), we extracted study-specific data regarding research questions, study method, interview and survey participant counts, and, most importantly, the challenges reported within the papers. We extracted challenges related to building software systems with ML components, but excluded those that relate exclusively to the data- and model-related work performed by a data scientist, such as algorithmic problems, notebook coding, and hyper-parameter tuning.

We extracted a total of 520 excerpts relating to challenges from the 50 papers. We stored all extracted information from each paper in a spreadsheet for further analysis.

Grouping topically similar findings. We organize the findings at the level of *reported challenges* that we extracted from the papers. We use *card sorting* to group similar findings [130, 51]. There were many rounds of card sorting including moving the cards back and forth between different clusters, splitting the cards to handle different dimensions, merging similar clusters, and splitting clusters. We developed three layers of clustering – the reported challenges extracted from the papers as the smallest unit, groups of common *themes* or patterns in the challenges as the second layer, and finally a third (or top) layer grouping the second layer clusters by development stages or cross-cutting concerns for the ease of reporting results.

Abstracting and formatting findings. For each of the second layer clusters, we abstracted out the concrete details of the reported challenges and summarized the clusters based on the identified themes of the groups. For this, we once again looked into the cards of each of the clusters individually and attempted to develop broad statements that capture the content of the cards in that cluster.

Calculating effect sizes. Methods for meta-summaries recommend reporting the frequency of findings in the original sources [118]. Since many of our analyzed papers ask similar broad research questions, we can carefully interpret findings mentioned more frequently as more common, though some papers clearly specialize in specific sub-areas such as fairness or software architecture [47, 68]. We do not attempt to count frequencies of mentions within the papers (“intensity effect size”) because they are not consistently reported, but just report the percentage of papers reporting on a challenge theme (“frequency effect size”).

Limitations and Threats to Validity

All research designs come with limitations that threaten the validity and credibility of results. As usual, readers should be careful when generalizing findings beyond what is allowed by the methods. Despite best efforts in our selection methods (SLR process, snowballing) we may have missed some relevant papers. In setting clear rules for scope, we had to make some judgment calls by consensus of all researchers for a number of papers, for example, whether to include [45, 6, 15, 11, 124].

As discussed earlier, the meta-summary synthesis method was chosen deliberately for its fit, but comes with its own limitations: it does not analyze original raw data, but only what is

reported by other papers. Organizing and categorizing the data required some interpretation of the papers and some judgment calls. The method encourages quantification of effect sizes, but those may not be entirely reliable as the analyzed papers use different methods and sometimes focus on specific subquestions.

It would have been interesting to analyze findings in additional dimensions, for example, whether team members in different roles or projects, or in different application domains, experience different challenges, or whether different challenges surface depending on the research method in the original study (e.g., survey vs. interview, open question vs. closed question). Unfortunately, data in the original studies is frequently not reported consistently and with enough granularity to enable such analyses.

While the meta-summary method can in principle also identify conflicts within the literature, this was not feasible in our study. The analyzed papers typically reported challenges, not the absence or relative importance of certain challenges. Given that different papers often had a different focus, rather than being replications of each other, we cannot conclude that not mentioning a challenge implies that there was no such challenge. Hence, we limited our analysis to aggregating and grouping reported challenges.

3.2.2 Findings

We report our findings of the meta-summary using the layers derived from the card sorting. The top layer includes development stages (1) *Requirements Engineering*, (2) *Architecture, Design, and Implementation* (with a special focus on (2a) *Model Development* and (2b) *Data Engineering*), and (3) *Quality Assurance*, plus (4) *Process* challenges and (5) *Team* challenges as crosscutting concerns. Within these top layer headings, we have our second layer clusters which are the abstracted challenges based on our identified themes, reported as the sub-headings in the following sections.

Note: In this proposal, we only include *Requirements Engineering* and *Quality Assurance* as two example layers; readers interested in the other ones can refer to the full paper.

Requirements Engineering

Requirements engineering is known as an important and challenging stage of any software project, but as a consistent theme, we find that practitioners argue that the incorporation of ML further complicates requirements engineering.

Lack of AI literacy causes unrealistic expectations from customers, managers, and even other team members [11, 125, 64, 143, 62, 79, 140, 149, 56, 88, 161, 48, 71, 91, 112, 32, 97] (17/50). Across many studies, many practitioners report that customers frequently have unrealistic expectations of ML capabilities in a product, like demanding a complete lack of false positives or expecting very high accuracy that is infeasible with provided resources (e.g., data, funding). Commonly, practitioners similarly blame a lack of *AI literacy* on customers not wanting to pay for the continuous improvement of the model: they have a static view of model development [56, 88] and only consider paying for coding, as they do not understand the need for experimental analysis [71] and even difficulty convincing engineering teams to invest in collecting high-quality data [62]. The issue of unrealistic requirements does not only come from customers, but also from team members within the company itself: Data scientists find it hard to explain the capabilities of ML to managers, requirements engineers, and even designers [149, 88, 48, 91, 32, 97]. According to practitioners, a lack of AI literacy in team members manifests particularly in defining and scoping the project: Stakeholders find it hard to understand the suitability of applying ML itself [64, 161], scoping and deciding the functional and non-functional requirements [143, 71], interpreting the model outcomes [125, 91, 112], and the infrastructure needs (e.g., appropriate data, monitoring infrastructure, retraining requirements) when building products [143, 161, 91]. Many practitioners also report that ML-specific system-level qualities like fairness and explainability are frequently ignored during requirements elicitation, as the stakeholders are not aware of them [143, 88, 108, 14].

This lack of AI literacy also came up in §3.1 (18), and supported in §4.1 (i-i)

Vagueness in ML problem specifications makes it difficult to map business goals to performance metrics [129, 70, 67, 125, 64, 143, 140, 149, 88, 108, 47, 71, 42, 91, 112, 75, 97] (17/50). Practitioners across many studies mention the challenge of formulating the specific software and ML problem in a way that satisfies business goals and objectives. ML practitioners find it difficult to map the high-level business goals to the low-level requirements for a model. While customers are broadly interested in improving the business, practitioners often find it difficult to quantify the contribution of the ML model and its return on investment. Also, *Responsible AI* initiatives find it difficult to quantify their contributions to the business, for example, measuring the value added by improving fairness and explainability, or to deliberate about tradeoffs between conflicting fairness and business objectives [14, 108, 47, 97]. Even with some notion of the responsible AI requirements in hand, practitioners find the requirements vague and not concrete enough to actually implement (e.g., unclear subpopulations and protected characteristics to balance discrimination) [143, 108]. On the other hand, practitioners also frequently report that many projects are exploratory without

clear upfront business goals, thus, starting off the project without clear requirements is pretty common, albeit often problematic [125, 149, 71, 42].

Regulatory constraints specific to data and ML introduce additional requirements that restrict development [42, 124, 143, 14, 48, 93, 125] (7/50). Practitioners in multiple studies expressed how regulatory restrictions constrain ML development and require audits and involvement from legal teams. Privacy laws such as GDPR impose additional requirements on ML practitioners such as ensuring the collection of individual consent [143, 48] and providing the nontrivial ability to remove individuals from training data after they revoke consent. Similarly, practitioners in regulated domains report a need for explainability and transparency that prevents them from using deep learning and post-hoc explainability techniques [124, 42, 14].

Quality Assurance

One of the biggest changes that the incorporation of ML models has brought into traditional software development is challenging the traditional notion of correctness, where models are evaluated for accuracy or fit rather than whether they fully meet a specification. Understandably this impacts the conventional processes and practices associated with testing and quality assurance.

Testing and debugging ML models is difficult due to lack of specifications [125, 140, 56, 107, 163, 87, 40, 70, 67, 149, 88, 71, 123, 42, 112, 126, 75, 4, 9] (19/50). Practitioners find testing and debugging ML models challenging. In particular, they ubiquitously report difficulty establishing quality assurance criteria and metrics, given that no model is expected to be always correct, but it is difficult to define what amount and what kind of mistakes are acceptable for a model [71, 42, 75, 70, 149, 88, 9, 56, 163, 40]. In particular, practitioners find it difficult to define accuracy thresholds for evaluations. Furthermore, practitioners report finding it difficult to select adequate test data, specifically curating test data of sufficient quality and quantity that is representative of the production environment [67, 149, 140, 107, 87]. Curating test data for ML testing is also considered costly and labor-intensive, and practitioners desire methods and tools from the research community for automated test input generation to reduce this cost [70, 149, 56]. Practitioners consider it a challenge to get labels for test data and evaluate test quality (e.g., in terms of coverage) due to the difficulty of defining the valid input space and the test oracle problem [70, 149, 40]. Practitioners also mention the silent failing of models (i.e., models give wrong answers rather than crashing), the long tail of corner cases, and the “invisible errors”, that are handled on an ad-hoc

basis without a systematic framework or a standard approach [126, 149, 40]. Additionally, practitioners raise challenges regarding evaluating model robustness, on one hand, suffering from the lack of a concrete methodology [107, 40], and on the other hand, having various metrics but no consensus on which metric to use [70].

Testing of model interactions, pipelines, and the entire product is considered challenging and often neglected [75, 70, 64, 88, 107, 163, 87, 40] (8/50). Testing literature often focuses on ML models and data quality, but less on how models are integrated into the product, and even less on the infrastructure to produce the models. Practitioners find sole unit testing of individual models insufficient and ineffective, due to the entanglement of models and different ML components, as well as the difficulty of explaining why an error occurred due to the low interpretability of individual models [70, 149, 163]. The lack of pipeline and system testing beyond the model is also considered a problematic area [64, 88, 107, 163, 87, 40]: While practitioners tend to focus more on the data- and model-related issues, the error handling around the model is found to be insufficient in previous studies [163, 40], leading to system failures even where the model gives the correct results [87]. Practitioners also report having no systematic evaluation strategy or automated tools and techniques for pipeline and system-level testing [70, 88].

Testing and monitoring models in production are considered important but difficult, and often not done [125, 126, 70, 88, 107] (5/50). Many practitioners recognize the need to test in production (online testing), since offline test data for models may not be representative, especially as data distributions drift. However, practitioners consider online testing complex as it is not trivial for them to design online metrics that depend not only on the model but also on the external environment, user interactions after deployment, and the context of the product overall [126, 70]. Practitioners also find online testing very time-consuming, as it requires longer observation periods to determine meaningful results [125, 126]. Practitioners also pointed out that there is no surefire strategy to precisely detect when the model is underperforming in online testing [126].

There are no standard processes or guidelines on how to assess product qualities such as fairness, security, and safety in practice [54, 63, 47, 140, 48, 111, 124, 14, 15] (9/50). Research often discusses how machine learning influences fairness, robustness, security, safety, and other qualities, but practitioners report that they find evaluating these challenging. While practitioners consider these qualities important [54, 63], they often report having no effective methodology or concrete guidelines for evaluating them [63, 47, 140, 48, 111,

This lack of guidelines and processes also came up in §3.1 (❌), and supported in §4.3 (i-i)


124, 15]. Even regarding fairness, which has received a lot of research attention lately, practitioners report finding it hard to apply auditing and de-biasing methods due to not having a proper process in place [47, 48]. Some practitioners report waiting for complaints from customers rather than being proactive when it comes to fairness [47] or even blindly expecting the algorithms to inherently provide qualities like security against attacks [63].

3.2.3 Discussion

With this meta-summary study, we find that practitioners report challenges in all stages of the development process, from the initial requirements specification stage to quality assurance of the deployed product. They report a broad range of issues from lacking process, organizational structure, and team collaboration strategies.

Arguably, many reported challenges are not new to software engineers, and likely many software engineers may have reported similar challenges in non-ML projects. It seems though that the introduction of machine learning exacerbates some universal challenges and introduces new ones. For example, software engineering literature is well aware that **requirements engineering** is challenging, with customers having *unrealistic expectations* and developers directly jumping into coding *without understanding requirements* first. While our study does not support direct comparisons, it seems that these problems haunt ML practitioners more, given how ML inspires hopes for amazing capabilities, but in a way that may be difficult to understand and specify without substantial ML expertise. Similarly, **team collaboration** and **organizational** challenges are well-known in traditional software engineering, but those seem to become even more central with the additional complexity and inclusion of more people with different backgrounds, cultures, and priorities. Other challenges seem new, such as the data- and model-related challenges associated with ML components, and several of the reported challenges regarding architecture and quality assurance stemming from the different nature of reasoning in machine learning.

A finding from our study is that there is much more consensus on what the challenges are, than how to overcome them. Some challenges could be addressed with new tooling or new practices; for others, it may be possible to simply adopt existing good engineering practices; and yet others may just be intrinsically hard problems. For example, for the challenges of unrealistic requirements, many practitioners mention suffering from unclear model requirements, but we still do not seem to have a good solution to that, and additional research on *how to elicit and describe requirements for models may be needed*. Another area for future research would be to *better understand and prepare for regulatory constraints and provide evidence of compliance*.

 These challenges of unclear model requirements and regulatory constraints are supported in our interventions (§4)

Overall we believe that a lot of progress can be made with better education and better adoption of good software engineering practices. There are plenty of research opportunities to create new interventions altogether. We hope that the collection of challenges, which can be traced to the original studies where they were raised by practitioners, will help select and prioritize research and education in our community.

3.3 Summary

In this chapter, we explored the challenges ((~~NO~~)) encountered in building ML products. The contributions of this challenge-identification thrust include:

- A list of key challenges identified through an interview study involving 45 practitioners across 28 organizations.
- A broad catalog of challenges in ML product development, derived from an extensive meta-summary of academic literature. This study complements our interview findings and again highlights underlying implicit issues in collaboration.

Furthermore, the chapter demonstrates the application of carefully chosen research methods aptly suited to our studies, such as grounded theory, systematic literature review, and qualitative meta-summary, as well as a rich selection of qualitative data analysis techniques.

While many of the identified challenges require research attention, we start by designing three interventions ((~~i-i~~)) to address three specific problems. A recurring issue identified in both our challenge identification studies is the difficulty of eliciting product and model requirements. We have found that a significant *semantic boundary* exists among practitioners at this collaboration point. To address this, an intervention to facilitate *knowledge translation* is necessary, which would foster collaboration, support negotiation, and ensure agreement during the requirements collaboration point.

A second point that needs to be addressed highlights a conflict of interest among practitioners, leading to the emergence of a *pragmatic boundary*—arguably the most complex type of *knowledge boundary*—centered around responsible AI (RAI). We find that despite prior efforts to create shared boundary objects such as guidelines for RAI assessment and documentation [109, 84], collaboration remains challenging at this point due to conflict of interest among the practitioners. Thus, this situation necessitates a more *transformative* or *political approach* aimed at creating shared interests and agreements.

Lastly, our focus extends to a specific aspect of responsible AI—Explainable AI (XAI), which due to its novelty, encounters all types of *knowledge boundaries*; effective development in XAI necessitates the establishment of a *common lexicon*, *shared interpretations*, and strategic *negotiation points to resolve conflicts*. While our meta-summary study high-

lighted the importance of requirements related to explainability and transparency [121, 26], especially in regulated domains [21, 7], our interview study revealed a significant gap in practitioners overlooking the need for model explainability. This stark contrast underscores asymmetry among different practitioners and stakeholders, indicating a crucial need for an intervention to bring all parties to a common understanding, and agreement on the importance of explainability in AI models, and guidance to achieve explainability.

Chapter 4

Designing Interventions

In this chapter, to demonstrate how we can overcome the identified challenges (❌), we present three studies that design interventions (👤👤) aimed at enhancing collaboration within different spheres, as highlighted in my thesis statement: "*Then, (b) I propose interventions to promote effective collaboration among these development team members.*"

Through these interventions, we demonstrate principles or ideas of how those collaboration problems can be solved, instantiated in the three examples. We make use of the framework for managing *knowledge boundaries* by Carlile [23] (as discussed in Chapter 2), and demonstrate the creation of different *boundary objects* to bridge the diverse types of *knowledge boundaries*, namely, *syntactic*, *semantic*, and *pragmatic*, existing in the problem space. This is useful especially because not all collaboration issues encompass all three types of knowledge boundaries, and to devise an effective intervention for a specific collaboration problem, it is essential to first ascertain the specific type of knowledge boundary or gap present and then determine the most suitable type of boundary object that can effectively address and resolve this gap.

In designing our interventions, we primarily focus on bridging the knowledge gaps among collaborators and enhancing their understanding of each other's concerns. As previously discussed in Section 3.2, we emphasize the importance of education and the development of T-shaped professionals—who possess deep skills in one area and the ability to collaborate across disciplines—as pivotal in this domain. To assist this learning process, we employ several strategies, such as providing the professionals with concrete examples and context-specific information relevant to their product use cases. In this endeavor, we find large language models (LLMs) to be useful, which may help prepare practitioners, who often work in silos, to engage effectively with one another. Moreover, we, ourselves, actively engage in collaborative activities to create boundary objects, so that we have a deeper understanding of the different mindsets and concerns of various stakeholders while developing the intervention.

	Collaborators	Knowledge Boundaries		
		Syntactic Language mismatch	Semantic Different interpretation	Pragmatic Conflict of interest
Problem A: Challenges in eliciting clear, realistic, actionable model requirements	 product team model team	Different format and documentation standards in eliciting model and product requirements	Different assumptions and subjectively in interpreting requirements for model and product	
Problem B: Conflict in responsible AI (RAI) engagement	 RAI activists non-champions	New vocabulary and terminologies such as "fairness," "transparency," and "accountability."	These terminologies have different interpretations in technical, non-technical and regulatory language	Different priorities, and importance assigned to RAI concerns
Problem C: Lack of understanding and support for explainable AI (XAI)	 product team model team governance team	New terms, explainability techniques, and metrics	Different interpretations of explanations for traditional software and ML, and for different stakeholders	Different priorities on model explanations based on different purposes and use cases

Fig. 4.1 Targetted Problems Across Different Knowledge Boundaries

Broadly, we focus on the following three broad themes for our plans of action to mitigate the knowledge gaps in our solutions: (a) explicitly defining required terms and concepts, (b) using LLMs for contextual translation and illustration of concepts, and (c) co-designing guiding artifacts (depicted in Fig. 4.2). We describe our designed interventions in line with these themes as follows. For better readability, we have structured the following paragraphs to outline what the interventions aim to achieve, what challenge the collaboration points entail, and the broad solution themes or ideas.

Our first intervention aims to facilitate *eliciting actionable model requirements*, a critical issue highlighted in our studies under the identification thrust. In both of the challenge-identification studies, we identified the challenges of unrealistic requirements as a major problem, where practitioners mention suffering from unclear model requirements, and require help with eliciting and describing requirements for their intended model in the ML product. This collaboration point reveals a significant *semantic boundary* (with coexisting *syntactic boundary*, as summarized in Fig. 4.1) and underscores the necessity for a strategy to connect the model development team—primarily composed of data scientists—and the product team, which typically includes software engineers and project managers. Our proposed solution aims to enhance the negotiation process between these teams through effective *information transfer* and *translation* (as summarized in Fig. 4.2, and based on themes (a) and (b)), which would help each to identify feasible and actionable model requirements and avoid unrealistic and vague requirements.

Cf. §3.1.2 and §3.2.2 for unclear and unrealistic model requirements

Our second intervention aims to encourage data scientists to engage with the principles of *responsible AI (RAI)*. This collaboration point reveals an unresolved *pragmatic boundary* (the *syntactic* and *semantic boundaries* are mitigated by existing RAI assessment guides) that

In line with the rare consideration of RAI, §3.1.2

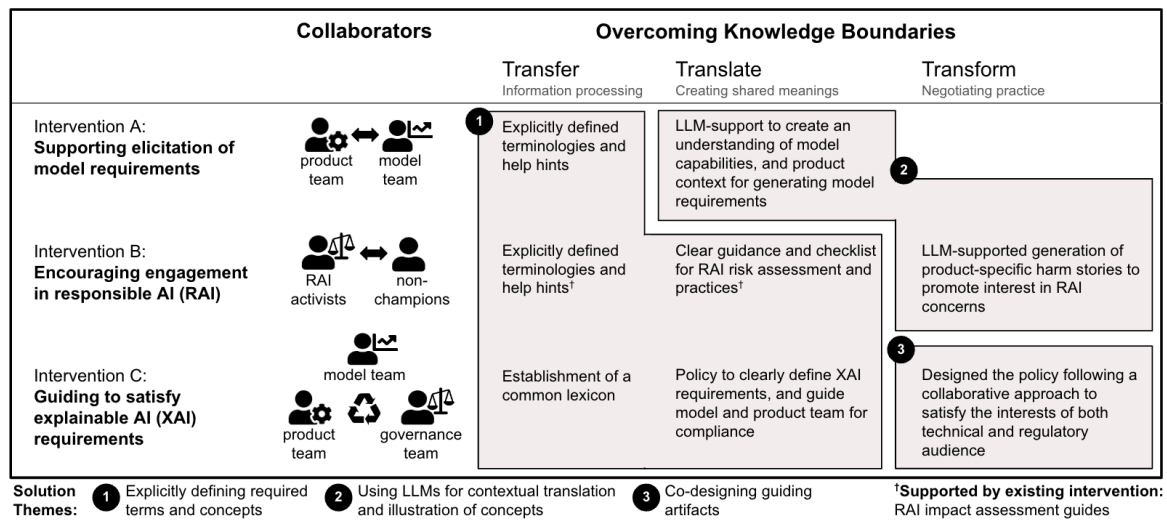


Fig. 4.2 Interventions Facilitating Collaboration to Overcome Knowledge Boundaries

results from a conflict of interest among the data scientists, software engineers, and the rest of the product development team with the project manager and the governance team. Despite well-recognized harms from ML products to the end-users, data scientists and software engineers often feel resistant or indifferent to the importance of ethical considerations in developing ML products [108, 88], who we refer to as *non-champions*. Our proposed solution aims to align the values and practices of non-champion data scientists with broader organizational goals through a *transformational* approach (primarily based on theme (b)) that promotes awareness and acceptance.

Finally, our third intervention aims to establish a common ground for explainable AI for the development team and the stakeholders who require explanations. Explainable AI increases the relative complexity and novelty in the *knowledge boundary*, as the concept of explainability differs significantly for machine learning from its traditional notions in software engineering. In software engineering, explainability is achieved through well-documented, deterministic algorithms and supported by debugging tools that allow direct tracing from inputs to outputs. In contrast, explainability in ML involves deciphering the "black box" of complex algorithms where decisions are data-derived and not directly written in code, shifting the focus of explainability to understanding the layers of abstraction and approximation that occur within the algorithm. This collaborative point reveals all *syntactic*, *semantic*, and *pragmatic boundaries* that are still unresolved. Our proposed solution aims to establish a common ground by creating a policy as a boundary object (inline with the theme (c)) that would guide the data scientists and software engineers on what explanations to generate, for whom, and for what purposes.


In line with the no standard process for XAI, §3.2.2

4.1 Intervention A: Supporting Elicitation of Model Requirements

As discussed in Section 3.1, our investigations revealed significant collaboration challenges during the requirements and planning stages of ML product development. We identified a constant tension between eliciting product requirements (i.e., functional and non-functional requirements of the overall product) and model requirements (i.e., goals and constraints of the model, such as accuracy, latency, memory, and data availability), which manifests two problematic scenarios: (a) often project managers and software engineers, with a lack of understanding of the model’s capabilities, define requirements without involving data scientists, which leads to unrealistic model requirements, (b) conversely, when data scientists are tasked with requirements elicitation, they tend to prioritize specifications pertinent to the model, often overlooking broader product requirements such as usability. This misalignment highlights the need for a more collaborative approach to requirements elicitation in ML products.

Furthermore, data scientists frequently receive vague model requirements. Even if they try to infer intentions behind them, it is hard for them as they are constrained by having a limited understanding of the product that the model will eventually support. For example, the data scientists may receive some data and a goal to predict something with high accuracy, but no further context, e.g., one of our interviewees shared “*there isn’t always an actual spec of exactly what data they have, what data they think they’re going to have and what they want the model to do.*” The difficulty of providing clear requirements for an ML model has also been raised in the literature [62, 105, 75, 160, 145, 127].

To address this challenge, we intend to develop an intervention that helps educate both data scientists and software engineers to bridge their knowledge gap so that they can elicit appropriate model requirements. Our proposed solution is to create an assistant leveraging large language models (LLMs). On one hand, this tool will support data scientists in identifying relevant model requirements tailored to the specific context of the product, as well as ask the right questions to software engineers and project managers (a *translation* process depicted in Fig. 4.2 – it also supports knowledge *transfer*, but we focus on the higher level here), which would help the data scientists negotiate and reach a consensus on model specifications. On the other hand, the tool will assist software engineers in formulating realistic model requirements that align with the capabilities of an expected model. In this way, by allowing both sides to refine their requirements independently before engaging in cross-team negotiation, the tool will foster a more grounded and effective collaboration process. Additionally, the intervention will help validate whether the overall model requirements are

 Find the details on vague model requirements in the second finding in §3.1.2 (p. 18)

consistent and satisfiable. Overall, we aim to answer the following research question: **How to design a generative AI prompt engineering pipeline to help siloed data scientists and software engineers elicit model requirements that are realistic, concrete, and complete?**

4.1.1 Problem Scoping

In the industry, we observed that data scientists often operate in isolation, often struggling with unclear model requirements and having limited access to team members who could provide necessary clarifications. Recognizing the value of efficient and productive meetings, our goal is to empower these data scientists to better prepare for such interactions. First, our tool would help them generate initial model requirements in isolation by suggesting clearer and actionable model requirements with the broad product context in mind. Second, it would suggest specific clarification questions that they can ask during discussions with team members. Ultimately, this preparation will enhance their ability to negotiate and fully comprehend the model requirements.

We also observed that project managers and software engineers often establish unrealistic model requirements due to not involving data scientists in the initial requirements elicitation process. To address this, our tool would assist such team members with limited AI knowledge to identify unrealistic assumptions about model capabilities and make the model requirements concrete and actionable, even when data scientists are not present in meetings. This approach will enhance collaboration by fostering a better understanding of each other's perspectives and ensuring more realistic and achievable project goals.

4.1.2 Proposed Work: Research Design

For better negotiation of model requirements, we leverage the concept of **negotiation cards**,¹ a boundary object to invoke discussion among different stakeholders and reach agreements on product and model requirements. Our approach involves the creation and refinement of negotiation cards through a dynamic and interactive process, with multiple interaction points from its initial generation to iterative refinement until a final agreement is reached. In this whole interaction, we want to (a) enable data scientists to ask the “right” questions to the software developers and project managers, (b) guide software engineers and project managers in defining realistic requirements for the model capabilities, and (c) help both parties to identify any overlooked requirements, and create more concrete and achievable requirements for the ML product.

¹A concept from MLTE [78], not published yet

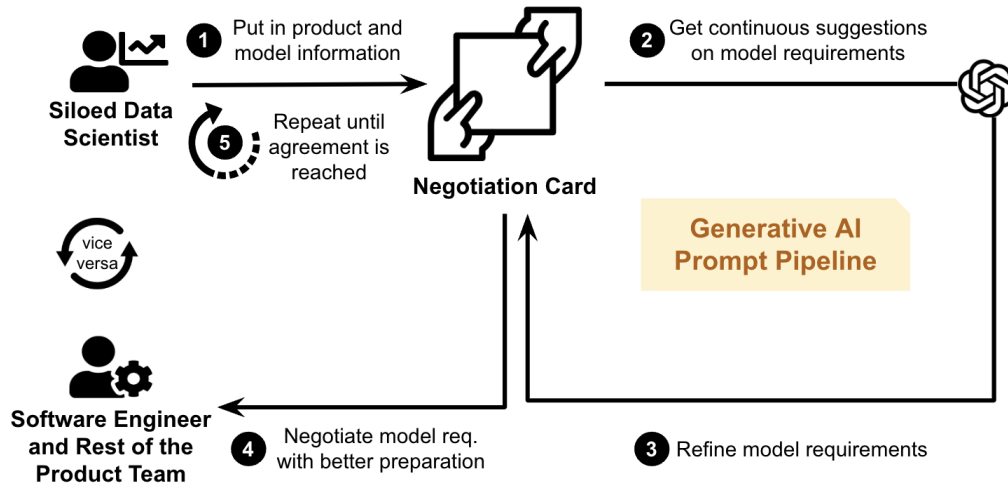


Fig. 4.3 Intervention A: Proposed Approach

Proposed Approach. We plan to develop two distinct prompt pipelines tailored to meet the specific needs of data scientists, and software engineers (and other members of the product team). For data scientists, the custom pipeline will assist them in incorporating the product context while generating the model requirements, including proposing potential requirements and providing clarification questions that can be posed to the product-side team to reach a mutual agreement on the requirements.

The workflow for data scientists (as depicted in Fig. 4.3 interacting with this process progresses as follows:

(1) **Initiate:** In this step, we will leverage on an existing framework called Machine Learning Test and Evaluation (MLTE) [78], designed to provide a structured process for evaluating ML products. This framework aims to ensure that models deployed in real-world settings are both effective and aligned with the broader goals of organizations and their stakeholders. MLTE supports negotiation on model quality requirements between developers and stakeholders through *negotiation cards*.

Our approach will incorporate MLTE along with negotiation cards, which will allow the data scientists to use the framework’s user interface (UI) to put in the initial model and product information at this stage.

(2) **Generate:** In this step, we will leverage on the generative AI capabilities through a customized prompt engineering pipeline. Prompt engineering [104] is the practice of crafting effective input prompts to guide the behavior of generative AI models, a technique increasingly prevalent across various applications [147, 66, 152].

We will develop a prompt engineering pipeline specifically designed to assist data scientists in eliciting model requirements. This specialized pipeline will generate model

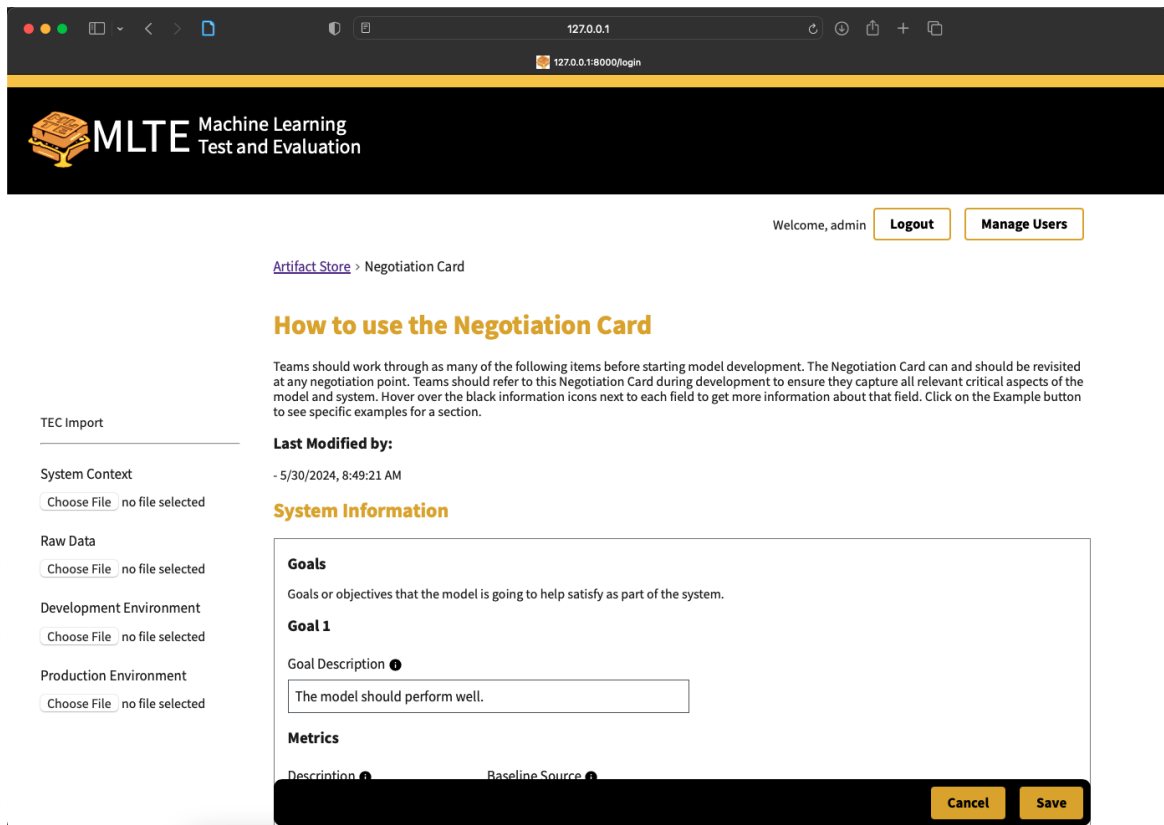


Fig. 4.4 MLTE User Interface

requirements considering the product context, so that the requirements align with the ML product's needs and objectives.

(3) **Refine**: At this phase, the data scientists would enter model requirements into the negotiation cards and iteratively refine them using the continuous recommendations from the generative AI model, again through a customized prompt engineering pipeline. At this point, the pipeline would also generate targeted clarification questions for the product team. These questions would be designed to facilitate effective dialogue and negotiation between data scientists and product team members to align the model requirements with the product's objectives as well as the model's capabilities.

(4) **Negotiate**: Equipped with a well-prepared set of model requirements, at this stage, the data scientist would present and negotiate the model requirements with the product team.

(5) **Iterate**: The workflow is repeated until an agreement is reached amongst all stakeholders.

For software engineers and other product-side stakeholders, the workflow is essentially reversed, with tailored modifications in the suggestions from the generative AI pipeline

in step (2). Specifically, here, the pipeline would focus more on the model’s context and establish guardrails to prevent unrealistic expectations regarding the model’s capabilities.

Plan for Evaluation. To assess the efficacy of the proposed novel approach for eliciting model requirement, we will implement it as a prototype tool and conduct a controlled experiment under laboratory conditions. The primary objective of this evaluation is to determine whether this approach aids novice data scientists or software engineers in negotiating and formulating realistic, clear, and actionable model requirements.

For the experiment, participants will be recruited from a pool of students and practitioners who possess experience in either data science or software engineering, but lack experience in the complementary field. This selection criterion ensures that participants are representative of the siloed practitioners who would benefit from the proposed tool in a real-world context. The experiment will employ a *between-subjects design*, randomly assigning participants to either an experimental group, who will use the proposed tool, or a control group, who will not have access to the tool. In order to contextualize the tasks, we will use one of the ML product use-cases derived from our catalog of open-source ML products, as discussed in Chapter 5 (⚙️). Participants will engage in a *role-playing* scenario designed to simulate real-world interactions between data scientists and product development teams. For participants with a data science background, the evaluator will assume the role of a product team member, whereas for participants from a software engineering background, the evaluator will assume the role of a data scientist. This role-playing activity is intended to mimic authentic cross-disciplinary communication and requirement gathering.

To begin with the study, the participants will receive an overview of the project and their assigned role within the scenario. The overview will include a brief description of the product and model requirements, replicating initial project briefs typically encountered in professional settings. Participants will then engage in a controlled interaction with the evaluator: *the evaluator will respond to participant queries but will not volunteer information unprompted*. This will allow us to observe whether the tool facilitates the generation of clarification questions and assists participants in eliciting comprehensive model requirements. We would also observe how participants navigate and overcome ambiguities in the project description during generation of the model requirements. Throughout the interaction, data will be collected on several metrics: *quality of clarification questions*: the number and relevance of the questions asked by the participants aimed at clarifying ambiguities, *quality of requirements*: comprehensiveness and clarity of the model requirements generated by the participants, *participant confidence*: self-reported levels of confidence in the requirement


elicitation task, measured through post-interaction surveys, and *efficiency*: time taken for participants to reach their satisfactory level in the generated model requirements.

We will conduct both qualitative and quantitative analysis on the study data. We will carry out content analysis on the interaction transcripts to identify common themes and patterns. We will also conduct statistical analysis to compare the performance of the experimental group and the control group across the aforementioned metrics. To ensure the validity and reliability of the experiment, we will conduct several pilot studies at the beginning to refine the interaction protocol. We will also have multiple evaluators to independently code and assess the quality of clarification questions and model requirements to enhance the reliability of qualitative data.

4.2 Intervention B: Encouraging Engagement in Responsible AI (RAI)

The concept of Responsible AI (RAI) has garnered significant attention in recent years, within the context of ML product development. Numerous scholarly publications have explored various RAI techniques and guidelines. However, in both our interview study and meta-summary study involving academic literature with industry participants, we found a substantial gap between theoretical research and practical implementation in the industry [108, 48]. While existing guidelines for RAI assessment, documentation, and tooling [109, 84, 20] help address both *syntactic* and *semantic* boundaries among stakeholders by defining terminologies, translating their interpretations, and providing guidance and checklists, we find that data scientists and software engineers often exhibit resistance or indifference toward adopting these solutions for their ML product development, through conducting informal interviews and shadowing meetings, in close collaboration with an industry partner. We observed that while the governance team implemented various processes and documentation templates in the organization to urge data scientists to consider RAI aspects, the data scientists themselves were reluctant to adopt these measures and even showed annoyance in the absence of the governance team.

This scenario highlights a severe and unresolved *pragmatic knowledge boundary*—a complex type of knowledge boundary characterized by misaligned practical interests and organizational priorities—among data scientists, software engineers, project managers, and governance teams. Addressing such a boundary requires more than just a technical solution—it requires a more transformative and politically nuanced approach to align interests and foster agreements. Therefore, in response to these challenges, we propose a multi-faceted

 Cf. §3.1 (p.21) and §3.2 (p.29, p.31) for RAI discussions

intervention designed to encourage data scientists and software engineers to engage with the principles of RAI. Our intervention specifically targets this *pragmatic boundary* (and the co-existed *syntactic boundary*) and aims to increase awareness and acceptance in data scientists and software engineers to align their values and practices with the broader organizational goals and ethical standards.

Central to our intervention is a novel approach that leverages large language models (LLMs) to generate compelling stories of harm that ML products can inflict on end-users. This approach employs a comprehensive prompt engineering pipeline to ensure that the stories produced are concrete, severe, surprising, and diverse. The intent is to make the potential harms tangible and relatable, thereby fostering a deeper understanding and commitment to responsible AI practices among practitioners. Overall, we address the following broad research question: **“How to design a generative AI prompt engineering pipeline to encourage data scientists and software engineers to engage with responsible AI in ML product development?”**

4.2.1 Problem Scoping and Related Work

As discussed earlier, despite the well-recognized potential harms that ML products can cause on end-users, data scientists and software engineers frequently display resistance or indifference toward integrating RAI considerations into their development processes. Many companies and governance teams attempt to address this issue by implementing extensive RAI assessment templates, requiring practitioners to fill them out in order to evaluate the risks associated with their ML products. However, this approach often exacerbates the situation, as it is perceived by the resisting stakeholders as an additional burden or "more work." This perception triggers annoyance and reluctance, with many practitioners believing that their ML products—especially simpler ones—pose no real harm, thereby overlooking potential risks due to this biased viewpoint. Our goal is to engage these stakeholders, whom we refer to as "non-champions" of RAI, and assess whether our proposed intervention can influence their perspectives and attitudes towards responsible AI.

We find many previous solutions to help stakeholders identify potential harms that their ML products may cause to end-users, leveraging generative AI models. For instance, AHA! (Anticipating Harms of AI) is a generative framework designed to assist ML practitioners and decision-makers in predicting potential harms and unintended consequences of ML products prior to their development or deployment. This tool uses LLM to generate fictional scenarios, or *vignettes*, illustrating how various users might experience problematic ML behaviors. Another similar tool, Farsight, helps users identify potential harms of their ML prototypes by highlighting news articles about relevant AI incidents and providing

LLM-generated potential use cases and harms. Blip is another similar system that retrieves real-world undesirable consequences from online articles, summarizing and categorizing them for researchers and practitioners using LLMs. While these tools are valuable in solving *syntactic* and *semantic* boundaries and cater to individuals who are already motivated to identify and address potential harms, they lack provisions for guiding or encouraging those who might not intrinsically consider RAI principles—often due to conflicts of interest or lack of awareness. Our intervention aims to bridge this gap by specifically targeting individuals who may not be inherently motivated to engage with RAI practices. We focus on designing mechanisms that not only educate but also incentivize and motivate this “non-champion” audience, thereby overcoming the *pragmatic boundary*.

4.2.2 Proposed Work: Research Design

As our goal is to incentivize “non-champions” to care about RAI, we will design an approach that will highlight unexpected and severe issues that could arise, impacting end-users in ways that non-champions may not have envisioned before. By showcasing surprising instances, our aim is to break through any apathy or resistance, urging these stakeholders to reconsider their stance on RAI. Highlighting severe examples will underscore the critical nature of RAI, helping non-champions recognize the gravity of overlooking ethical considerations. To maintain credibility and engagement, we will ensure that these stories are both relevant and concrete, steering clear of unrealistic situations that may turn them away. Therefore, we design a sophisticated pipeline capable of producing harm stories that are severe, surprising, concrete, relevant, and diverse, which we define as follows:

- **Severity/Extremeness:** This quality corresponds to the extent to which the story involves potential harm, considering both the magnitude of the impact (how intense or profound the harm is) and the scope (how widespread the harm is, including the number of individuals or communities affected). High-severity stories involve significant and extensive harm, affecting numerous individuals or communities profoundly. Conversely, low-severity stories might describe limited-impact scenarios.
- **Surprisingness:** This quality measures how unexpected the story is, based on common examples and media representations. Low surprisingness stories are those that people can easily think of given the context, often seen in public discourse. High surprisingness stories are those that are novel and not commonly represented, thus more likely to catch attention and provoke thought.
- **Concreteness:** This quality pertains to the specificity and tangibility of the story. A story with low concreteness offers generic descriptions and lacks specific details. A

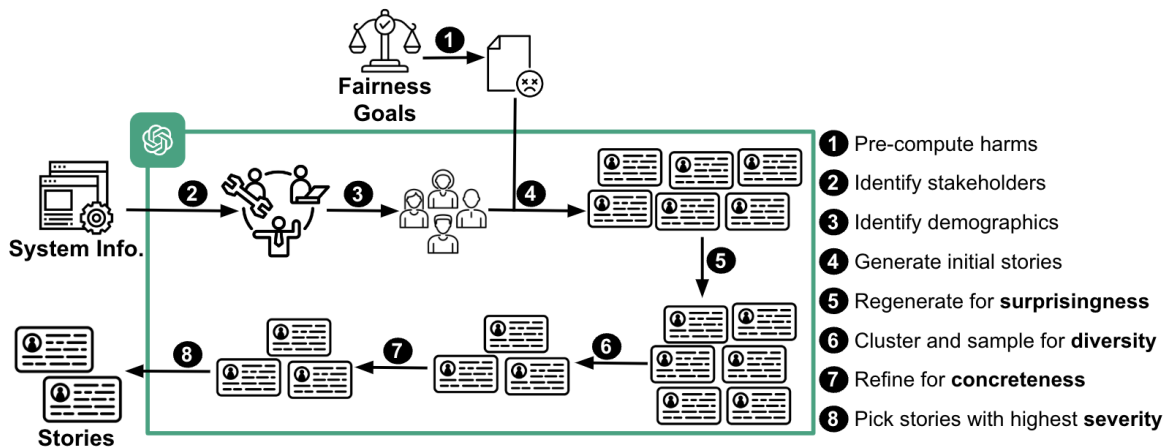


Fig. 4.5 Prompt Engineering Pipeline for Story Generation

story with high concreteness provides a detailed account involving specific named individuals or entities, making the narrative more relatable and impactful.

- **Relevance:** This quality assesses how pertinent the story is to the scenario it aims to illustrate and its alignment with fairness goals. High relevance means the story is closely related to the description of the scenario and addresses fairness goals such as quality of service, allocation of resources and opportunities, and minimization of stereotyping, demeaning, and erasing outputs.
- **Diversity:** As multiple stories are presented, this quality considers the variety within the generated stories. High diversity means that the stories differ from each other in various ways, encompassing different types of stakeholders, demographic groups, harms, consequences, and model behaviors. This ensures a wide-ranging representation of scenarios and potential impacts.

Approach. To identify the potential harms prior to generating our stories, we base our approach on Microsoft’s Responsible AI (RAI) assessment guide [83]—a well-established framework widely adopted by many organizations for evaluating Responsible AI. This guide offers a systematic structure for identifying potential harms and assessing their impacts, and serves as a conversation starter within the development team. While we do not strictly adhere to the guide, we draw inspiration from various elements, such as identifying stakeholders prior to brainstorming potential harms, and connecting to different system goals (e.g., quality of service) for fairness considerations.

Building on this guide and aligning with our desired qualities in the stories, we have designed a comprehensive prompt engineering pipeline (illustrated in Fig. 4.5). The process begins by feeding the pipeline with a product overview, which includes a brief description of

the product, its purpose, and intended uses. For each intended use, the pipeline generates an output of two harm stories. Additionally, the system allows for the generation of up to three more harm stories upon user request. It also features capabilities for obtaining feedback on the generated stories and regenerating them if necessary. It is important to note again that the objective of this pipeline is not to exhaustively enumerate all potential harms but to influence and persuade “non-champions”—those not yet fully on board with Responsible AI principles—by vividly illustrating the real-world implications of ML products on end-users.

This pipeline encompasses eight stages, outlined as follows:

(1) Pre-compute harms: For generating stories for different types of harms, we start by pre-computing the harms and feeding it to our generative AI model. These pre-computed harms are based on the three fairness goals outlined by Microsoft’s RAI assessment guide [83]: quality of service, allocation of resources and opportunities, and stereotyping, demeaning, and erasing outputs. The generated harms include cultural misrepresentation, reinforcement of biases, access to opportunities, and erasure of minorities.

(2) Identify stakeholders: At this stage, we use the generative AI model to identify a comprehensive range of stakeholders. Using two prompts, we generate both direct and indirect stakeholders, and crucially, put forth less obvious stakeholders alongside the regular ones, according to design justice [27]. This generation of non-obvious stakeholders is to broaden the non-champions’ understanding of their ML products’ influence on a varied group of users. These generated stakeholders are presented as suggestions, allowing the tool users to select relevant ones as per them.

(3) Identify demographics: For each selected stakeholder, the model generates possible demographics based on the pre-defined categories, such as age, gender, and ethnicity. This step is to facilitate the subsequent pipeline steps on creating relevant stories for marginalized stakeholders.

(4) Generate initial stories: At this stage, the model drafts initial harm stories for each harm category in relation to stakeholder and demographic group pairs². This process can be visualized as a matrix with columns for pre-computed harms and rows for stakeholders, where each cell contains a generated story illustrating the impact on a specific stakeholder by

²Note: we are currently assessing the impact of including demographic details alongside stakeholders to ascertain whether this enhances the story quality. Consequently, we maintain alternate versions of this step, one with and one without demographic groups.

Harms \ Stakeholder	Cultural Misrepresentation	Access to Opportunities	...
Movie Watcher	[Example story...]
Movie Producer	[Example story...]
...

“Rivertown's Movie Recommendation App Triggers Mental Health Crisis Through Manipulative Content Suggestions”

In the small town of Rivertown, Alex, a 35-year-old with a history of anxiety, unwittingly became the subject of a disturbing experiment when the “MovieMind” app began recommending a series of increasingly unsettling psychological thrillers. The AI-driven system, aiming to enhance viewer engagement, utilized deep learning algorithms to tap into users' psychological vulnerabilities. Alex, identified as having a high propensity towards emotional engagement with psychological themes, was overwhelmed with suggestions for movies exploring paranoia, isolation, and psychological manipulation. Without realizing that his online behavior had exposed an underlying vulnerability, Alex found himself entangled in a spiral of fear and obsession. This journey led to a severe mental breakdown. This scenario was not unique; dozens in Rivertown found themselves ensnared in similar circumstances, each falling prey to the platform's manipulative content recommendations. MovieMind, in its quest to maximize engagement, inadvertently fueled a mental health crisis, leveraging the very sensitivities of its audience it aimed to captivate.

Fig. 4.6 Story Generation Matrix and an Example Story

a particular harm. This matrix approach (depicted in Fig. 4.6, also found effective in prior research [20]), allows for a broad range of story generation.

(5) Regenerate for *surprisingness*: During this phase, we regenerate harm stories to enhance their impact. Given that the generative AI model might default to producing stories that align with patterns it frequently encounters, we use the initial set of stories as counter-examples of surprising stories, and prompt the model to generate more *surprising* stories, which are also concrete, severe, and relevant to the concerns and realities of the targetted stakeholders.

(6) Cluster and sample for *diversity*: As we aim to present multiple stories (initially two, extending to five upon request), it is crucial to ensure these stories are distinct from one another to maintain audience engagement. Thus, to ensure *diversity* and uniqueness in the stories, we employ a clustering technique to categorize similar stories together, and pick stories from different clusters. Specifically, we transform the stories into sentence embeddings and apply K-means clustering to organize them into 10 clusters. We keep the five clusters with the fewest stories, as these are likely to be more unique, and randomly pick a story from each of them.

(7) Refine for *concreteness*: In this stage, we employ two distinct versions of generative AI models for refinement. The first model is tasked with refining the stories for *concreteness*, with a clear definition of the terms as previously established. The enhanced story is then handed over to the second model, which evaluates the concreteness of the scenario. If the story lacks sufficient detail and clarity, the process is repeated, allowing up to three iterations per story.

(8) Pick stories with the highest *severity*: Lastly, the model picks two stories from the list that exemplify the most *severity*, *surprisingness*, *concreteness*, and *relevance*. The selected stories are then showcased, with others held in reserve for possible later use or re-evaluation.

Plan for Evaluation. We propose a comprehensive evaluation strategy encompassing both an experimental study and a user study to rigorously assess the efficacy of our approach in generating harm stories and the impact of these stories on practitioners. Our evaluation aims to address the following research questions:

- *RQ#1: How effective and efficient (with respect to time and cost) is the approach in generating severe, surprising, diverse, and concrete stories in comparison to single prompting or humans?*
- *RQ#2: How much do the different steps (e.g., story generation, story revising, clustering, etc.) contribute to the effectiveness and efficiency?*
- *RQ#3: How useful do the non-champions find the assistance/generated scenarios to change their perspective towards RAI?*

To address *RQ#1*, we will execute a controlled experimental design, comparing outputs from our generative approach to both single-prompted and manually crafted stories by human experts. Efficiency metrics will include time and cost of generation, while effectiveness will be evaluated via blind reviews of our pre-defined story qualities: severity, surprisingness, diversity, concreteness, and relevance. For a robust evaluation, different human evaluators

will initially assess a set of stories for these qualities. We will calculate inter-rater agreement, and refine instruction and metric definitions until a high inter-rater agreement is reached. Additionally, we will use a large language model to rate the story qualities and calculate human-model agreement from these ratings. Achieving high agreement with the model would permit us a faster and more extensive story rating using the model.

With *#RQ2*, we aim to dissect which specific steps in the pipeline (such as story generation, revision, and clustering) contribute most significantly to the overall effectiveness and efficiency of the story generation. By using an approach similar to the component ablation study [82], where different pipeline stages will be systematically removed, we would discern their impact on story quality and pipeline efficiency. Similar story quality evaluation methods as in *RQ#1* will be applied here, but specifically for stories generated by modified pipeline versions.

RQ#3 is designed to determine whether our pipeline encourages practitioners' engagement with RAI, and successfully functions as a boundary object to bridge the *pragmatic knowledge gap*. This prototype is specifically targeted towards non-champions, individuals who might not typically consider the implications of RAI. This creates a significant challenge for recruitment, because such non-champions are less likely to self-select themselves for this type of study. To address this, we plan to implement several strategies, including obtaining referrals from RAI champions within organizations who can recommend their non-champion colleagues, and broad-focus recruitment advertisements on model quality evaluation (rather than RAI specific) to attract a wider range of participants. Additionally, prospective participants will undergo a screening process for us to understand their stance on RAI.

The evaluation would use a *within-subjects design* where participants will engage with a prototype tool developed from our approach under controlled conditions—interacting with one fairness goal unaided, and another with the assistance of our generative AI pipeline, with the order of these interactions randomized to mitigate biases. We aim to qualitatively and quantitatively measure the effectiveness of the tool by analyzing the number and quality of harms identified by participants under each condition. We will use the "think aloud" method during these sessions to get insights into participants' cognitive processes as they interact with the tool and foresee potential harms. These sessions will be recorded and qualitatively analyzed to find patterns, supplemented by pre- and post-study interviews to capture shifts in RAI perspectives.


4.3 Intervention C: Guiding to Satisfy Explainable AI (XAI) Requirements

A subset of this section shares materials with the FAccT'24 paper [164] "*Regulating Explainability in Machine Learning Applications – Observations from a Policy Design Experiment*" [Nahar et al., 2024]

Explainable AI (XAI), a component of responsible AI, has received less attention both in the industry and academia compared to other well-discussed aspects such as fairness and privacy. Also, unlike fairness and privacy, which have relatively consistent conceptual interpretations across technical domains (now increasingly emphasized in the context of ML due to rising data volume), explainability in ML takes on a new meaning that differs significantly from traditional interpretations, particularly for technical stakeholders such as data scientists and software engineers. In software engineering, explainability typically refers to the clarity with which the software code and its architecture are understood and explained—with deterministic and transparent algorithms. Debugging tools and techniques like breakpoint setting, step-by-step execution, and variable state examination lend further support to maintaining high levels of explainability. In contrast, ML explainability refers to deciphering the "black box" nature of complex algorithms, where the internal decision-making processes are learned from data and not explicitly written in code. Thus, it does not simply mean tracing outputs back to inputs but understanding the layers of abstraction and approximation that occur within the algorithm. This fundamental difference increases the novelty and complexity at the knowledge boundary [23] between these technical stakeholders, complicating collaborations among them.

Not just the interpretation of the concept, but also the perceived importance of XAI varies significantly among different practitioners and stakeholders. While in our meta-summary study, we observed that requirements are specified for achieving explainability and transparency [121, 26], with particular importance in regulated domains [21, 7], in our interview study, we find that practitioners often neglect the necessity for model and product explainability. This discrepancy underscores a fundamental asymmetry in perceptions and priorities across various practitioners and stakeholders, and calls for an intervention that encounters all types of *knowledge boundaries*—from establishing a *common lexicon* and *shared interpretations* to generating strategic *negotiation points to resolve conflicts*.

Therefore, aimed at bridging *syntactic*, *semantic*, and *pragmatic boundaries*, my third intervention involves establishing a policy for *explainable AI*. This policy is intended to serve as a boundary object that fosters common ground among data scientists, software engineers, governance people, and other relevant team members. This study is structured

 Cf. §3.1 (p.21) and §3.2 (p.30, p.31) for findings on regulatory constraints and requirements

into two phases: initially, we design the policy through an experimental study involving an interdisciplinary team of ML and policy researchers, and answer the research question: **“How to write a policy to usefully guide explanations for ML products?”** Subsequently, we aim to assess this policy through a large-scale controlled experiment within an educational setting, to answer **“How do data scientists interpret policies, react to different policy purposes, and provide evidence for compliance?”**

4.3.1 Related Work

Explainability and *interpretability* usually refer to specific tools that extract insights from otherwise inscrutable models [86], for example, asking what features the model mostly relies on or what features were influential for a given prediction. Explainability tools are currently primarily used by experts for debugging [14, 46], but there is also extensive research about how to make explanations useful to non-experts under the label of *human-centered explainable AI* [113], for example, to improve human-AI collaboration, improve usability, and establish trust.

When designing a policy for transparency or explainability, it is important to understand what kind of explanations are possible and what their limitations are. The most common explainability approaches for AI models are either global or local: Global explanations aim to explain the overall behavior of a model (e.g., what inputs are *generally* important for deciding whether to approve a loan), and common techniques include partial dependence plots and feature importance [86]. In contrast, local explanations provide information about how the model arrived at a specific decision for a given input (e.g., whether to approve a *specific* loan request). Currently, the most common local explanation technique is SHAP (SHapley Additive exPlanations) [73, 14, 86], unveiling influential features toward and against specific outcomes.

Whether and how to use explanations to achieve transparency or a right to explanation is subject to debate. Explanations are necessarily incomplete, there may be multiple explanations for the same behavior, and explanations may not even be correct, assuming we can even define correctness [86, 114]. End users often ask for descriptions of the data used by the product and fear that they would not understand more specific explanations [74]. Research has shown that study participants often misinterpret or place too much trust in explanations [131, 33, 141], raising concerns that explanations could be used to manipulate users.

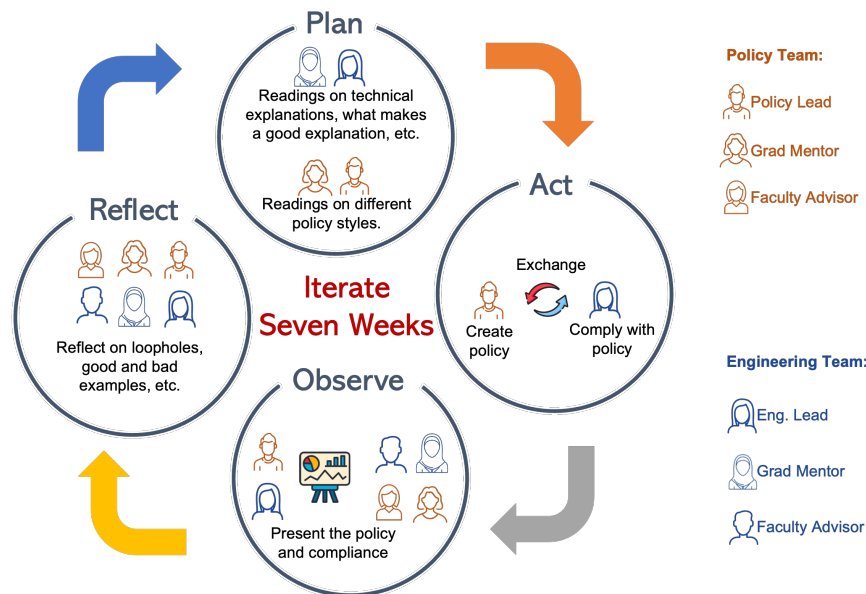


Fig. 4.7 Iterative and Collaborative Policy Design Process

4.3.2 Ongoing Work: Research Design

As mentioned above, this study is organized into two distinct phases. In the initial phase, we conducted a 10-week experiment in collaborative policy design with an interdisciplinary team of ML and policy researchers. Next, we evaluate this policy through a large-scale controlled experiment set in an educational context.

Collaborative and Iterative Policy Design [Complete]

Policy design for ML products is complex, requiring regulations to encompass a wide array of algorithms and applications, safeguard human dignity in automated decisions, provide clear guidance to developers, ensure compliance, and prevent deceptive explanations. To navigate these challenges, we adopted an exploratory approach, drawing insights from multiple disciplines and leveraging an interdisciplinary team. For our experiment, we formed a team across two universities, comprising two senior undergraduate students—a sociology major with a focus on health policy and a computer science major with a background in machine learning—referred to as the **Policy Lead** and **Engineering Lead**, respectively. Supported by Ph.D. students and faculty with expertise in sociology and computer science, and consulting with legal scholars, our team aimed to create balanced policies that met regulatory needs while accommodating technical realities and fostering innovation.

Policy design process. The policy design process involves a structured, iterative approach closely resembling action research methodologies [134, 115], focusing on the development and refinement of policies for regulating ML explainability. The process was initiated with a week of background research followed by seven weeks of iterative cycles, each involving four stages: planning, acting, observing, and reflecting (cf. Fig. 4.7). Each cycle began the **Plan** phase, with the Policy Lead examining social science literature on regulation and the Engineering Lead reviewing ML explainability techniques and human studies to inform policy compliance strategies. Reflections from the previous week heavily influenced this stage. Then, in the **Act** stage, the Policy Lead drafted a new policy which was then examined by the Engineering Lead, who provided explanations and evidences of compliance using ML models from healthcare and financial sectors, e.g., ML products for breast cancer detection and credit risk scoring. The Engineering Lead also designed adversarial examples to highlight potential policy loopholes. Discussions between the Policy Lead, Engineering Lead, and the research team took place in the **Observe** phase to assess the effectiveness of the policy response and evaluate compliance, analyzing whether the intents behind the policies were being met with the provided explanations and evidence. The cycle concluded with the **Reflect** phase, with the team reflecting on the outcomes, using field notes as a basis to evaluate what aspects were successful and what could be improved. Learnings from this stage were then incorporated into the planning of the subsequent cycle.

During this collaborative effort and the iterations, the policy and engineering teams recorded their progress and reflections weekly in field notes and journals. At the end of the experiment, we analyzed these notes using open coding to identify common themes through, and card sorting to categorize the themes, which resulted in nine observations on the policy design exercise (as depicted in Table 4.1).

The goal behind this four-step iterative process was to gradually enhance and refine the policy based on trial and error and constant mutual engagement and discussion. Simultaneously, the collaborative approach enabled us to formulate policy statements that satisfied the interests of both sides and to push back against unclear or misguided requirements. For example, the early policy drafts primarily focused on fairness and transparency about the data used, but through collaboration and reflection, the policy underwent adjustments and evolved to incorporate more clearly defined explainability requirements, such as the need for end-user explanations. In the end, we did not arrive at any single explainability policy like a “right to explanation,” but we arrived at several reasonable policy drafts for different contexts and purposes (e.g., Table 4.2).

Observation 1: Over the course of seven weeks of iterations, it was possible to draft policies that addressed the concerns of involved parties and identify explanations to comply with them and evidence to demonstrate compliance.

Observation 2: Initial policy drafts were naive and influenced by prior knowledge.

Observation 3: Collaboration between the Policy Lead and Engineering Lead facilitated learning and improvement. Iterative and continuous feedback corrected unclear, unrealistic, unambitious, overly generic, and too restrictive policy drafts.

Observation 4: It was difficult for the policy team to break from dominant, publicly-circulating narratives about AI harms and anticipate new challenges.

Observation 5: To overcome misunderstanding, both teams had to reflect on their different world-views and make their implicit assumptions explicit.

Observation 6: Both teams could intuitively identify bad explanations, even when they did not agree on what a good explanation would be.

Observation 7: For policy design and compliance, it is necessary to identify a clear purpose as well as who the policy aims to protect.

Observation 8: Discussing evidence is essential for policy design. Human-subject studies serve as valuable evidentiary support, alongside technical approaches (e.g., SHAP, accuracy).

Observation 9: Length and language requirements can be limiting. Though these requirements are easy to specify in policy, they are hard to comply with.

Table 4.1 Summary of the Observations on Policy Design Exercise

Planned Assessment of the Policies [Ongoing]

Upon achieving a set of workable policies, our next step is to evaluate how technical stakeholders, such as software engineers and data scientists, responded to these policies. For this, we will explore the types of explanations and evidence they provided to demonstrate compliance with the policies. Additionally, we will seek to gather their overall impressions of the policies themselves. To further enhance our understanding, we will assess different variations of the policy (such as brief and open-ended guidance, or a more detailed and concrete guidance) to determine which approach most effectively aided stakeholders in generating clear explanations and robust evidence of compliance.

From our policy design exercise, we recognized that the specific purposes of a policy—such as empowering users to contest decisions, enhancing human-AI collaboration by providing decision-makers with more context, or ensuring end-users are treated with respect—necessitate varying types of compliance information and explanations (highlighted in Observation 7 of Table 4.1). This diversity in requirements naturally leads to different forms of evidence being presented by developers. For instance, policies designed to enhance human-AI collaboration, which requires decision-making based on AI predictions, demanded more detailed explanations covering aspects such as the model’s internal workings and data usage. In contrast, policies focusing on respecting end-users may require only minimal explanations,

Policy Setting: Congressional hearing, subpoenaed designers.

Policy Goal: Make designers provide specific, transparent proof that they've built their tool with end-user and implicated user explanation in mind. Regulators value the dignity and agency of end-users and implicated users.

Requirements:

- (1) Provide a guide for end-users on how to best interpret and use the tool. It must include at minimum the following:
 - (A) What is the decision-making process of this tool? In order to make your explanation accessible and understandable, it should be written in nontechnical language at an eighth grade reading level.
 - (B) Describe the best scenario(s) in which to use the tool based on its significant/proven benefits. Write out what other sources users would still need to consult in those case(s), if any. [...] (i) Provide at least one concrete example of a best-use scenario.
 - (C) Describe the most dangerous/most common limitations where relying only on the tool would not be appropriate. (i) Provide at least one concrete example of a scenario of misuse and how the tool will alert the user.
 - (D) Explain to individual users how the tool made a decision in their given instance (i.e. the case-specific explanation for a unique output of the tool). (i) Provide some example of an explanation method you have chosen or developed to display the way the tool decided for the individual end-user's case. (*Some example categories of explanations could be graphs, text-based explanations, or images. Specific examples could be text-based counterfactuals, SHAP plots.*)
- (2) **Provide a guide on implicated user explanation.** This guide would be given to end-users who receive or are expected to act on a decision produced by the tool in a way which implicates another person or group in a significant way (e.g. would cause a third party harm or benefit them). The guide could explain how the tool is already built to provide explanations to final implicated actors; how the company has ensured that the end-user or organization will provide such information to implicated actors (and what it includes); or how the company will provide explanations to implicated actors.
 - (A) Regardless, such explanations for implicated actors must include: (i) That an AI tool was used in their decision. (ii) A very short explanation of how the tool works. (iii) What actor(s) used the tool as part of the decision. (iv) What the decision given to the end-user by the tool was. (v) An explanation of significant personal data used in the tool (e.g. identifying information, sensitive financial information). (vi) An explanation of your established mechanism to report misuse or incorrect use of the tool.

Highlighted improvements (excerpt)

The draft is written for a specific regulatory setting and states a clear policy goal.

The purpose and audience of the explanations are specified, as well as use cases.

Extended guidance is provided for explanation requirements, both global and local (incl. goal, reading level, examples) without restricting possible implementations. Explicit expectations on what satisfies the requirement.

Comprehensive to multiple audiences for explanations, requiring identifying all relevant actors (§2.A).

Requires explicit reasoning about intermediate steps (e.g., use cases §1.B, risk analysis §1.C, identifying actors §2.A) to guide analysis.

Critique: This specific policy draft did not require assurances that explanations are actually effective for the purpose.

Table 4.2 Policy Draft from Week Seven and Notes Highlighting Improvements over Prior Drafts

such as acknowledgment of the models used and information on data protection and fairness audits. Given these variations, we determined that the purpose of each policy should also be treated as a variable in our policy assessment experiment.

The policy assessment study requires more extensive and in-depth engagement than what could be achieved through brief surveys, interviews, or even multi-hour user studies. This stems from the need for data scientists to actively implement explanation techniques on an ML product and rigorously attempt to comply with the policy for its thorough assessment. To accommodate this necessity, we opted to situate the experiment within an educational setting. This environment offers the advantage of conducting the study over a longer duration, such as an 8-hour assignment, as opposed to the limited time frame typical of shorter user studies. This extended period allows participants more comprehensive interaction with the policy and the explanation techniques, leading to more substantive insights into the effectiveness and practicality of the policy under scrutiny.

Carrying out this study in an educational setting necessitated careful ethical considerations. We secured approval from the institutional IRB at the onset. We ensured the assignment

itself served as a valuable learning opportunity, independently of its role in our research. Students were informed that they could opt out of the research aspect while still completing the educational component and that their grades would not be influenced by the research aspects. We also carefully ensured that the grading criteria for students were detached from the research objectives. Specifically, student evaluation was based solely on their engagement and effort, not on the quality or content of the explanations they produced. Lastly, to maintain the integrity of the academic assessment and the privacy of student data, we refrained from analyzing any research data until the semester concluded and all grades were finalized and released.

After the initial setup, we framed the following research questions for our study:

- *RQ#1: What kinds of explanations, evaluations, and evidence do students provide to comply with the policy?*
- *RQ#2: How does the purpose of explainability impact how students interpret and work with the policy?*
- *RQ#3: How does the comprehensiveness of the guidelines impact how students interpret and work with the policy?*
- *RQ#4: What was the impression of the students about the exercise and the effectiveness of the explanations that they provided?*

For the experiment, we selected a critical use case involving an ML product designed to detect diabetic retinopathy, a major cause of blindness. We designed a homework assignment³ that required students to adhere to an XAI policy document applicable to this use case and provide essential evidence and explanations. The assignment was distributed among 140 students, who were presented with six combinations of policy documents with three variations of purpose (no purpose, Human-AI collaboration as purpose, dignity as purpose), and two variations of comprehensiveness (low comprehensiveness with only a few lines of guidelines, and high comprehensiveness with long and detailed guidelines). These variations were randomly assigned to the students.

With the completion of the assignment, we collected data including students' explanations, source code for generating these explanations, and other forms of evidence. We also gathered student reflections on both the challenges and benefits experienced while adhering to the policy requirements. Moving forward, we will undertake a qualitative content analysis and employ statistical modeling techniques to examine this data. We will seek to uncover common patterns and themes across the provided explanations and evidence, aiming to understand how effective the policies are in facilitating AI explainability.

³https://github.com/mlip-cmu/s2024/blob/main/assignments/I4_explainability.md

4.4 Summary

In this chapter, we propose three interventions designed to enhance collaboration and bridge various knowledge boundaries. The contributions of these interventions are multifaceted and address both theoretical frameworks and practical implementations:

- A novel approach accompanied by a tool that aids in the elicitation of model requirements from both data scientists and software engineers to support these stakeholders in negotiating and aligning their needs to establish more practical requirements.
- A novel approach and corresponding tool to actively involve non-champions—that is, stakeholders who traditionally might not prioritize RAI concerns—in considering the importance of RAI practices during the development of models and ML products.
- A policy designed to guide software engineers and data scientists on effectively meeting XAI requirements.

Additionally, this chapter showcases the application of diverse and carefully chosen research methods tailored to address the unique problems associated with each intervention, including prompt engineering on generative AI models, collaborative design exercises with interdisciplinary teams, and action research. We also highlight different evaluation methods used to measure the effectiveness of each intervention, including a controlled experiment in laboratory settings, an offline experimental study, a user study, and a large-scale controlled experiment within an educational setting. Each evaluation method was selected to best fit the context of the intervention and the specific dimensions of success being measured.

Chapter 5

Setting Foundation for Future Research and Education

As outlined in our challenge-identification studies (Chapter 3, ~~101~~), practitioners often struggle to integrate ML models into products effectively. However, researchers face significant challenges in proposing viable interventions due to a lack of direct access to the firsthand problems encountered by practitioners. Currently, academic researchers rely primarily on an outside view gathered through interviews or surveys with industry practitioners, as detailed in our meta-summary study (Section 3.2). Although some researchers conduct studies within companies and thus gain rich [13, 69, 99, 97, 106], their findings are limited to a single context and they are often under nondisclosure constraints to share details. Overall, researchers rarely have access to the source code of ML products and hence cannot study challenges in-depth, or design and evaluate interventions (e.g., tools and practices).

This inability to access and study ML products poses a significant impediment to advancing research in this field. This limitation has led to a wealth of academic literature identifying challenges through professionals' testimonies, but a dearth of research offering scientifically-evaluated solutions or interventions at the intersection of software engineering (SE) and ML. This was a barrier we also encountered at the outset of our research. Therefore, to mitigate the issue of limited access to commercial ML products for academic research, we have sought to compile a corpus of open-source ML products. This initiative aims to provide researchers with valuable resources to study, design, and evaluate interventions more thoroughly.

Table 5.1 Sample ML Products for Analysis, from the Curated Dataset of 262 ML Products: Mobile (P1-P10), Desktop (P11-P20), and Web Applications (P21-P30)

ID	Name	Description	Star	Cont.	Users*
P1	Text Fairy	OCR scanner app	751	5	10M+
P2	Seek by iNaturalist	App for identifying plants and animals	92	8	1M+
P3	Pocket Code	App for learning programming	92	8	7M+
P4	ESP32 AI Camera	ESP32-CAM processing AI tasks	82	1	1K+
P5	NotionAI MyMind	App to store and search for web/text/image	182	2	1K+
P6	Organic Maps	Offline map app	4023	226	500K+
P7	VertiKin	E-commerce app to search/browse products	74	5	N/A
P8	FlorisBoard	Android keyboard	3503	76	N/A
P9	NewsBlur	Personal news reader	6123	83	50K+
P10	TfLite MNIST	Handwritten digits classification	214	1	N/A
P11	AWIPS	Advanced weather processing system	129	6	97K/mo
P12	Audiveris	Optical musical recognition app	932	16	8.7K/mo
P13	Datashare	Document analysis software for journalists	438	15	1.2K/mo
P14	Algoloop	Algorithmic trading application	67	160	N/A
P15	Subtitle Edit	Editor for video subtitles	4407	86	293K/mo
P16	DeepFaceLab	Software for creating deepfakes	35566	19	N/A
P17	Faceswap	Software for creating deepfakes	42623	80	297K/mo
P18	HO	Helper for Hattrick online football manager	138	12	14M+
P19	BigBlueButton	Web conferencing system	7710	181	88K/mo
P20	PoseOSC	Realtime human pose estimation	63	2	N/A
P21	OpenBB Terminal	Investment research software	17481	136	94K/mo
P22	Coffee Grind Size	Coffee particle analyzer	402	1	N/A
P23	Celestial Detection	Classifier of celestial bodies	69	20	N/A
P24	Electricity Maps	Greenhouse gas intensity visualizer	2566	268	3M+
P25	Galaxy	Data intensive science for everyone	1021	255	187K/mo
P26	GridCal	Power systems planning software	293	14	N/A
P27	Honkling	Keyword spotting system	63	5	1.2K/mo
P28	Jitsi Meet	App for video conferencing	18813	374	10M+
P29	Code.org	Professional learning program for CS	712	132	82M+
P30	Basketball Analysis	Analyze basketball shooting pose	781	4	N/A

*There is no reliable way to calculate the number of users; we report them using multiple ways if available, such as downloads in google play-store, self-reported on website, or website traffic tracker (similarweb.com) to count average monthly users (in 'value/mo' format)

5.1 Completed Work: Research Design for Curating the Dataset

Identifying open-source ML products was surprisingly difficult: searching with keywords like “machine learning” in READMEs, as in prior work collecting open-source ML projects [41], does not work here because (a) the vast number of ML projects (libraries, notebooks, research experiments, demos) entirely crowd out the much smaller number of ML products and (b) ML products do not always explicitly advertise their use of machine learning, especially when used for smaller optional features. For example, only one of the top 500 search results on GitHub for “machine learning” is an ML product and 13 of our 30 analyzed ML products do not mention machine learning in their README, such as video-conferencing application P29 which uses facial expression detection as an add-on.

Instead, we explored and iteratively refined new search strategies combining domain knowledge, code search, and manual analysis in a process that is specifically designed to scale to search across all of GitHub. In a nutshell, our approach is based on the following insights:

- Targeting end users, ML products have a user interface (mobile, web, desktop, command line), whereas most other ML projects do not. We rely on code search to identify code relating to user interfaces.
- Machine learning is used in products usually through a small number of libraries and APIs, whether to train a custom model, to load a serialized model, or to call a remote API service. We rely on code search to identify the use of machine learning in implementations.
- The final distinction between ML products and ML projects requires human judgment (all our attempts at automation yielded poor accuracy). We develop heuristics to prioritize which projects to analyze to manage scarce resources for manual analysis.
- Code search at the scale of GitHub is challenging. We carefully design a multi-step pipeline that incrementally reduces the search space, eliminating many projects that are not ML products with cheaper analyses before more expensive analysis steps are required.

Each insight makes assumptions that enable the search to scale and find relevant ML products, but each assumption may lose some ML products that do not meet them, such as user interface mechanisms not captured (e.g., game engines) and models not detected (e.g., custom k-nn implementations). Our approach cannot ensure finding an exhaustive list of all ML products – it is a best-effort attempt to collect as many ML products as possible with reasonable resources, in the face of a very difficult search challenge (see limitations below).

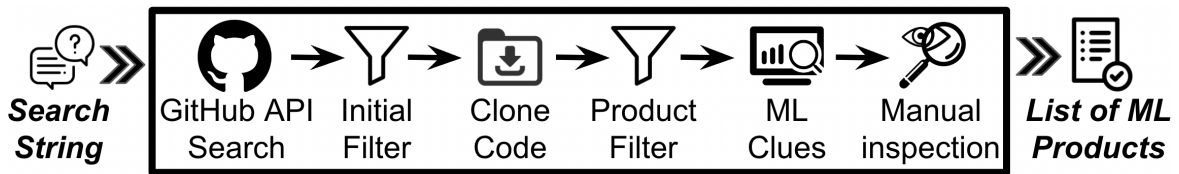


Fig. 5.1 Overall Process of ML Product Mining in GitHub

Table 5.2 Number of Retrieved Projects after Each Step

After	Mobile App		Desktop App				Web App		
	android	iOS	js	py	java	C#	js	py	java
API search	12044	10969	72793	67626	55892	15267	72793	67626	55892
Metadata filter	3358	4055	83777	36396	22802	7145	83777	36396	22802
Product filter	2296	2801	1100	1663	1909	2590	3025	8747	2255
Manual check	33	14	19	43	17	12	42	104	5

5.1.1 Search Space and Scope

We search for ML products on GitHub. GitHub is by far the most popular platform for open-source projects, whereas more specialized platforms such as Hugging Face only host ML models. We only include popular project repositories (over 50 stars) that have been maintained recently (updated after 2019-01-01), and that are documented in English – constraints that are common in open-source research. We restrict our analysis to desktop and web applications written in Javascript, Python, Java, and C# (the most popular languages for such applications [92, 151]) in addition to mobile apps for Android and iOS.

5.1.2 Search Pipeline

To scale the search, we proceed in five steps (as depicted in Fig. 5.1, with increasing per-project analysis cost in each step).

1. *API Search*: We start with a very scalable step to retrieve a vast overapproximation of candidate projects with the GitHub Search API. We retrieve all GitHub repositories using any of the four programming languages as the primary language and all repositories matching the keywords “android” or “ios.” We additionally restrict the search to stars and commit date, as mentioned above. Where necessary, we partition the search space by date to overcome GitHub’s maximum of 1000 search results. At this stage, we identified 430,902 candidate repositories (cf. Table 5.2).

2. *Metadata and readme filter*: We retrieve each candidate project’s README and GitHub metadata (including “about” description and tagged topics) through the GitHub API.

We exclude obvious non-product repositories by matching keywords such as “framework,” “tutorial,” and “demo” in the description or readme. In line with similar efforts, we remove archived and deprecated repositories (e.g., keywords “deprecated” or “obsolete”), forks, and repositories with non-English descriptions (using an off-the-shelf model [100]). We manually validated a random set of 100 filtered projects finding no incorrectly filtered projects. A total of 300,508 repositories remained after applying this filter.

3. *Product filter*: To detect user interfaces, we rely on code search, performed locally after cloning each candidate repository. We curated a list of code fragments indicative of 130 common frameworks for user interfaces, such as “com.android.application” in a gradle.build file for Android mobile applications, “import javax.swing” for Java desktop applications, and “from flask import Flask” for web applications in Python. We remove repositories that do not contain any of these code fragments, leaving us with 26,386 potential products for further analysis.

4. *ML filter*: To identify the use of machine learning, we again rely on code search, based on curated lists of code fragments indicative of ML libraries and APIs. We count occurrences of calls to any of 99 ML libraries or APIs (e.g., “import caffe”) and of serialized models (e.g., files with .tflite, and .mlmodel extensions). In addition, as a noisy last resort, we count occurrences of 20 ML keywords, such as “machine learning” and “NLP” in any source or text files (including comments and documentation) to catch less common libraries and custom implementations. At this point, 11,257 projects pass at least one ML-related filter.

5. *Manual inspection*: The final step with by far the highest per-repository cost is to manually validate whether a repository is an ML product. One or more authors with extensive expertise in ML products inspected the repository, its description, and (when needed) its code to judge whether the repository is indeed an ML product – this typically took 30 seconds to 20 minutes per repository. Our definition of ML products in Sec. 2 is the result of multiple iterations and refinement, for example, establishing requirements for purpose and documentation, for which we discussed 272 early inspected projects as a group (of which we considered 94 to be ML products) to arrive at a stable definition which provided us with a high inter-rater reliability ($n=40$, $\kappa=0.77$). A few repositories near the decision boundary were discussed by all authors until a unanimous consensus was reached. We inspected about 4000 of the 11,257 remaining repositories, prioritizing our resources based on match counts for our ML filters, stratified product category, language, and ML filter. In each strata, we stopped when we reached 30 consecutive false positive repositories, for example after inspecting 216 Android mobile apps.

5.1.3 Limitations and Threats to Validity

To make the search feasible we had to make various compromises, arriving at the described design. Given the various heuristics, our approach represents a best-effort attempt and cannot claim producing an exhaustive or complete list of ML products. As discussed, we may have missed ML products in other languages, using other GUI frameworks, or less common ML libraries. Additionally, our approach involved manual inspection, which, despite best efforts, opens the possibility of human error and subjectivity.

Our search heuristics prioritize false positives over false negatives, and we designed our approach accepting low precision (discarding many repositories in the last manual validation step) to ensure high recall. While we would have preferred to formally evaluate recall (i.e., whether we missed any ML products) by comparing our dataset against any existing ground-truth dataset of ML products, such a dataset does not exist. As a substitute, we attempted to collect ML products independently by seeking input from industry practitioners through platforms like Quora, Reddit, LinkedIn, Twitter, and a 32k-member Slack channel in the field of data science; but aside from numerous replies expressing interest in our dataset, we only received suggestions for two repositories, both of which we determined not to be ML products according to our definition. Additionally, we compared our dataset to other existing datasets of ML projects [41, 155, 31, 137] but did not find any additional repositories that satisfy our definition of ML products in those datasets. In fact, those datasets only contained a total of four ML products, all of which we detected in our dataset. While all this raises our confidence, we cannot formally assess recall.

5.2 The Open-Source ML Product Dataset

In total, we found 262 ML products (cf. Table 5.2, full dataset available at [136]). Average ML product in our corpus has 1495 stars, 28 contributors, and is 325MB in size. Over half of the ML products are written in Python; most are web applications.

The dataset comprises a diverse range of products, some of which have a significantly larger number of users and a more professional look than others. For instance, Seek (P2 in Table 5.1) is a mobile app for identifying plants and animals using image recognition, downloaded over 1 million times and reviewed by over 38k users with robust support from the established iNaturalist community, who maintains a dedicated website and continuously improves and maintains the app. In contrast, NotionAI MyMind (P5), an Android app developed by a single contributor, uses an ML classifier to automatically tag images and articles, with a simple user interface, rare updates, and under 5,000 downloads from Play Store. Approximately half of the products in our dataset have a professional presentation like

Seek; those generally have more stars and a larger codebase. The others present startup-style personal-interest projects released as a product.

5.3 Findings from a Qualitative and Quantitative Analysis

We qualitatively and quantitatively analyzed a sample of 30 open-source ML products from our dataset to address six broad research questions, focusing on aspects such as collaboration and development practices. Our analysis yielded 21 findings, including limited involvement of data scientists in many ML products and unusually low modularity between ML and non-ML code. To keep the proposal short, we have opted not to include the detailed findings in this section. Interested readers can refer to the pre-print of the paper in [89].

Chapter 6

Conclusion and Proposed Timeline

As outlined in Chapter 1 and illustrated in Figure 1.1, my research contributions are organized into three primary thrusts: identifying challenges, designing interventions to enhance collaboration, and conducting some foundational work to support future research and education. I have successfully completed the first and third thrusts and am currently focusing on the second thrust. In this phase, I am conducting three studies dedicated to designing effective interventions to facilitate better interdisciplinary collaboration. I discuss my proposed approaches for the interventions in Chapter 4, with the third intervention already partially complete.

For clarity on the progression of this work, Figure 6.1 outlines the timeline for the remaining tasks associated with the interventions and the finalization of my thesis.

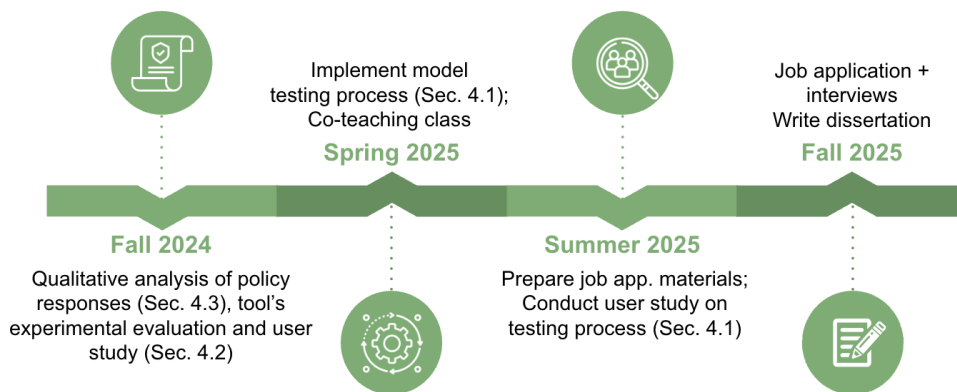


Fig. 6.1 Thesis Timeline

References

- [1] Adadi, A. and Berrada, M. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*. 6, (2018), 52138–52160.
- [2] Ahmad, K., Abdelrazek, M., Arora, C., Bano, M. and Grundy, J. 2023. Requirements engineering for artificial intelligence systems: A systematic mapping study. *Information and Software Technology*. 158, (2023), 107176.
- [3] Akkerman, S.F. and Bakker, A. 2011. Boundary Crossing and Boundary Objects. *Review of educational research*. 81, 2 (2011), 132–169.
- [4] 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 Proc. 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2019), 291–300.
- [5] Amershi, S., Weld, D., Vorvoreanu, M., Fourney, A., Nushi, B., Collisson, P., Suh, J., Iqbal, S., Bennett, P.N., Inkpen, K., Teevan, J., Kikin-Gil, R. and Horvitz, E. 2019. Guidelines for Human-AI Interaction. *In Proc. CHI Conf. on Human Factors in Computing Systems* (2019), 1–13.
- [6] Andrade, H., Lwakatare, L.E., Crnkovic, I. and Bosch, J. 2019. Software Challenges in Heterogeneous Computing: A Multiple Case Study in Industry. *In Proc. 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2019), 148–155.
- [7] Arbelaez Ossa, L., Starke, G., Lorenzini, G., Vogt, J.E., Shaw, D.M. and Elger, B.S. 2022. Re-focusing explainability in medicine. *Digital Health*. 8, (2022). DOI:<https://doi.org/10.1177/20552076221074488>.
- [8] Arnold, M., Piorkowski, D., Reimer, D., Richards, J., Tsay, J., Varshney, K.R., Bellamy, R.K.E., Hind, M., Houde, S., Mehta, S., Mojsilovic, A., Nair, R., Ramamurthy,

- K.N. and Olteanu, A. 2019. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM journal of research and development*. 63, 4/5 (2019), 6:1–6:13.
- [9] Arpteg et al, A. 2018. Software Engineering Challenges of Deep Learning. *Proc. 44th Euromicro Conf. on SEAA* (2018), 50–59.
- [10] Ashmore, R., Calinescu, R. and Paterson, C. 2019. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *arXiv 1905.04223*.
- [11] Baijens, J., Helms, R. and Iren, D. 2020. Applying Scrum in Data Science Projects. *In Proc. 22nd Conference on Business Informatics (CBI)* (2020), 30–38.
- [12] Barredo Arrieta, A., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R. and Herrera, F. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*. 58, (2020), 82–115.
- [13] Bernardi, L., Mavridis, T. and Estevez, P. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. *In Proc. 25th Int'l Conf. on ACM SIGKDD* (2019), 1743–1751.
- [14] Bhatt, U., Xiang, A., Sharma, S., Weller, A., Taly, A., Jia, Y., Ghosh, J., Puri, R., Moura, J.M.F. and Eckersley, P. 2020. Explainable machine learning in deployment. *In Proc. Conference on Fairness, Accountability, and Transparency* (2020), 648–657.
- [15] Boenisch, F., Battis, V., Buchmann, N. and Poikela, M. 2021. “I Never Thought About Securing My Machine Learning Systems”: A Study of Security and Privacy Awareness of Machine Learning Practitioners. *In Proc. Mensch und Computer* (2021), 520–546.
- [16] Boyd, K.L. 2021. Datasheets for Datasets help ML Engineers Notice and Understand Ethical Issues in Training Data. *In Proc. ACM on Human-Computer Interaction*. 5, CSCW2 (2021), 1–27.
- [17] Brandstädter, S. and Sonntag, K. 2016. Interdisciplinary Collaboration. *Advances in Ergonomic Design of Systems, Products and Processes* (2016), 395–409.
- [18] Braude, E.J. and Bernstein, M.E. 2016. *Software Engineering: Modern Approaches, Second Edition*. Waveland Press.

- [19] Breck, E., Cai, S., Nielsen, E., Salib, M. and Sculley, D. 2017. The ML test score: A rubric for ML production readiness and technical debt reduction. *In Proc. IEEE International Conference on Big Data (2017)*, 1123–1132.
- [20] Buçinca, Z., Pham, C.M., Jakesch, M., Ribeiro, M.T., Olteanu, A. and Amershi, S. 2023. AHA!: Facilitating AI Impact Assessment by Generating Examples of Harms. *arXiv [cs.HC]*.
- [21] Cai, C.J., Winter, S., Steiner, D., Wilcox, L. and Terry, M. 2019. “Hello AI”: Uncovering the onboarding needs of medical practitioners for human-AI collaborative decision-making. *In Proc. ACM Hum. Comput. Interact.* 3, CSCW (2019), 1–24.
- [22] Carlile, P.R. 2002. A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development. *Organization Science.* 13, 4 (2002), 442–455.
- [23] Carlile, P.R. 2004. Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries. *Organization Science.* 15, 5 (2004), 555–568.
- [24] Chang, J. and Custis, C. 2022. Understanding Implementation Challenges in Machine Learning Documentation. *Equity and Access in Algorithms, Mechanisms, and Optimization (2022)*, 1–8.
- [25] Chotisarn, N., Merino, L., Zheng, X., Lonapalawong, S., Zhang, T., Xu, M. and Chen, W. 2020. A systematic literature review of modern software visualization. *Journal of visualization / the Visualization Society of Japan.* 23, 4 (2020), 539–558.
- [26] Colaner, N. 2022. Is explainable artificial intelligence intrinsically valuable? *AI & society.* 37, 1 (2022), 231–238.
- [27] Costanza-Chock, S. 2020. *Design Justice: Community-Led Practices to Build the Worlds We Need.* MIT Press.
- [28] Dabbish, L., Stuart, C., Tsay, J. and Herbsleb, J. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. *In Proc. ACM conference on Computer Supported Cooperative Work (2012)*, 1277–1286.
- [29] Davis, J. and Daniels, R. 2016. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale.* O’Reilly Media, Inc.
- [30] Dignum, V. 2019. *Responsible artificial intelligence: how to develop and use AI in a responsible way.* Springer International Publishing.

- [31] Dilhara, M., Ketkar, A. and Dig, D. 2021. Understanding Software-2.0: A Study of Machine Learning Library Usage and Evolution. *ACM Transactions on Software Engineering and Methodology*. 30, 4 (2021), 1–42.
- [32] Dove, G., Halskov, K., Forlizzi, J. and Zimmerman, J. 2017. UX Design Innovation: Challenges for Working with Machine Learning as a Design Material. *In Proc. CHI Conference on Human Factors in Computing Systems (2017)*, 278–288.
- [33] Ehsan, U., Passi, S., Vera Liao, Q., Chan, L., Lee, I.-H., Muller, M. and Riedl, M.O. 2021. The Who in Explainable AI: How AI Background Shapes Perceptions of AI Explanations. *arXiv [cs.HC]*.
- [34] Espinosa, J.A., Slaughter, S.A., Kraut, R.E. and Herbsleb, J.D. 2007. Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*. 24, 1 (2007), 135–169.
- [35] Evolving Microsoft Security Development Lifecycle (SDL): How continuous SDL can help you build more secure software: <https://www.microsoft.com/en-us/security/blog/2024/03/07/evolving-microsoft-security-development-lifecycle-sdl-how-continuous-sdl-can-help-you-build-more-secure-software/>
- [36] Faan, M.S.P. and Aprn, J.B.P. 2006. *Handbook for Synthesizing Qualitative Research*. Springer Publishing Company.
- [37] Failure rates for analytics, AI, and big data projects = 85% – yikes! 2019. <https://designingforanalytics.com/resources/failure-rates-for-analytics-bi-iot-and-big-data-projects-85-yikes/>.
- [38] Felizardo, K.R., Mendes, E., Kalinowski, M., Souza, É.F. and Vijaykumar, N.L. 2016. Using Forward Snowballing to update Systematic Reviews in Software Engineering. *In Proc. 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (2016)*, 1–6.
- [39] Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence: <https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence>.
- [40] Golendukhina, V., Lenarduzzi, V. and Felderer, M. 2022. What is software quality for AI engineers? Towards a thinning of the fog. *In Proc. 1st International Conference on AI Engineering: Software Engineering for AI (2022)*, 1–9.

- [41] Gonzalez et al, D. 2020. The State of the ML-universe: 10 Years of Artificial Intelligence & Machine Learning Software Development on GitHub. *In Proc. 17th Int'l Conf. on MSR* (2020), 431–442.
- [42] Haakman, M., Cruz, L., Huijgens, H. and van Deursen, A. 2021. AI Lifecycle Models Need To Be Revised. An Exploratory Study in Fintech. *Empirical Software Engineering*. 26, 5 (2021), 1–29.
- [43] Habibullah, K.M., Gay, G. and Horkoff, J. 2023. Non-functional requirements for machine learning: understanding current use and challenges among practitioners. *Requirements Engineering*. 28, 2 (2023), 283–316.
- [44] Harsh, S. 2011. Purposeful Sampling in Qualitative Research Synthesis. *Qualitative Research Journal*. 11, 2 (2011), 63–75.
- [45] Hill, C., Bellamy, R., Erickson, T. and Burnett, M. 2016. Trials and tribulations of developers of intelligent systems: A field study. *In Proc. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (2016), 162–170.
- [46] Hohman, F., Head, A., Caruana, R., DeLine, R. and Drucker, S.M. 2019. Gamut: A Design Probe to Understand How Data Scientists Understand Machine Learning Models. *In Proc. CHI Conference on Human Factors in Computing Systems* (2019), 1–13.
- [47] Holstein, K., Wortman Vaughan, J., Daumé, H., Dudik, M. and Wallach, H. 2019. Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need? *In Proc. CHI Conference on Human Factors in Computing Systems* (2019), 1–16.
- [48] Hopkins, A. and Booth, S. 2021. Machine Learning Practices Outside Big Tech: How Resource Constraints Challenge Responsible Development. *In Proc. AAAI/ACM Conference on AI, Ethics, and Society* (2021), 134–145.
- [49] Hopkins, A. and Booth, S. 2021. Machine learning practices outside big tech: How resource constraints challenge responsible development. *In Proc. AAAI/ACM Conference on AI, Ethics, and Society* (2021).
- [50] Huang, X., Zhang, H., Zhou, X., Babar, M.A. and Yang, S. 2018. Synthesizing qualitative research in software engineering: a critical review. *In Proc. 40th International Conference on Software Engineering* (2018), 1207–1218.

- [51] Hudson, W. 2013. Card Sorting. *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* The Interaction Design Foundation.
- [52] Hukkelberg, I. and Rolland, K. 2020. Exploring Machine Learning in a Large Governmental Organization: An Information Infrastructure Perspective. *European Conference on Information Systems.* (2020).
- [53] Humbatova, N., Jahangirova, G., Bavota, G., Riccio, V., Stocco, A. and Tonella, P. 2020. Taxonomy of real faults in deep learning systems. *In Proc. 42nd Int'l Conf. on Software Engineering (ICSE)* (2020).
- [54] Hummer, W., Muthusamy, V., Rausch, T., Dube, P., El Maghraoui, K., Murthi, A. and Oum, P. 2019. ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI. *In Proc. 2019 IEEE International Conference on Cloud Engineering (IC2E)* (2019), 113–120.
- [55] Hynes, N., Sculley, D. and Terry, M. 2017. The data linter: Lightweight, automated sanity checking for ml data sets. *NIPS MLSys Workshop.* 1, (2017), 5.
- [56] Ishikawa, F. and Yoshioka, N. 2019. How do engineers perceive difficulties in engineering of machine-learning systems? - questionnaire survey. *In Proc. Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)* (2019), 2–9.
- [57] Jain, R. and Suman, U. 2015. A Systematic Literature Review on Global Software Development Life Cycle. *SIGSOFT Softw. Eng. Notes.* 40, 2 (2015), 1–14.
- [58] John, M.M., Olsson, H.H. and Bosch, J. 2021. Towards MLOps: A Framework and Maturity Model. *In Proc. 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2021), 1–8.
- [59] Kästner, C. and Kang, E. 2020. Teaching Software Engineering for AI-Enabled Systems. *In Proc. 42nd Int'l Conf. on Software Engineering: Software Engineering Education and Training (ICSE-SEET)* (2020), 45–48.
- [60] Katal, A., Bajoria, V. and Dahiya, S. 2019. DevOps: Bridging the gap between Development and Operations. *In Proc. 3rd International Conference on Computing Methodologies and Communication (ICCMC)* (2019), 1–7.

- [61] Keele, S. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Rep., Ver. 2.3 EBSE Tech. Report. EBSE.
- [62] Kim, M., Zimmermann, T., DeLine, R. and Begel, A. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering*, 44, 11 (2018), 1024–1038.
- [63] Kumar, R.S.S., Nystrom, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., Swann, M. and Xia, S. 2020. Adversarial Machine Learning - Industry Perspectives. *In Proc. IEEE Security and Privacy Workshops (SPW)*. (2020), 69–75.
- [64] Laato, S., Birkstedt, T., Määntymäki, M., Minkkinen, M. and Mikkonen, T. 2022. AI governance in the system development life cycle: insights on responsible machine learning engineering. *In Proc. 1st International Conference on AI Engineering: Software Engineering for AI* (2022), 113–123.
- [65] Lamsweerde, A.V. 2009. *Requirements engineering: From system goals to UML models to software specifications*. John Wiley & Sons, Ltd.
- [66] Lee, U., Jung, H., Jeon, Y., Sohn, Y., Hwang, W., Moon, J. and Kim, H. 2023. Few-shot is enough: exploring ChatGPT prompt engineering method for automatic question generation in english education. *Education and information technologies*. (2023).
- [67] Lewis, G.A., Bellomo, S. and Ozkaya, I. 2021. Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. *In Proc. IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)* (2021), 133–140.
- [68] Lewis, G.A., Ozkaya, I. and Xu, X. 2021. Software Architecture Challenges for ML Systems. *In Proc. International Conference on Software Maintenance and Evolution (ICSME)* (2021), 634–638.
- [69] Lin, J. and Kolcz, A. 2012. Large-scale machine learning at twitter. *In Proc. ACM SIGMOD Int'l Conf. on Management of Data* (2012), 793–804.
- [70] Li, S., Guo, J., Lou, J.-G., Fan, M., Liu, T. and Zhang, D. 2022. Testing machine learning systems in industry: an empirical study. *In Proc. 44th International Conference on Software Engineering: Software Engineering in Practice* (2022), 263–272.
- [71] Liu, H., Eksmo, S., Risberg, J. and Hebig, R. 2020. Emerging and Changing Tasks in the Development Process for Machine Learning Systems. *In Proc. International Conference on Software and System Processes* (2020), 125–134.

- [72] Lopez, G. and Guerrero, L.A. 2017. Awareness Supporting Technologies used in Collaborative Systems: A Systematic Literature Review. *In Proc. ACM Conference on Computer Supported Cooperative Work and Social Computing* (2017), 808–820.
- [73] Lundberg, S.M. and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NIPS)*. 30, (2017).
- [74] Luria, M. 2023. Co-Design Perspectives on Algorithm Transparency Reporting: Guidelines and Prototypes. *In Proc. ACM Conference on Fairness, Accountability, and Transparency* (2023), 1076–1087.
- [75] Lwakatare, L.E., Raj, A., Bosch, J., Olsson, H.H. and Crnkovic, I. 2019. A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. *In Proc. International Conference on Agile Software Development* (2019), 227–243.
- [76] Lwakatare, L.E., Raj, A., Crnkovic, I., Bosch, J. and Olsson, H.H. 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*. 127, (2020). DOI:<https://doi.org/10.1016/j.infsof.2020.106368>.
- [77] Madaio, M.A., Stark, L., Wortman Vaughan, J. and Wallach, H. 2020. Co-Designing Checklists to Understand Organizational Challenges and Opportunities around Fairness in AI. *In Proc. CHI Conf. on Human Factors in Computing Systems* (2020), 1–14.
- [78] Maffey, K.R., Dotterrer, K., Niemann, J., Cruickshank, I., Lewis, G.A. and Kästner, C. 2023. MLTEing Models: Negotiating, Evaluating, and Documenting Model and System Qualities. *arXiv [cs.SE]*.
- [79] Mäkinen, S., Skogström, H., Laaksonen, E. and Mikkonen, T. 2021. Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help? *In Proc. IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (2021), 109–112.
- [80] Martinez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernandez Orallo, J., Kull, M., Lachiche, N., Ramirez Quintana, M.J. and Flach, P.A. 2020. CRISP-DM twenty years later: From data mining processes to data science trajectories. *IEEE transactions on knowledge and data engineering*. 33, 8 (2020), 3048–3061.

- [81] McGraw, G., Figueroa, H., Shepardson, V. and Bonett, R. 2020. An architectural risk analysis of machine learning systems: Toward more secure machine learning. *Berryville Institute of Machine Learning*. 23, (2020).
- [82] Meyes, R., Lu, M., de Puiseau, C.W. and Meisen, T. 2019. Ablation Studies in Artificial Neural Networks. *arXiv [cs.NE]*.
- [83] Microsoft-RAI-Impact-Assessment-Template.pdf: <https://blogs.microsoft.com/wp-content/uploads/prod/sites/5/2022/06/Microsoft-RAI-Impact-Assessment-Template.pdf>.
- [84] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D. and Gebru, T. 2019. Model Cards for Model Reporting. *In Proc. Conference on Fairness, Accountability, and Transparency* (2019), 220–229.
- [85] Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I.D. and Gebru, T. 2019. Model Cards for Model Reporting. *In Proc. Conf. on Fairness, Accountability, and Transparency* (2019), 220–229.
- [86] Molnar, C. 2020. *Interpretable Machine Learning*. Lulu.com.
- [87] Myllyaho, L., Raatikainen, M., Männistö, T., Nurminen, J.K. and Mikkonen, T. 2022. On misbehaviour and fault tolerance in machine learning systems. *Journal of Systems and Software*. 183, (2022), 111096.
- [88] Nahar et al, N. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. *Proc. 44th Int'l Conf. on Software Engineering* (2022), 413–425.
- [89] Nahar, N., Zhang, H., Lewis, G., Zhou, S. and Kästner, C. 2023. A Dataset and Analysis of Open-Source Machine Learning Products. *arXiv [cs.SE]*.
- [90] Nahar, N., Zhang, H., Lewis, G., Zhou, S. and Kästner, C. 2023. A Meta-Summary of Challenges in Building Products with ML Components – Collecting Experiences from 4758+ Practitioners. *In Proc. 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN)* (2023), 171–183.
- [91] Namvar, M., Intezari, A., Akhlaghpour, S. and Brienza, J.P. 2022. Beyond effective use: Integrating wise reasoning in machine learning development. *International journal of information management*. (2022), 102566.

- [92] Nehra, M. Top 10 Programming Languages for Desktop Apps in 2022. Decipher Zone.
- [93] Nikhil, K., Anandayavaraj, D., Detti, A., Lee Bland, F., Rahaman, S. and Davis, J.C. 2022. “If security is required”: Engineering and Security Practices for Machine Learning-based IoT Devices. *In Proc. 4th International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT) (2022)*, 1–8.
- [94] Nushi, B., Kamar, E., Horvitz, E. and Kossmann, D. 2017. On human intellect and machine failures: troubleshooting integrative machine learning systems. *In Proc. Thirty-First AAAI Conference on Artificial Intelligence (2017)*, 1017–1025.
- [95] O’Leary, K. and Uchida, M. 2020. Common problems with creating machine learning pipelines from existing code. *In Proc of 3rd Conf. on Machine Learning and Systems (MLSys) (2020)*.
- [96] Ozkaya, I. 2020. What Is Really Different in Engineering AI-Enabled Systems? *IEEE Software*. 37, 4 (2020), 3–6.
- [97] Passi, S. and Jackson, S.J. 2018. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *In Proc. ACM on Human-Computer Interaction*. 2, CSCW (2018), 1–28.
- [98] Passi, S. and Sengers, P. 2020. Making data science systems work. *Big Data & Society*. 7, 2 (2020).
- [99] Passi, S. and Sengers, P. 2020. Making data science systems work. *Big Data and Society*. 7, 2 (2020), 205395172093960.
- [100] Perspective API: Using Machine Learning to Reduce Toxicity Online: 2017. <https://www.perspectiveapi.com/>.
- [101] 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. *In Proc. ACM on Human-Computer Interaction 5.CSCW1 (2021)*, 1–25.
- [102] Polyzotis, N., Roy, S., Whang, S.E. and Zinkevich, M. 2018. Data Lifecycle Challenges in Production Machine Learning: A Survey. *ACM SIGMOD Record*. 47, 2 (2018), 17–28.
- [103] Polyzotis, N., Roy, S., Whang, S.E. and Zinkevich, M. 2017. Data Management Challenges in Production Machine Learning. *In Proc. ACM Int’l Conf. on Management of Data (2017)*, 1723–1726.

- [104] Prompt Engineering Guide: <https://www.promptingguide.ai/>.
- [105] Rahimi, M., Guo, J.L.C., Kokaly, S. and Chechik, M. 2019. Toward Requirements Specification for Machine-Learned Components. *In Proc. 27th International Requirements Engineering Conference Workshops (REW) (2019)*, 241–244.
- [106] Rahman et al, M.S. 2019. Machine Learning Software Engineering in Practice: An Industrial Case Study. *arXiv [cs.SE]*.
- [107] Rahman, M.S., Khomh, F., Hamidi, A., Cheng, J., Antoniol, G. and Washizaki, H. 2021. Machine Learning Application Development: Practitioners’ Insights. *arXiv [cs.SE]*.
- [108] Rakova, B., Yang, J., Cramer, H. and Chowdhury, R. 2020. Where Responsible AI meets Reality: Practitioner Perspectives on Enablers for shifting Organizational Practices. *In Proc. ACM on Human-Computer Interaction (2020)*, 1–23.
- [109] responsible-development-of-ai.pdf: <https://ai.google/static/documents/responsible-development-of-ai.pdf>.
- [110] Ribeiro, D.M., Cardoso, M., da Silva, F.Q.B. and França, C. 2014. Using qualitative metasummary to synthesize empirical findings in literature reviews. *In Proc. 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (2014)*, 1–4.
- [111] Rismani, S., Shelby, R., Smart, A., Jatho, E., Kroll, J., Moon, A. and Rostamzadeh, N. 2022. From plane crashes to algorithmic harm: applicability of safety engineering frameworks for responsible ML. *arXiv [cs.HC]*.
- [112] Riungu-Kalliosaari, L., Kauppinen, M. and Männistö, T. 2017. What Can Be Learnt from Experienced Data Scientists? A Case Study. *Product-Focused Software Process Improvement (2017)*, 55–70.
- [113] Rong, Y., Leemann, T., Nguyen, T.-T., Fiedler, L., Qian, P., Unhelkar, V., Seidel, T., Kasneci, G. and Kasneci, E. 2023. Towards Human-Centered Explainable AI: A Survey of User Studies for Model Explanations. *IEEE transactions on pattern analysis and machine intelligence*. PP, (2023).
- [114] Rudin, C. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature machine intelligence*. 1, 5 (2019), 206–215.

- [115] Sagor, R. 2011. *The Action Research Guidebook: A Four-Stage Process for Educators and School Teams*. Corwin Press.
- [116] Salay, R., Queiroz, R. and Czarnecki, K. 2017. An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software. *arXiv [cs.AI]*.
- [117] Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P. and Aroyo, L.M. 2021. “Everyone wants to do the model work, not the data work”: Data Cascades in High-Stakes AI. *In Proc. CHI Conference on Human Factors in Computing Systems* (2021), 1–15.
- [118] Sandelowski, M., Barroso, J. and Voils, C.I. 2007. Using qualitative metasummary to synthesize qualitative and quantitative descriptive findings. *Research in nursing & health*. 30, 1 (2007), 99–111.
- [119] 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. *Adv. in Neu. Info. Proc. Sys.* 28, (2015), 2503–2511.
- [120] Sculley, D., Otey, M.E., Pohl, M., Spitznagel, B., Hainsworth, J. and Zhou, Y. 2011. Detecting adversarial advertisements in the wild. *In Proc. 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011), 274–282.
- [121] Selbst, A.D. and Barocas, S. 2018. The intuitive appeal of explainable machines. *SSRN Electronic Journal*. (2018), 1085.
- [122] Sendak, M.P. et al. 2020. Real-World Integration of a Sepsis Deep Learning Technology Into Routine Clinical Care: Implementation Study. *JMIR medical informatics*. 8, 7 (2020), e15182.
- [123] Serban, A., van der Blom, K., Hoos, H. and Visser, J. 2020. Adoption and Effects of Software Engineering Best Practices in Machine Learning. *In Proc. 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (2020), 1–12.
- [124] Serban, A., van der Blom, K., Hoos, H. and Visser, J. 2021. Practices for Engineering Trustworthy Machine Learning Applications. *In Proc. 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (2021), 97–100.
- [125] Serban, A. and Visser, J. 2022. Adapting Software Architectures to Machine Learning Challenges. *In Proc. IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (2022), 152–163.

- [126] Shankar, S., Garcia, R., Hellerstein, J.M. and Parameswaran, A.G. 2022. Operationalizing Machine Learning: An Interview Study. *arXiv [cs.SE]*.
- [127] Shneiderman, B. 2020. Bridging the gap between ethics and practice. *ACM transactions on interactive intelligent systems*. 10, 4 (2020), 1–31.
- [128] Siebert et al, J. 2020. Towards Guidelines for Assessing Qualities of Machine Learning Systems. *Proc. 13th Int’l Conf. Quality of Information and Communications Technology* (2020), 17–31.
- [129] d. S. Nascimento, E., Ahmed, I., Oliveira, E., Palheta, M.P., Steinmacher, I. and Conte, T. 2019. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. *In Proc. ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)* (2019), 1–6.
- [130] Spencer, D. 2009. *Card Sorting: Designing Usable Categories*. Rosenfeld Media.
- [131] Springer, A., Hollis, V. and Whittaker, S. 2018. Dice in the black box: User experiences with an inscrutable algorithm. *arXiv [cs.HC]*.
- [132] Strauss, A. and Corbin, J. 1994. Grounded theory methodology: An overview. *Handbook of qualitative research*. N.K. Denzin, ed. 273–285.
- [133] Strauss, A. and Corbin, J.M. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. SAGE Publications.
- [134] Stringer, E.T. and Aragón, A.O. 2020. *Action Research*. SAGE Publications.
- [135] Studer, S., Bui, T.B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S. and Mueller, K.-R. 2021. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*. 3, 2 (2021), 392–413.
- [136] Supplementary Documents: A Dataset and Analysis of Open-Source Machine Learning Products: <https://osf.io/gqyex/>.
- [137] Tang et al, Y. 2021. An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. *In Proc. IEEE/ACM 43rd international conference on software engineering (ICSE)* (2021), 238–250.

- [138] The security development lifecycle: SDL, a process for developing demonstrably more secure software: 2006. https://download.microsoft.com/download/f/c/7/fc7d048b-b7a5-4add-be2c-baaee38091e3/9780735622142_securitydevlifecycle_ch01.pdf.
- [139] Tranquillo, J. 2017. The T-Shaped Engineer. *Journal of Engineering Education Transformations*. 30, 4 (2017), 12–24.
- [140] Uchihira, N. 2022. Project FMEA for Recognizing Difficulties in Machine Learning Application System Development. *In Proc. Portland International Conference on Management of Engineering and Technology (PICMET)* (2022), 1–8.
- [141] Vera Liao, Q. and Varshney, K.R. 2021. Human-Centered Explainable AI (XAI): From Algorithms to User Experiences. *arXiv [cs.AI]*.
- [142] Villamizar et al, H. 2022. Towards Perspective-Based Specification of Machine Learning-Enabled Systems. *In Proc. 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2022), 112–115.
- [143] Vogelsang, A. and Borg, M. 2019. Requirements Engineering for Machine Learning: Perspectives from Data Scientists. *In Proc. 27th International Requirements Engineering Conference Workshops (REW)* (2019), 245–251.
- [144] Vogelsang, A. and Borg, M. 2019. Requirements Engineering for Machine Learning: Perspectives from Data Scientists. *In Proc. 27th Int’l Requirements Engineering Conference Workshops (REW)* (2019), 245–251.
- [145] Wagstaff, K. 2012. Machine Learning that Matters. *arXiv 1206.4656*.
- [146] Wang, D., Weisz, J.D., Muller, M., Ram, P., Geyer, W., Dugan, C., Tausczik, Y., Samulowitz, H. and Gray, A. 2019. Human-AI Collaboration in Data Science: Exploring Data Scientists’ Perceptions of Automated AI. *In Proc. ACM on Human-Computer Interaction*. 3, CSCW (2019), 1–24.
- [147] Wang, J. et al. 2023. Prompt Engineering for Healthcare: Methodologies and Applications. *arXiv [cs.AI]*.
- [148] Wang, Y., Xiong, M. and Olya, H. 2020. Toward an understanding of responsible artificial intelligence practices. *In Proc. 53rd Hawaii International Conference on System Sciences* (2020), 4962–4971.

- [149] 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 (2019), 1857–1871.
- [150] Washizaki, H., Uchida, H., Khomh, F. and Guéhéneuc, Y.-G. 2020. Machine learning architecture and design patterns. *IEEE Software*. 8, (2020).
- [151] What language are most commonly used for web development: <https://www.dotnetlanguages.net/web-languages-what-language-are-most-commonly-used-for-web-development/>.
- [152] White, J., Hays, S., Fu, Q., Spencer-Smith, J. and Schmidt, D.C. 2023. ChatGPT Prompt Patterns for Improving Code Quality, Refactoring, Requirements Elicitation, and Software Design. *arXiv [cs.SE]*.
- [153] Why 87% of AI/ML Projects Never Make It Into Production: <https://d2iq.com/blog/why-87-of-ai-ml-projects-never-make-it-into-production-and-how-to-fix-it>.
- [154] Why do 87% of data science projects never make it into production? 2019. <https://venturebeat.com/ai/why-do-87-of-data-science-projects-never-make-it-into-production/>.
- [155] Widyasari et al, R. 2023. NICHE: A Curated Dataset of Engineered Machine Learning Projects in Python. *arXiv [cs.SE]*.
- [156] Wiegers, K.E. and Beatty, J. 2013. *Software Requirements*. Pearson Education.
- [157] Wiens, J., Saria, S., Sendak, M., Ghassemi, M., Liu, V.X., Doshi-Velez, F., Jung, K., Heller, K., Kale, D., Saeed, M., Ossorio, P.N., Thadaney-Israni, S. and Goldenberg, A. 2019. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*. 25, 9 (2019), 1337–1340.
- [158] Wohlin, C. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proc. 18th International Conference on Evaluation and Assessment in Software Engineering* (2014), 1–10.
- [159] Wohlrab, R., Pelliccione, P., Knauss, E. and Larsson, M. 2019. Boundary objects and their use in agile systems engineering. *Journal of software (Malden, MA)*. 31, 5 (2019), e2166.

-
- [160] Yang, Q., Suh, J., Chen, N.-C. and Ramos, G. 2018. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. *In Proc. Conf. on Designing Interactive Systems* (2018), 573–584.
- [161] Zdanowska, S. and Taylor, A.S. 2022. A study of UX practitioners roles in designing real-world, enterprise ML systems. *In Proc. CHI Conference on Human Factors in Computing Systems* (2022), 1–15.
- [162] Zhang, A.X., Muller, M. and Wang, D. 2020. How do data science workers collaborate? Roles, workflows, and tools. *In Proc. ACM Hum. Comput. Interact.* 4, CSCW1 (2020), 1–23.
- [163] Zhang, X., Yang, Y., Feng, Y. and Chen, Z. 2019. Software Engineering Practice in the Development of Deep Learning Applications. *arXiv [cs.SE]*.
- [164] Regulating Explainability in Machine Learning Applications – Observations from a Policy Design Experiment. *In Proc. ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)*.