Facilitating Collaboration in Building Machine Learning Products

Nadia Nahar

CMU-S3D-26-102
April 2026

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Christian Kästner, Chair
James D. Herbsleb
Claire Le Goues
Kenneth Holstein
Samir Passi (Microsoft)

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

This research was supported in part by the National Science Foundation under Grant Nos. 1813598, 2106853, and 2131477; by the Software Engineering Institute under Contract No. FA8702-15-D-0002_0003; and by PricewaterhouseCoopers through the DTIC project "Scaffolding Responsible AI Practice at the Earliest Stages of Ideation, Problem Formulation, and Project Selection" (OSR-1247). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the sponsoring organizations or the U.S. government.

Keywords: Software Engineering for Machine Learning, Machine Learning, Large Language Models, Collaboration, Responsible AI.

Abstract

In this dissertation, I investigate the collaboration challenges between software engineers and data scientists in building machine learning (ML) products, and identify 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 challenges in integrating ML into software products by synthesizing the collective knowledge of the field through a qualitative meta-summary of the academic literature, and then examine collaboration challenges–an aspect largely underexplored in prior work–through a qualitative interview study with industry practitioners; I further investigate challenges when integrating large language models into software products. I then demonstrate principles for addressing these collaboration challenges through three studies: (a) a mixed-method study examining emerging practices for evaluating large language model-generated content in software applications, (b) a set of interventions (i.e., policy guidance, role-playing chatbots, and educational support) to guide the development of explainable AI and improve transparency and interpretability of ML models in products, and (c) an intervention designed to engage practitioners in responsible AI practices, fostering ethical awareness and compliance. These interventions are designed to address the *syntactic*, *semantic*, and *pragmatic knowledge boundaries* that hinder effective teamwork in ML product development.

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

Acknowledgements

This dissertation would not have been possible without the support, mentorship, and encouragement of many people.

First and foremost, I would like to thank my advisor, Christian Kästner, who has been an advisor to me in every sense of the word. Beyond guiding this research, he supported me through many aspects of my academic and professional development. His mentorship, encouragement, and trust allowed me to grow as a researcher, and I am deeply grateful for the many conversations, ideas, and opportunities that shaped this work.

Christian consistently encouraged me to grow beyond my research. I am especially grateful for how much he supported my development as a teacher. He pointed me toward resources, pushed me to invest time in improving my teaching, and encouraged me to take this responsibility seriously—even when that sometimes meant slower research progress. His support for my mentoring of students, and the trust and autonomy he gave me in that role, helped me grow tremendously. Over the years, Christian has also helped me develop many skills that extend far beyond the scope of this dissertation—presentation and design, writing and communication, project leadership, and many others that are difficult to enumerate. He helped me learn how to drive research projects forward while also providing thoughtful direction whenever I felt stuck.

Equally important, Christian has been deeply supportive of my personal growth. Many of our meetings extended well beyond research discussions and often felt like life counseling sessions. Whenever I felt overwhelmed or uncertain, I knew I could turn to him for guidance. Without fail, I would leave those conversations feeling clearer, calmer, and more grounded. I cannot thank him enough for the patience, care, and generosity he has shown me throughout this journey. I feel incredibly fortunate to have had him as my advisor—**the best advisor I could have asked for**—and I will carry the lessons I learned from him, both in research and in life, with me throughout my career.

I would also like to thank Grace Lewis, who has been a wonderful collaborator and mentor. Since my first project in 2020, she has consistently provided thoughtful guidance, valuable feedback, and encouragement that helped shape my work. I am especially grateful for how generously she supports and champions my work among her colleagues and collaborators.

I would like to thank my informal mentors—Alka Menon, Eunsuk Kang, Bogdan Vasilescu, Rohan Padhye, and Michael Hilton—for their guidance and support throughout my PhD. Alka, Eunsuk, Bogdan, and Rohan generously provided feedback and advice that helped shape my research and thinking, while Michael was especially supportive in helping me grow as a teacher. Their doors were always open whenever I needed advice or discussion. Although I formally had a single advisor, which is somewhat uncommon in S3D, I never felt that I had only one source of guidance because of their generosity with their time and perspectives.

I would also like to thank the members of my dissertation committee—Jim Herbsleb, Claire Le Goues, Kenneth Holstein, and Samir Passi—for their thoughtful feedback and for helping strengthen the ideas presented in this dissertation.

I would like to thank my PhD buddy, Shurui Zhou, for helping me start my journey at CMU. I am also grateful to my project collaborators for the many ideas and discussions that shaped this work. I thank my students and mentees, who helped me grow as a mentor and made mentoring such a rewarding experience. I am deeply appreciative of my labmates for their camaraderie and for sharing this journey with me.

I would also like to thank the S3D administrative team for their constant support throughout my time at CMU. I am especially grateful to Jenni Cooper, Connie Herold, Helen Higgins, Dabney Schlea, Alisha Roudebush, Thomas Pope, and Cole Jester for always being helpful whenever I needed assistance. Their support behind the scenes made many aspects of this journey much smoother.

I am also deeply grateful to my S3D friends who made this journey joyful and light with countless laughs and shared moments. I especially want to thank Manisha, my best friend, who has been there for me through everything—scolding me when I needed it, comforting me when things were hard, and always caring unconditionally. I am incredibly thankful to my wonderful officemates, Leo and Andy, who always looked out for me and made the office feel like a welcoming place every day. I am also grateful to Jane, Simon, Luke, and Ao Li for their friendship since my very first days at CMU. I thank Chenyang for the many fun moments and thoughtful conversations that made this journey richer, and Jenni for her constant encouragement and positivity. I am also thankful to Kyle and Vasu for their friendship and for simply being around through the ups and downs of this journey. I would also like to thank my Pittsburgh friends—Monty, Mohona, Samia, Tarnuma, and Tanvir—for always being there for me. Your constant care, check-ins, and friendship made the long and difficult PhD journey feel lighter, warmer, and much more fun.

I would like to thank my father, whose dream this was long before it became mine. In many ways, I feel that I have simply carried forward what he envisioned for me, long before

I could even imagine it myself. He always believed in me, encouraged me to aim high, and supported me with unwavering love and pride. He gave me the wings to fly while also keeping me safe from the harshness of the world. Throughout my life, he treated me like his princess, always making me feel capable of achieving anything I set my mind to. I miss him every day, and there are so many moments in this journey when I wished I could share them with him. I know that seeing this day would have made him incredibly happy. His golden girl is now a doctor, and I know that would have meant the world to him.

I am especially grateful to my mother, who lived this entire journey with me. Her constant presence, encouragement, and care sustained me through both the challenges and the achievements of this process. She prayed for me day and night, quietly carrying my worries as if they were her own. She cared for me in the most tender ways—making sure I had food and water when I was too absorbed in work to notice my hunger and thirst, and sitting beside me through sleepless nights, gently patting me to sleep when my mind refused to rest. She encouraged me to keep going when the work felt heavy, but also reminded me to pause and take care of myself. I do not know many mothers who walk so closely beside their daughters through a PhD journey. This degree is not only mine—it belongs to her as well.

I would also like to thank my brother, who has always been one of my strongest supporters. At a time when I did not yet know what path to choose, he recognized my potential and encouraged me to pursue software engineering. In many ways, that early guidance helped set me on the path that ultimately led me here. His constant encouragement and pride in my achievements have meant more to me than I can express.

I am deeply grateful to everyone who supported me through this journey.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

In this dissertation, I study the collaboration challenges between software engineers and data scientists in building machine learning (ML) products, and I identify naturally occurring interventions from industry practitioners, as well as propose interventions to facilitate collaboration 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 [83, 182, 322, 377, 376, 278, 367], with collaboration issues as one of the major obstacles [382, 383]. 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.

To address these collaboration challenges, I identify naturally occurring industry interventions for evaluating large language model (LLM) outputs in software products and design two additional interventions: (1) a policy- and persona-based approach, including an LLM role-playing chatbot and educational support, to guide practitioners in building ML products with explainable AI, and (2) a novel approach to engage practitioners in responsible AI practices[1]. Throughout this work, I deliberately select and combine various research methods tailored to the research questions in each study.

[1]This dissertation 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.

The challenges of building products with ML components, which I refer to as *ML products* in this dissertation, 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 [315, 8, 259]. In fact, according to Gartner, an estimated 85% of machine learning projects fail to transition from prototype to production [106, 96], and a recent report from MIT indicates that 95% of generative AI pilots fail [221]. These anecdotal claims call for a systematic examination of the challenges in this domain and the development of interventions to support ML product development.

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. As AI coding assistants and agentic development tools increasingly participate in these

workflows, coordinating this broader ecosystem of contributors becomes even more complex, as well as important.

Collaboration is hard in ML product development. Collaboration in general is challenging, especially across disciplines among team members with different educational backgrounds and experiences [76, 93], 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 [77, 162] and the Security Development Lifecycle (SDL) [355, 94] to improve teamwork and establish a culture of collaboration between developers, operations teams, and security experts. These approaches already account for iterative development, continuous integration, and ongoing system evolution. Yet, the development of ML products introduces additional hurdles for teamwork, not because collaboration itself is new, but because the nature of ML systems changes some of the assumptions under which these practices operate, particularly due to the role of data and learned behavior [376, 266], 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 [57, 58] and related challenges, particularly between software engineers, who are used to working with systems whose behavior can be more explicitly specified and inspected, and data scientists, who are engaged in the exploratory development of models whose behavior is learned from data. 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. Unlike many traditional software components, where behavior is primarily determined by code, ML system behavior is also shaped by data, which may evolve over time and introduce shifts that are difficult to anticipate at design time. This creates additional collaboration challenges around how data is interpreted, curated, and maintained across roles. Additionally, while continuous updates and monitoring are not unique to ML (as seen in DevOps practices), ML systems do introduce additional complexity due to data drift, changing data distributions, and the need to continuously reassess

Fig. 1.1 Overview of Contributions

model behavior in response to new data. While some progress has been made in addressing these issues, such as the introduction of MLOps [155] to facilitate effective teamwork between 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 dissertation, where I am particularly interested in facilitating the collaboration between data scientists and software engineers in building machine learning products.

1.1 Thesis Statement

This dissertation argues that **collaboration challenges** in **machine-learning product** development arise from distinct knowledge gaps between stakeholders. I show that intervention effectiveness depends on matching the intervention to the type of gap, and that generic approaches often fail.

I support this by (a) identifying and characterizing these gaps, and (b) developing interventions that **make concepts explicit** and **use LLMs to contextualize information** across roles.

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 identify and design interventions to facilitate better collaboration. In the last thrust,

I develop foundational infrastructure to support future research and education, which is excluded from the scope of this dissertation.

I have conducted two studies in the first thrust (*Chapter 3*) 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. I report the three major collaboration points that practitioners consistently report as challenging. I also triangulate the challenges with findings from existing literature.

The identified challenges demonstrate problems at different *knowledge boundaries*, namely, *syntactic*, *semantic*, and *pragmatic* [58] (details in *Chapter 2*), and thus, to demonstrate how collaboration can be improved at these boundaries by *transferring*, *translating*, and *transforming* knowledge, I demonstrate three interventions, as outlined in the second thrust (*Chapter 4*). Each of these interventions addresses fundamental gaps in communication and collaboration, as mentioned below:

- My first intervention study surfaces empirically grounded, naturally occurring practices that industry practitioners use to evaluate large language model (LLM) outputs in real-world software products. These emerging solutions address different types of knowledge boundaries. For example, to mitigate *syntactic boundaries*, such as the challenge that defining a "*bug*" in LLM outputs is not as straightforward as in traditional software, practitioners collaboratively define new evaluation metrics through expert consultation. To address *semantic boundaries* arising from subjectivity in judging what constitutes a correct LLM response, practitioners establish explicit rubrics and scoring mechanisms. To resolve *pragmatic boundaries* related to responsible AI (RAI) practices, governance teams standardize and mandate RAI processes across the organization. These are illustrative examples; the study identifies a total of 19 practices addressing specific challenges.
- Aimed at bridging *syntactic*, and *semantic boundaries*, my second intervention involves an iterative set of experiments to support generating meaningful explanations for ML outcomes. I start by establishing a *policy* for explainable AI, to serve as a boundary object [392, 57] that fosters common ground among data scientists, governance people, and end users expectations. How-

ever, explanations generated in the initial experiments were not meaningful to their intended audiences, revealing a lack of developer empathy toward end users and demonstrating that the *semantic boundary* remains unaddressed. In response, in later experiments, I complement the policy with role-playing chatbots that embody end-user personas. This is further supported by education to guide persona development and help practitioners avoid common pitfalls. Through these iterative experiments, I provide empirical evidence that participants' ability to produce explanations that meet expected levels of explainability improves progressively across iterations.

- Finally, my third intervention is designed to encourage data scientists to engage with the principles of *responsible AI* to address the *pragmatic boundary* that arise when practitioners feel resistant or indifferent to ethical considerations in ML product development. To mitigate this, I introduce *sticky stories*: narrative descriptions of potential harms tailored to ML products. Through a user study, I provide empirical evidence that this narrative-based intervention increases practitioner engagement, and deepens consideration of ethical risks, helping align practitioner values with organizational responsible AI goals.

Throughout the document, I use side notes to cross-reference related chapters, provide additional updates or reflections beyond the original studies, and indicate the types of knowledge boundaries involved.

Example side note: AE *Syntactic knowledge boundary*. Terminology around evaluation metrics is defined in intervention §4.1.

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, we[2] 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 set of emerging practices to support evaluation of LLM-based products (Section 4.1).** Through a mixed-methods study comprising 26 interviews across 12 Microsoft product teams and a follow-up survey with 332 practitioners, I identify and validate 19 emerging practices for evaluating large language model (LLM) outputs in production software systems. These practices address evaluation challenges stemming from non-determinism, subjectivity, and the absence of clear correctness criteria in LLM-based products. The practices include defining custom evaluation metrics, combining qualitative and quantitative measures, automating offline evaluations, using LLMs as validators for subjective metrics, and establishing layered guardrails and automated checks. Together, these practices provide an empirically grounded characterization of how industry teams operationalize evaluation for LLM-integrated products and offer concrete guidance for both researchers and practitioners.
- **A set of interventions for supporting explainability in ML products (Section 4.2).** This work contributes design and empirical insights into supporting practitioners in generating meaningful explanations for ML outcomes through a large-scale controlled experiment within an educational setting. Across

[2]I use "we" instead of "I" in the rest of the dissertation

iterative experiments conducted over four academic semesters with a total of 409 students as part of an 8-hour homework assignment, I evaluate a set of complementary interventions, including an explainable AI policy, role-playing chatbots that embody end-user personas, and targeted education to support persona development. The findings show that while policy plays an important role in clarifying expectations and responsibilities, it is insufficient on its own; combining policy with persona-driven role play and education enables practitioners to produce explanations that better align with end-user needs and expected levels of explainability. Together, these interventions translate high-level explainability principles into actionable guidance for ML product development.

- **A novel intervention for responsible AI engagement (Section 4.3).** This work investigates how sticky stories—narrative descriptions of potential harms that are concrete, relevant, severe, surprising, and diverse—can be used to increase practitioner engagement with responsible AI (RAI) practices. Informed by a formative study that revealed limited motivation and shallow engagement with existing RAI processes, I design this narrative-based sticky story intervention that generates and presents harm scenarios tailored to ML products. Through a subsequent user study, I show that exposure to sticky stories leads practitioners to identify a broader range of harms, and engage more deeply with ethical risks compared to baseline approaches.

Chapter 2

Background

This chapter aims to provide the foundational knowledge necessary for understanding the subsequent discussions in this dissertation. We will delve into the key concepts surrounding machine learning (ML) products, including their development process and various qualities of concern. Given that this dissertation focuses on collaboration challenges, we also discuss different aspects of collaboration in ML product development. In particular, we introduce the concept of *knowledge boundaries*, which serves as a central analytical lens throughout this thesis for understanding the collaboration challenges and the interventions designed to address them.

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 dissertation 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.

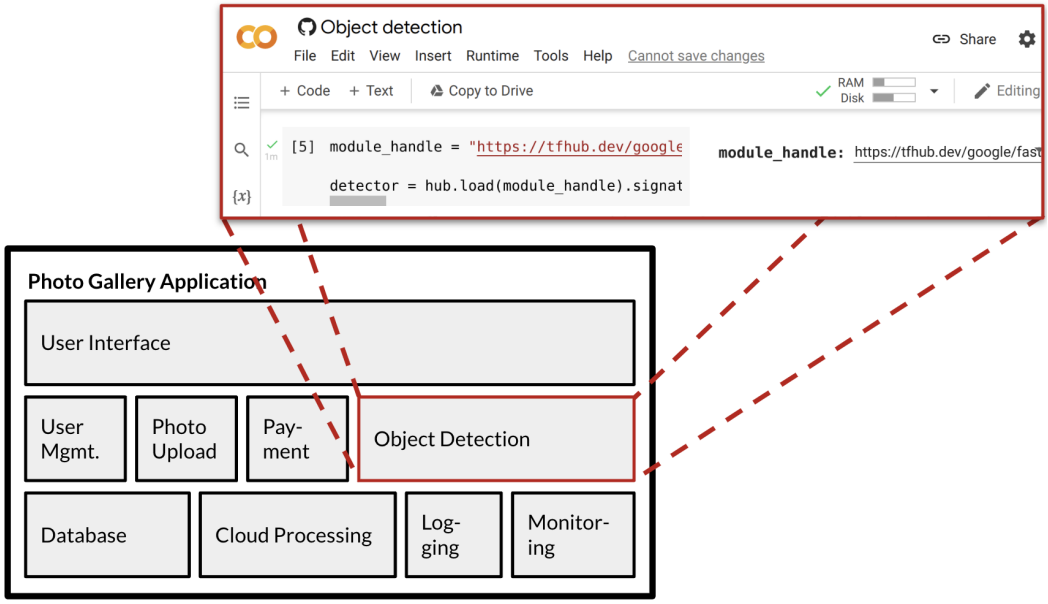


Fig. 2.1 Example of an ML Component within an ML Product

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 [366, 230], or ML systems [331, 182]. Thus, to eliminate any ambiguity in this dissertation, 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 [376, 83]. 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 [367, 278]. Similarly, ML impacts the ways systems are architectured 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 [322, 377, 182]. 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 [50, 331, 242]. In addition to traditional concerns such as performance, safety, and security, ML systems often require evaluating qualities such as fairness and explainability, which are shaped by data and context and are harder to specify and assess in advance [133, 282, 314, 209, 305, 33]. 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 [307, 270, 46, 145].

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 [266, 167, 230]. Organizational resource constraints restrict ML practitioners from achieving the model qualities they plan to provide [134, 9, 14]. The need for diverse skills in development teams requires additional training and education for both software engineers and data scientists [167, 371, 230]. 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 [46, 12, 220], but we are still lacking standardization [61]. On top of all, we still do not have a good process to integrate the data science pipeline with the software development lifecycle [115, 206, 348].

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 (*Responsible AI and Explainable AI*)

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 dissertation, 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.

2.1.3.1 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 [82, 374]. 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.

2.1.3.2 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 [21, 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 dissertation 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 (*Syntactic, Semantic, Pragmatic*)

Knowledge boundaries [57, 58] 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* (artifacts that can be interpreted and used across different communities while maintaining a shared reference) [2], and fostering environments that promote mutual learning and understanding across different knowledge domains.

There are three categories of knowledge boundaries: Syntactic, semantic, and pragmatic. **Syntactic boundary** (denoted by 🔤) is the most basic level of knowledge boundary that refers to the communication barriers among team members due to differences in technical language and terminology. For example, ML development introduces domain-specific terms such as *training data*, *model drift*, *precision and recall*, or *feature engineering*, which software engineers, project managers, or other stakeholders may not be familiar with. These differences in vocabulary can create misunderstandings even when stakeholders are otherwise aligned in their goals, making it difficult to communicate requirements, expectations, or evaluation criteria effectively. Overcoming this boundary requires *transferring knowledge* by creating common vocabularies or standardized codes that all team members can understand.

Semantic boundary (denoted by 💬) arises when stakeholders use similar terms but interpret them differently due to differences in expertise, perspectives, or goals. In ML product development, this often occurs when the same concept carries different meanings across roles. For example, the term "*performance*" may refer to prediction accuracy for data scientists, whereas software engineers typically interpret it as response time or latency. Similarly, a data scientist may evaluate model accuracy through statistical metrics such as F1 score or ROC-AUC, while a product manager or domain expert may interpret accuracy in terms of whether the system produces reliable outcomes in real-world scenarios. These differences in interpretation can lead to misaligned expectations about model quality, evaluation criteria, or the readiness of a system for deployment. Bridging this boundary may involve *translating knowledge* through shared artifacts and discussions that align stakeholders' interpretations of key concepts.

Pragmatic boundary (denoted by 🤜) is the most complex boundary, which arises when stakeholders not only interpret knowledge differently but also have different priorities, incentives, or interests that influence decision-making or practices. At this boundary, each party might recognize the other's differences but still not value them sufficiently to motivate change in their practices or beliefs. In ML product development, this can occur when different teams prioritize competing objectives. For instance, software engineers may prioritize system reliability and latency, data scientists may focus on maximizing model accuracy, while governance or compliance teams emphasize responsible AI considerations such as fairness,

transparency, or privacy. These differing priorities can create tensions when negotiating trade-offs during system design, evaluation, and deployment. 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 in software development is DevOps (discussed in detail in the next section), which merges development and operations disciplines into a cohesive process with common goals and practices.

2.2.2 Collaboration in Software Engineering

Collaboration challenges in software development have been observed for decades, often arising from conflicts between teams with differing roles, expertise, and goals. Within the software engineering community, several approaches have been developed to support collaboration and reduce these tensions. Many of these approaches can be seen as mechanisms for addressing different types of *knowledge boundaries*, for example by establishing shared representations, aligning interpretations, or reconciling competing priorities. Notably, paradigms such as DevOps and MLOps have been extensively explored to improve coordination between developers and operators, and similarly between data scientists and operators in ML systems. These methodologies address not only tooling and operational practices but also promote shared responsibility and closer interdisciplinary collaboration across the system lifecycle. In this section, we first discuss knowledge boundaries in software engineering, and then briefly discuss DevOps and MLOps as examples of collaboration-support mechanisms from which we can draw inspiration.

Knowledge Boundaries in Software Engineering Knowledge boundaries are not unique to ML systems; they have long been observed in traditional software engineering, particularly in collaborative settings involving developers, operators, designers, and domain experts. These boundaries arise when individuals or groups draw on different expertise, vocabularies, and perspectives, creating challenges in how systems are described, understood, and ultimately built.

At the **syntactic level**, challenges arise from differences in terminology. For example, software engineers may use terms such as *idempotency*, *eventual consistency*, or *race condition*, which designers, product managers, or other stake-

holders may not be familiar with. These differences in vocabulary can lead to confusion even when stakeholders are otherwise aligned in their goals, making it difficult to communicate requirements or system behavior effectively. Addressing syntactic boundaries typically involves establishing a shared vocabulary through documentation, standards, or interface definitions.

At the **semantic level**, challenges arise when stakeholders use the same terms but interpret them differently. For instance, a requirement such as "*user-friendly*" may be interpreted by developers in terms of minimizing friction and streamlining interactions (e.g., fewer steps, more automation), while designers or product managers may interpret it in terms of transparency and user control, even if that requires exposing more information or adding steps to support informed decision-making. Although the same term is used, these differing interpretations can lead to misaligned design decisions and expectations. Addressing semantic boundaries often requires aligning interpretations through shared artifacts, design discussions, and iterative refinement of requirements.

At the **pragmatic level**, boundaries emerge when stakeholders share an understanding of a concept but differ in their priorities or incentives. For example, stakeholders may agree that a system should be "*secure*," yet differ in how much priority it should receive relative to other objectives such as development speed or feature delivery. While security teams may advocate for stricter controls and longer review cycles, developers or product teams may prioritize rapid iteration to meet deadlines. These tensions arise not from misunderstanding, but from competing goals, making them more difficult to resolve. Addressing pragmatic boundaries typically requires changes in processes, incentives, or organizational structures to better align stakeholder priorities.

These examples illustrate that software engineering has developed a range of mechanisms to manage knowledge boundaries, often by making system behavior explicit, shared, and enforceable through code, interfaces, and processes.

Past Collaboration Success Stories: DevOps and MLOps 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 [415, 416]. From a knowledge boundary perspective, these approaches help address

pragmatic boundaries by aligning incentives and responsibilities across roles, while also reducing *semantic gaps* through shared practices and tooling. 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. More broadly, these approaches illustrate how addressing knowledge boundaries requires not only technical solutions, but also organizational and cultural alignment. 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 How ML Collaboration Differs from Traditional Software?

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. While collaboration is also central to traditional software development, ML product development introduces additional

challenges due to the nature of learned behavior and data dependencies. In contrast to traditional software development, where projects often aim to operate under well-defined requirements and expected behavior, ML products frequently operate under partially specified requirements 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.

Modern software engineering practices recognize that development is often iterative and uncertain (e.g., agile and DevOps methods), and provide mechanisms to manage this complexity. However, many of these mechanisms assume that system behavior can be specified, inspected, and verified with reasonable clarity. These assumptions are central to how software engineering addresses knowledge boundaries—for example, shared interfaces and APIs help resolve 🔤 *syntactic boundaries* by enforcing common representations, while documentation and specifications help align interpretations across roles. In ML product development, these assumptions are often weakened or do not fully hold. Because model behavior is learned from data rather than explicitly defined, it is often difficult to fully specify system behavior in advance, making it harder to establish stable interfaces or shared representations that traditionally help mitigate 🔤 *syntactic boundaries*. Similarly, interpreting model outputs and assessing their validity in context introduces additional 💬 *semantic* and 🤝 *pragmatic* challenges that are not easily addressed through existing mechanisms.

In ML product development, 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, where system behavior is learned rather than explicitly programmed. As a result, collaboration shifts from coordinating predefined components to jointly interpreting and shaping system behavior over time. This iterative experimentation process necessitates frequent adjustments and reinterpretation of results, often requiring coordination across multiple perspectives. Effective collaboration, therefore, becomes essential not only due to interdisciplinarity, but because stakeholders must align on how to interpret and act on system behavior that is not fully specified in advance. In this sense, ML product development does not eliminate knowledge boundaries, but reshapes them in ways that make existing solutions less effective, motivating the need for new, boundary-aware approaches to collaboration.

Chapter 3

Identifying Challenges

In this chapter, I present two studies that lay the foundation for my dissertation. The objective behind these studies was to gain a deeper understanding of the obstacles in building machine learning (ML) products, with particular attention to the collaboration challenges that arise among stakeholders throughout this process of ML product development. Understanding these challenges is essential for supporting effective collaboration among software engineers, data scientists, and other members of the development team. This work constitutes the first phase 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 synthesize prior research on practitioner challenges in ML product development through a qualitative meta-summary study. By systematically analyzing prior research, this study consolidates and interprets the collective understanding of challenges in ML product development, providing a broader perspective on the issues reported across different studies and contexts.

The second study focuses specifically on the collaboration challenges that arise in industry practice during ML product development. To investigate these challenges, I conduct a qualitative interview study with practitioners. This approach enables the collection of rich, firsthand accounts from professionals, revealing the practical dynamics and difficulties that software engineers, data scientists, and other stakeholders encounter when collaborating on developing ML products.

Additionally, as large language models (LLMs) rapidly gain popularity in software products, I also examine the challenges that arise when integrating LLM-based functionality into applications. Practitioners face new difficulties that differ from those encountered in traditional ML systems, particularly due to the open-ended and subjective nature of LLM outputs. Evaluating LLM-generated outputs has therefore emerged as a central challenge for development teams (Table 4.2). Because the practices practitioners adopt to address this challenge form the intervention studied in this dissertation, a detailed discussion is deferred to the intervention chapter (§4.1).

3.1 Identification A: Meta-Summary of Challenges in Building ML Products

> The findings of this section have been published in CAIN'23*: "A Meta-Summary of Challenges in Building Products with ML Components – Collecting Experiences from 4758+ Practitioners,"* for which I received the 🏆 **Early Career Researcher Award** [232].

While there was 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. Studying this broader literature is important because collaboration challenges rarely appear in isolation; instead, they emerge through concrete development activities such as defining requirements, evaluating models, or addressing fairness concerns.

While many of the prior studies focus on specific aspects, such as challenges regarding requirements [367, 278], or fairness [133, 282], 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

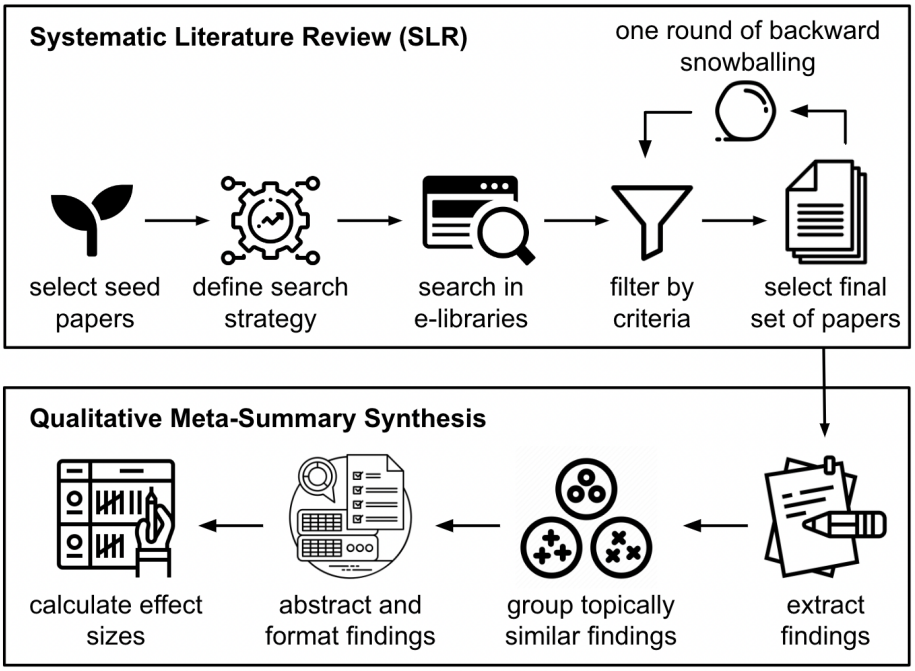


Fig. 3.1 Identification A: Research Design

catalog also enables us to triangulate the collaboration challenges identified in section 3.2 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* [308, 291, 95], we analyze, organize, and synthesize findings across all these studies (as shown in Figure 3.1), 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) organization and teams. Table 3.1 contains a summary of the findings.

3.1.1 Related Work

With the advance of ML techniques, many organizations have invested substantial efforts in building products with ML components. While there is a large amount of research that focuses entirely on the challenges that data scientists face in their model-development work (e.g., development responsibilities [167], data exploration [226, 190], data-science processes [115, 206], development in notebooks [62, 125, 265], AutoML [371]), another body of work focuses on the challenges of building products with those models, often with interdisciplinary teams, and placing substantial attention on qualities like safety and observability. The latter work, which forms the scope of this survey, moves beyond the model-centric view of classic data-science workflows and considers building automated pipelines and entire software systems with many ML and non-ML components, as well as the engineering challenges involved. It emerges in a growing research community often named *Software Engineering for Machine Learning (SE4ML)* studying the engineering challenges of building both ML components and products that contain ML components.

3.1.1.1 Understanding practitioner needs

Academic research is often criticized for being far removed from the needs faced by practitioners in industry [368, 282, 28]. If researchers want to achieve rapid impact in industry, they need to understand what problems are important to practitioners; conversely, practitioners may attempt to attract researchers to work on their problems. Attempts to close the gap between academia and practice typically need to navigate a tradeoff between (a) investigating one or few teams in depth with findings that may not generalize or (b) exploring common problems across many teams with more shallow engagements. Focused on individual teams, we see a few ethnographic studies [259, 258], many direct collaborations with an industrial partner [168, 279], and many experience reports published by practitioners in papers [30, 158, 124, 186], talks [208, 121, 102], or blog posts [408, 129, 336]. To understand problems across teams, many researchers conduct interviews across multiple

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

teams and organizations, e.g., [367, 182, 322, 230, 197], to be either addressed by the same researchers or reported as open problems to the community. Other researchers have focused on surveying practitioners at scale across companies and regions, e.g., [181, 147, 376, 404].

In this paper, we go one step further in aggregating and analyzing results from prior interviews and surveys with over 4758 practitioners, which we hope will help guide future research and educational activities toward challenges relevant to practitioners.

3.1.1.2 Previous literature reviews

There have been several prior literature reviews on topics related to building products with ML components. Most surveys review academic papers proposing solutions in subfields, such as testing ML components [403, 15, 293, 47, 137], safety and security [42, 191, 137], data management [270], and even trying to cover published research on SE4ML broadly [109, 205]. The closest to our work are two literature surveys that analyze practitioner experience reports published at academic conferences (not including grey literature) collecting the self-reported challenges of a few dozen teams [198, 250]. In this work, we specifically perform a meta-summary of academic papers reporting on interviews and surveys with practitioners.

3.1.2 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 [164]. 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 [138]. 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 [138, 291, 308]. The meta-summary method provides a well-balanced 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 [291]. Thus, we apply the meta-summary method [308, 95, 291] to perform the quantitative aggregation of the qualitative evidence that we present as findings. Figure 3.1 shows the overview of the research process followed in this study.

3.1.2.1 Paper Selection

To increase reliability, reproducibility and objectivity of the process for paper selection, we follow the established procedure of conducting systematic literature reviews [164].

Relevant years. Much of the research on engineering products with ML components was inspired by the seminal 2015 ML technical debt paper by Sculley et al. [315], which outlined various engineering challenges in building and operating ML infrastructure. For completeness, we selected the year range of the papers to be from 2010 to 2022.

Publication venues. To search for papers, we select digital libraries and databases commonly used by software engineering review papers, e.g, [150, 66, 193]. We do not filter by the venue, as we expect to find papers that are published in different communities including software engineering, human-computer interaction, and machine learning. Since we aim to aggregate results from robust empirical studies, we did not include gray literature, such as blog posts, which typically reflect opinions or individual experience only. However, we did include arXiv as a data source, as it contains many relevant academic papers in this field, even if some have not been peer reviewed. Specifically, we use the following 8 data sources: IEEE Xplore (ieeexplore.ieee.org), ACM Digital library (portal.acm.org/dl.cfm), Wiley InterScience (www.interscience.wiley.com), Elsevier Science Direct (www.sciencedirect.com), SpringerLink (www.springerlink.com), EI Compendex (www.engineeringvillage.com), and arXiv (https://arxiv.org).

Search query. 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 [100, 198]). 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.

We realized that our research question has three aspects, and therefore to retrieve the papers that would satisfy our research question, we focused on those three parts to formulate the search query: (A) **The paper needs to mention an ML-related keyword**, since we focus on challenges introduced by ML components. (B) **The paper needs to mention a software engineering or ML deployment-related keyword**, since we focus on engineering challenges that go beyond local concerns of data scientists; for example the paper should discuss concerns related to actual product development where models are deployed and incorporated into larger software systems. Finally, (C) **the paper needs to mention surveys or interviews**, since we are interested in the challenges mentioned by industry practitioners and these are the most common relevant research methods; we are not interested in a single-team case study or ethnographic study, as the challenges found in such papers may be specific to individual products.

After adding some semantically similar terminologies, we developed the following search query fragments – A: “machine learning” OR “artificial intelligence” OR “deep learning” OR “ML component” OR ”data science”; B: ”software engineering” OR “software systems” OR ”production-ready systems” OR ”ML systems” OR “deploying ML” OR “ML deployment”; C: “interview” OR “survey” OR “questionnaire”. The final query was of the following format “A AND B AND C.”

We searched with this query within the abstract of the papers in all the digital data sources except SpringerLink, as it did not have the option to search within abstracts. For SpringerLink, we retrieved 5612 papers based on a full-text search, and subsequently used a custom script to search within the abstracts of these papers. This provided us with a total of 341 papers from all the sources (see Table 3.2).

This search query retrieved 18 of the 21 seed papers. Two papers were missed because the conducted interviews were not mentioned in the abstract (the abstract framed the research as a case study), and one paper was not listed within the libraries searched (only available on TechRxiv). To account for this difference we performed one round of snowballing, as explained later in this section.

Data Source	Initial Search Result	After Filtering by Title/Abstract and Snowballing	Final Selection
IEEE	69	30	19
ACM	48	11	10
Willey	6	0	0
ScienceDirect	32	5	3
Engineer Village	101	3	0
Springer	6*	3	2
arXiv	79	8	5
Snowballing	-	26	11
Total	**341**	**86**	**50**

*abstract filtering from 5612 papers retrieved with fulltext search

Table 3.2 Paper Selection

Selection criteria. The initial search returned many papers that were not directly relevant to our research question. Next, we selected 86 relevant papers by reading the title and abstract, evaluating them against the inclusion and exclusion criteria (see Table 3.3), which we incrementally refined. Finally, we read the full paper, and once again evaluated each against the inclusion and exclusion criteria, which narrowed our set down to 39 papers. Multiple researchers participated in this process and discussed papers at the boundary.

Most of the papers that were discarded in this round were either literature surveys in the domain of *machine learning for software engineering* (i.e., using ML techniques to facilitate software engineering tasks; not relevant to this study) or used interviews or surveys to evaluate tools. We also removed papers that have a narrow focus or are entirely model-centric, e.g., interviewing only data scientists about their modeling work (e.g., [131, 151, 91, 240]) or interviewing only non-technical people (e.g., [357, 40, 127, 303]).

Snowballing. To capture relevant papers that did not match our keywords in their abstract, we performed one iteration of backward snowballing [391], which means that we went through the selected papers' reference list to find whether we missed any relevant papers. We analyzed 26 additional papers and considered 11 of them

Inclusion Criteria	
I1:	Paper includes software engineering challenges for ML systems
I2:	Paper uses interview or survey with industry practitioners (software engineers, data scientists, etc.) to identify the challenges
I3:	Paper appears in a refereed publication (including conference proceedings, journal, etc.) or uploaded in arxiv in a publication format
I4:	Paper is written in English
Exclusion Criteria	
E1:	Paper has a strict ML model view and does not consider the system or product using the model
E2:	Paper interviews/surveys only non-technical people (end-users, domain experts, etc.)
E3:	Paper focuses on ML for software engineering instead of software engineering for ML systems
E4:	Paper falls in the category of gray literature: blog post, technical report, government report, webinar, poster session, presentation, etc.

Table 3.3 Inclusion and Exclusion Criteria

as relevant based on the inclusion and exclusion criteria, which included the three papers from the seed set we previously missed.

Final paper set. Overall, our process resulted in a final set of 50 papers. Most of the papers were published recently, since 2019 (see Figure 3.2). 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 identification of challenges (or issues, problems, difficulties) in different aspects of building products with ML components. The other papers do not

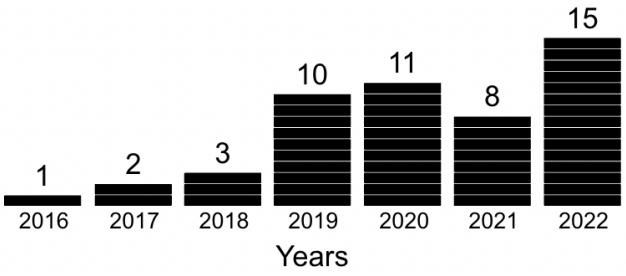


Fig. 3.2 Year Distribution of the Selected Papers

explicitly set a goal of identifying challenges but more broadly study the process of building products with ML components, yet they also report practitioner challenges in their findings.

3.1.2.2 Qualitative Meta-Summary Process

As stated earlier, we used the meta-summary research method [308, 291, 95] 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 [342, 139]. 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 [308]. 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 [133, 182]. 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").

3.1.2.3 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 [131, 9, 37, 16, 321].

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.

3.1.3 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. Also, although *MLOps*, *Fairness*, and other more specific categories are often used to organize results in the surveyed papers, we eventually settled on minimizing the number of cross-cutting topics. We decided to include operations challenges in the *Architecture and Design* group, as we consider them primarily as a *design for change* issue; and we separate and group various concerns for specific qualities, such as *fairness,* in the development stages where the concerns arise, such as requirements and quality assurance. 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.

3.1.3.1 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 [16, 322, 173, 367, 167, 203, 360, 376, 147, 230, 400, 134, 189, 235, 297, 87, 258] **(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 [147, 230] and only consider paying for coding, as they do not understand the need for experimental analysis [189] and even difficulty convincing engineering teams to invest in collecting high-quality data [167]. 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 [376, 230, 134, 235, 87, 258]. 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 [173, 400], scoping and deciding the functional and non-functional requirements [367, 189], interpreting the model outcomes [322, 235, 297], and the infrastructure needs (e.g., appropriate data, monitoring infrastructure, retraining requirements) when building products [367, 400, 235]. 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 [367, 230, 282, 33].

Also observed in our collaboration study (§3.2), reflecting a semantic knowledge boundary (💬).

Vagueness in ML problem specifications makes it difficult to map business goals to performance metrics [339, 188, 181, 322, 173, 367, 360, 376, 230, 282, 133, 189, 115, 235, 297, 197, 258] **(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

Also observed in our collaboration study (§3.2).

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 [33, 282, 133, 258]. 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) [367, 282]. 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 [322, 376, 189, 115].

An instance of semantic knowledge boundary (💬).

An instance of pragmatic knowledge boundary (🤹).

Regulatory constraints specific to data and ML introduce additional requirements that restrict development [115, 321, 367, 33, 134, 241, 322] **(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 [367, 134] 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 [321, 115, 33].

An instance of semantic knowledge boundary (💬).

3.1.3.2 Architecture, Design, and Implementation

We find that many ML practitioners struggle with designing the architecture of products with ML components.

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 [182, 322, 173, 203, 400, 189, 229, 197, 8, 225, 144] **(11/50).** Practitioners frequently report challenges in migrating from exploratory model code, often in a notebook, to deployable production-quality code in automated ML pipelines [400, 189]. Building an end-to-end ML pipeline is considered to be a

Involves both technical and collaboration issues; the paradigm shift partly introduces syntactic (🔠) and semantic (💬) knowledge boundaries.

challenge due to the difficulties of integrating various ML and non-ML components in a system operating within an environment [182, 322, 173], the overwhelming complexity of integrating many tools and frameworks [203, 197, 225], the need for engineering skills beyond the comfort zone of some data scientists [225], and so on. While practitioners emphasized the importance of pipeline automation for many projects where frequent re-training and deployment of models are needed, they also consider it time-consuming, labor-intensive, error-prone, and not well supported by current tools [322, 229, 197, 8, 144].

ML adds substantial design complexity with many, often implicit, data and tooling dependencies, and entanglements due to a lack of modularity [322, 324, 197, 339, 188, 182, 376, 400, 8, 14, 87] **(11/50).** Many practitioners report challenges from additional complexity when designing systems incorporating machine learning, and the traditional software architecture and design practices no longer fit [188, 182, 400, 87]. ML changes the assumptions in traditional software systems such as encapsulation and modularity and causes entanglements of data, source code, and ML models, which can lead to *"pipeline jungles"* and *"change anything changes everything"* integrations that are hard to maintain [322, 197, 188, 376, 324, 8]. Unlike traditional systems, ML requires the incorporation of data pipelines that need to handle a high volume of data and often data architectures of distributed nature, and practitioners also need to understand and design for the data flow in the entire system [339, 376, 400]. Practitioners also point out that complexities arise due to a large amount of surrounding "glue code" to support the ML models [322, 14], and complicated dependency and configuration management [376, 14].

Difficulty in scaling model training and deployment on diverse hardware [322, 203, 189, 320, 115, 297, 324, 197, 144, 225] **(10/50).** Practitioners commonly report difficulty dealing with cloud and computational resources, even with the recent emergence of MLOps. Practitioners find the technologies to be difficult to integrate into the production environment and require substantial time, effort, and money [203, 189, 115, 297, 197, 225]. Among the common problems of such deployments, practitioners brought up the mismatch of development and production environments [189, 324], difficulties in building a scalable pipeline [320, 115, 197, 144], adhering to serving requirements such as latency and throughput [322, 197], as well as undocumented tribal knowledge within the team, hampering future

deployments [324]. Despite the emerging MLOps tooling, practitioners still raise many questions about how to utilize those resources and sometimes express being overwhelmed by the sudden flood of tools and frameworks to choose from [167, 9].

While monitorability and planning for change are often considered important, they are mostly considered only late after launching [322, 115, 324, 144, 182, 181, 400, 8, 173, 172, 230, 14, 33, 295, 225] **(15/50).** Practitioners report struggling with monitoring their deployed models for detecting drift, bias, or even failures. While many highlight monitoring as very important, planning for monitoring is rare [230]. Even for companies that adopt a monitoring infrastructure, practitioners report struggling with ad-hoc monitoring practices of logging, creating alerts, or doing everything manually [324, 182]. Similar concerns were raised about model evolution, where practitioners acknowledge it to be important, but fall behind in planning for change in their architectural design [322, 115, 144, 8, 14]. Practitioners mentioned that ML-centric software goes through frequent revisions more than traditional software (e.g., due to model retraining, or even model replacement for data change, hyperparameter tuning, or change of domain, etc.), and the changes tend to be nontrivial and nonlocal, raising the need for an architecture that supports such changes. As a result, we find practitioners' soliciting the need for adapted architectural patterns to design for such post-launch activities for products with ML components with monitorability as a significant quality attribute [182, 400].

3.1.3.3 Model Development

Although we explicitly exclude challenges relating only to the work and tools of data scientists when building models, we find reports of engineering challenges during model development, which we report in this section.

Model development benefits from engineering infrastructure and tooling but provided infrastructure and technical support are limited in many teams [189, 115, 144, 181, 376, 173, 172, 14, 360, 167, 241, 25, 280, 404, 229, 110, 235, 9, 37] **(19/50).** ML practitioners share tooling needs for different tasks including data analysis and visualization, feature engineering, model development, integration, evaluation, deployment, monitoring, reproducibility, and support for specific qualities like privacy, security, and explainability. They report a lack of adequate tools in these areas and find the existing tools and techniques to be (a)

unavailable in their environment [115, 25], (b) not automated enough [280], (c) requiring too much expert knowledge to be used [404, 181, 241, 280], (d) limited to specific tasks and types of data sets [25, 280], or (e) not suitable for their own problems [25, 37, 167]. This raises demand for custom tools but many teams lack the resources and engineering support.

Code quality is not standardized in model development tools, leading to conflicts about code quality [324, 376, 230] **(3/50)**. Practitioners report that code quality and review processes are usually not standardized and are inconsistent across development and production environments. The expectations around code quality and versioning also differ widely in teams and create conflicts within teams, especially among team members with different roles and backgrounds. Practitioners commonly complain about low code quality in data science code, especially in notebooks.

3.1.3.4 Data Engineering

In developing machine learning models, data plays an important role. While we exclude challenges related exclusively to data-related work within ML pipelines, we report engineering challenges related to handling data within the system.

Data quality is considered important, but difficult for practitioners and not well supported by tools [322, 203, 189, 115, 297, 324, 197, 188, 8, 230, 14, 367, 167, 134, 280, 110, 225] **(17/50).** ML practitioners commonly report struggling with validating and improving data quality. Even with significant research efforts in building tools for data labeling, cleaning, visualization, and management, data work is still reported as a problematic area for practitioners. Practitioners reported that they need to invest significant effort and time in data pre-processing, cleaning, and assembly [110, 134, 230, 8, 280, 197, 189, 167]. Practitioners also mention their pain points in handling data errors and validating data quality, where better tool support is desired [320, 322, 324, 167, 297, 367, 188, 225]. Although it is common to associate these data issues within the model building pipeline, practitioners feel the need for cooperation from other parts of the organization (e.g., requirements engineers need to identify and specify requirements regarding data collection, formats, and the ranges of data and domain experts need to help to

understand the structure and semantics of the data), which they mention is lacking [297, 367, 280, 230, 134].

Pragmatic (🏋) knowledge boundary.

Internal data security and privacy policies restrict data access and use [203, 189, 115, 297, 197, 173, 230, 14, 167, 154] **(10/50).** Data access is often restricted due to security and privacy policies within organizations, beyond possible regulatory restrictions, e.g., policies ensuring that customer data is not shared outside the company. Due to restrictions on the flow of data, ML practitioners need to deal with additional complexities in the data pipeline, as only a restricted number of team members can analyze the data and as they have limited access to the right data and no access to data locally for model optimization or model debugging due to data movement constraints [154, 189, 197, 115].

Although training-serving skew is common, many teams lack support for its required detection and monitoring [115, 324, 197, 181, 400, 230, 14] **(7/50).** The mismatch between training data and production data is a common problem in products with ML components, where models work well on test data but generalize poorly to real-world data in production. Even if training the model with a representative dataset initially, the production environment often encounters drift toward data distributions that are less well supported by the model. Practitioners explain that monitoring models in production for staleness is an important activity that supports detecting the degradation of model performance and retraining it with new data if needed. However, they also find it challenging to set up the monitoring infrastructure and report a lack of tool support.

Data versioning and provenance tracking are often seen as elusive, with not enough tool support [203, 320, 144, 8, 173, 360, 134] **(7/50).** While software engineers routinely adopt mature version control systems for code, practitioners report challenges in versioning data, typically due to the large volumes of data involved. Practitioners mention that they need to have traceability and transparency to answer questions like *"Which data was this model trained on?"* or *"Which code or data change made our accuracy deteriorate?" [144]*, but it's not possible for them to keep track of data and models across the life cycle without technological support [144, 8, 402]. This is a bigger problem for practitioners in small companies as they do not want to invest in storage capacity to version their models and datasets, though they understand the importance [134].

3.1.3.5 Quality Assurance

One of the key ways that the incorporation of ML models changes how correctness is established in software systems is that system behavior is often evaluated in terms of statistical performance (e.g., accuracy or fit) rather than full conformance to a predefined specification. While software systems have long faced challenges in defining and verifying correctness, ML systems make these challenges more prominent because behavior is learned from data and may not be fully specified in advance. This complicates conventional processes and practices associated with testing and quality assurance.

Testing and debugging ML models is difficult due to lack of specifications [322, 360, 147, 280, 404, 229, 110, 188, 181, 376, 230, 189, 320, 115, 297, 324, 197, 8, 14] **(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 [189, 115, 197, 188, 376, 230, 14, 147, 404, 110]. 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 [181, 376, 360, 280, 229]. 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 [188, 376, 147]. 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 [188, 376, 110]. 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 [324, 376, 110]. Additionally, practitioners raise challenges regarding evaluating model robustness, on one hand, suffering from the lack of a concrete methodology [280, 110], and on the other hand, having various metrics but no consensus on which metric to use [188].

Also observed in our collaboration study (§3.2), as well as our LLM-evaluation study (§3.3 + §4.1), reflecting both syntactic (🔤) and semantic (💬) knowledge boundaries.

Testing of model interactions, pipelines, and the entire product is considered challenging and often neglected [197, 188, 173, 230, 280, 404, 229, 110] **(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 [188, 376, 404]. The lack of pipeline and system testing beyond the model is also considered a problematic area [173, 230, 280, 404, 229, 110]: 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 [404, 110], leading to system failures even where the model gives the correct results [229]. Practitioners also report having no systematic evaluation strategy or automated tools and techniques for pipeline and system-level testing [188, 230].

Semantic (💬) knowledge boundary.

Testing and monitoring models in production are considered important but difficult, and often not done [322, 324, 188, 230, 280] **(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 [324, 188]. Practitioners also find online testing very time-consuming, as it requires longer observation periods to determine meaningful results [322, 324]. Practitioners also pointed out that there is no surefire strategy to precisely detect when the model is underperforming in online testing [324].

Also came up in §3.1, and supported in intervention §4.1

There are no standard processes or guidelines on how to assess product qualitiessuch as fairness, security, and safety in practice [144, 172, 133, 360, 134, 295, 321, 33, 37] **(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 [144, 172], they often report having no effective methodology or concrete guidelines for evaluating them [172, 133, 360, 134, 295, 321, 37]. 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

This lack of guidelines and processes also came up in §3.1, and supported in interventions §4.1 and §4.2

proper process in place [133, 134]. Some practitioners report waiting for complaints from customers rather than being proactive when it comes to fairness [133] or even blindly expecting the algorithms to inherently provide qualities like security against attacks [172].

3.1.3.6 Process

Building software products with ML components involves many moving parts that need to be planned and integrated. Fitting all of these together in a cohesive process can be challenging.

Development of products with ML component(s) is often ad-hoc, lacking well-defined processes [189, 115, 339, 376, 173, 14, 360, 147, 167, 229, 16] **(11/50)**. Many practitioners report that they struggle with finding a good process for developing ML components and products around them [189, 115, 339, 376, 147], often coming up with ad-hoc strategies and experiencing a lack of good engineering practices [147, 229]. ML practitioners have explored using the traditional software development life cycles and found those to be a poor fit for exploratory development work. Even with a flexible agile methodology, practitioners identified that small iterations of sprints cannot fit the initial feasibility study that ML requires, with the timeline being too fixed and too short [189, 115, 16]. Also, they find it hard to set expectations for each sprint, as the project objectives may remain unclear at the beginning and need to be revisited after the initial investigation [173, 16].

The uncertainty in ML development makes it hard to plan and estimate effort and time [189, 376, 14, 360, 147, 110, 16] **(7/50)**. Machine learning work tends to be iterative and exploratory and as such uncertain, where practitioners cannot estimate upfront how long it may take to reach a model with a certain level of accuracy or whether that is even possible at all; instead, they commonly progress with many experiments with different algorithms and datasets [14, 147]. Practitioners, therefore, report having difficulties setting expectations and (intermediate) deadlines for a project [189, 376, 14, 110, 16] and providing any upfront estimates about effort and cost [189, 14, 360].

Semantic (💬) knowledge boundary.

Practitioners find documentation more important than ever in ML, but find it more challenging than traditional software documentation [189, 320, 115, 181, 230, 61, 171, 402, 267] **(9/50)**. Many practitioners point out various process

and coordination challenges rooted in poor documentation. Some practitioners emphasize that documentation is even more important when it comes to ML components, as human decisions are inscribed in different stages of ML pipelines and cannot be retrieved from code or data without documentation [115, 402]. The final model code is the outcome of many different explorations and experimentations that include multiple rounds of data processing, feature engineering, hyperparameter tuning, and other activities. Many problem-specific decisions have been made in those stages that cannot be understood from the resulting model or pipeline code. Some argue that not recording these decisions in documentation causes them to slowly become invisible, severely impacting future re-analysis and revisions, or even model integration and deployment [115, 181, 402]. Others emphasize that, along with model documentation, data documentation is also imperative to share hidden information inside the data and create a shared data understanding, yet mostly missing in organizations [320, 230]. Others report that, with the incorporation of ML, the documentation process becomes more complicated as ML practitioners find it difficult to present complex model information in an accessible way to all levels of stakeholders [61, 171, 267]. It is also non-trivial for practitioners to decide on the right amount of details to include in the documentation. They place the blame mostly on the lack of organizational incentives, resources, and unclear and vague guidelines for ML documentation [61, 267].

Semantic (💬) knowledge boundary.

Syntactice (🔤) knowledge boundary.

3.1.3.7 Organization and Teams

Along with the challenges faced in different development stages, practitioners also mention challenges they suffer from the organizational and teamwork perspective while building products with ML components.

Building products with ML components requires diverse skill sets, which is often missing in development teams [322, 203, 376, 400, 14, 360, 378, 154, 229, 235, 16, 9] **(12/50).** Incorporation of ML in a product does not merely mean adding just another component to the system; it requires people from multiple disciplines to get involved to support different aspects of this component. The team requires many diverse skill sets to develop, deploy, and integrate the model into the complete product, including hardware expertise, engineering skills, knowledge of math and statistics, business understanding, UX design ability, operations, and

domain expertise. The lack of this varied expertise in the team is commonly mentioned to be a challenge by practitioners [322, 376, 360, 154, 16, 9]. Also, as discussed in the next subsection that communication is often hindered by a lack of AI literacy or common terminology [115, 8, 14, 229, 235, 400], cross-disciplinary knowledge seems to be important for team members to interact and understand each other's vocabulary; however, practitioner experiences seem to indicate that such cross-disciplinary education is not broadly available yet [324, 110].

Syntactic (AB) and semantic (💬) knowledge boundaries.

Many teams are not well prepared for the extensive interdisciplinary collaboration and communication needed in ML products [203, 320, 400, 230, 14, 25, 229, 235, 51, 87, 376] **(11/50).** For building a product with ML components, team members need to collaborate with people from different disciplines as mentioned above, such as business leaders, engineers, designers, and various other departments inside the company, and even outside the organization [320, 376, 400, 235, 87]. Practitioners report that they often struggle to collaborate effectively in such interdisciplinary teams, because team members often do not understand the concerns of other members from other backgrounds, like data scientists lacking knowledge of engineering practices, testing frameworks, continuous integration and delivery, and such [115, 8, 14, 229]; software engineers lacking AI literacy [115, 229]; and data scientists and software engineers not understanding or interacting members with in with business roles [400, 235]. Practitioners report struggling with cultural differences, differences in expectations, and conflicting priorities [203, 14, 25, 235], and they often do not agree on assigned responsibilities [189, 230]. These multidisciplinary teams also suffer from miscommunications arising from inconsistency in their technical terminologies [230, 229, 51]. Siloing of teams by specialization and lack of communication across such silos are also observed in many production settings, fostering integration problems even further [230, 25].

Involves all levels of knowledge boundaries.

ML development can be costly and resource limits can substantially curb/limit efforts [203, 324, 14, 154, 235, 9] **(6/50)**. Practitioners report that organizations involved in the development of products with ML components often suffer from resource and budget limitations. Hardware, infrastructure, cloud storage, GPUs, etc., are expensive, and especially for small companies, it is difficult to justify such expenditures based on the expected return on investment from the model.

Lack of organizational incentives, resources, and education hampers achieving all system-level qualities [172, 282, 241, 154, 295, 235, 321, 37] **(8/50)**. Practitioners mention that organizational incentives also have an impact on achieving certain qualities of products with ML components. A quality that practitioners reported frequently as particularly challenging due to the lack of organizational incentives is fairness [282, 133]. Awareness of potential problems, including potential consequences from biased models, seems to be the main reason for lacking responsible AI practices, along with the lack of organizational incentives and structures, as well as priority conflicts. Safety, security, and privacy also seem to suffer from similar issues of awareness, education, resource constraints, and are often disregarded due to tradeoffs with development cost [172, 241, 154, 295, 37].

Pragmatic (🏃) knowledge boundary.

3.1.4 Discussion

With this meta-summary, we aggregate and summarize the challenges reported by industry practitioners who build software products with ML components. 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, to lacking tool support for data, model building, deployment, and monitoring.

3.1.4.1 Old, new, and harder challenges

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 soft-

ware 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.

3.1.4.2 Toward better engineering of ML products

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. We conclude this study by reflecting on possible directions.

- **Requirements Engineering.** For the challenges of unrealistic requirements, several studies mentioned that practitioners found it useful to conduct training sessions with clients and other team members on AI literacy, before starting the ML projects [230, 291, 318, 367]. But again, while many practitioners mention suffering from unclear model requirements, 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.
- **Architecture, Design, and Implementation.** Machine learning seems to provide significant challenges to architectural design of software systems, but arguably many challenges are similar to other large and complex and distributed software systems. While there are nascent discussions on organizing architecture knowledge as patterns [322, 377, 182, 175], it does not seem like the field has reached saturation. This seems to be a field though, where industry-oriented research (similar to the data architecture of facebook [124]) has more access to the complicated real-world scenarios where architectural planning becomes important than what academics can typically access. From the challenges raised by practitioners, it is apparent that along with the need for design practices, patterns, and mechanisms to handle system and

model-level considerations (e.g., dependency management, scalability, monitorability), we also need to support teams in shifting from model-centric work to system thinking, possibly through tailored education for ML practitioners.

- **Model Development and Data Engineering.** Consistent across many papers, we find that ML practitioners desire more engineering support, such as better infrastructure and tools for model and data work. Data scientists also indicate a need for more cooperation from other team members in terms of support for data, which necessitates better collaboration strategies and data education for the entire team. On the other hand, a few practitioners highlighted the necessity of standardization of ML code quality, which may be a low hanging fruit technically, but may require a change to the culture and practices in many projects.
- **Quality Assurance.** Quality assurance for machine learning, especially for models, is a very active area of research, with proposals for many different testing strategies to validate different model characteristics covered in multiple literature surveys [403, 293, 292, 137]. While we found that a lot of practitioners mentioned concerns about specifying model adequacy goals, few practitioners showed concerns about system testing, monitoring in production, and testing for fairness, security, and safety. We are surprised to not see more concerns about system-level quality beyond the model, which might indicate either that practitioners do not consider these testing areas as challenging, or that most organizations (especially outside of big tech) are not yet mature enough to even start thinking about such testing needs. Monitoring though is recognized as an important challenge, with many available tools but common adoption problems that may be worth investigating further.
- **Process.** While there is research on the development processes for ML models [348, 206], there seems to be little work on addressing process challenges that arise when integrating ML and non-ML work in production projects that are commonly mentioned by practitioners. We believe that this is an area with plenty of research opportunities to evaluate what processes and practices work well in different contexts.
- **Organization and Teams.** While there is lots of research on technical issues, practitioners often see organizational and team issues (such as a lack of AI literacy in teams, unclear responsibility boundaries, and a lack of

team synchronization) as some of the most difficult challenges to overcome. Education and better collaboration strategies seem to be the factors that might put a positive impact on mitigating many of the challenges that the practitioners mentioned.

Overall, we believe that substantial progress can be made through better education and broader adoption of established software engineering practices. At the same time, many of the challenges identified in this meta-summary point to deeper collaboration issues across ML development teams. For example, a lack of AI literacy often leads to unrealistic expectations from customers and managers during requirements engineering; the vagueness of ML problem specifications makes it difficult to align business goals with measurable performance metrics; and interdisciplinary teams frequently struggle to coordinate responsibilities when developing, deploying, and monitoring ML systems. These observations suggest that many of the difficulties in building ML products are not purely technical but arise from collaboration challenges among stakeholders with different expertise and perspectives. In the next chapter, I therefore examine these collaboration dynamics more closely through an interview study with industry practitioners.

3.2 Identification B: Interview Study of Collaboration Challenges in Building ML Products

The findings of this section have been published in ICSE'22*: "Collaboration Challenges in Building ML-enabled Systems: Communication, Documentation, Engineering, and Process,"* which was recognized with the 🏆 **Distinguished Paper Award** [230].

Following the meta-summary of practitioner challenges in the literature, we next investigate how these challenges emerge during collaboration in real ML product development. To better understand these collaboration dynamics in depth, 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. 3.3), interactions of project members with different technical backgrounds, and where conflicts arise between teams.

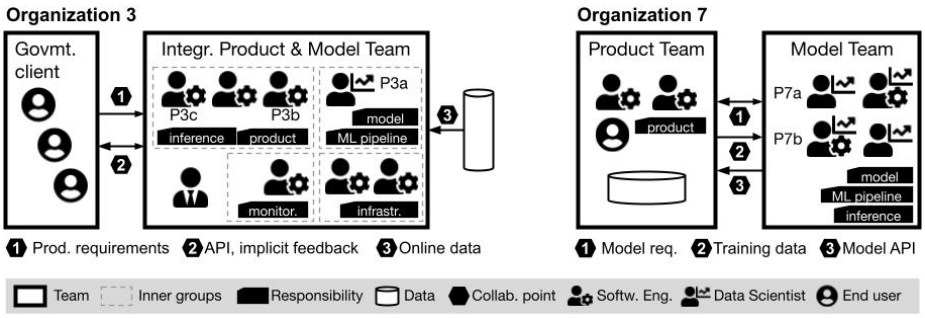


Fig. 3.3 Structure of Two Interviewed Organizations

While some organizations have adopted better collaboration practices than others, many struggle setting up structures, processes, and tooling for effective collaboration among team members with different backgrounds when developing ML-enabled systems. To the best of our knowledge, and confirmed by the practitioners we interviewed, there is little systematic or shared understanding of common collaboration challenges and best practices for developing ML-enabled systems and coordinating developers with very different backgrounds (e.g., data science vs. software engineering). We find that smaller and new to ML organizations struggle more, but have limited advice to draw from for improvement.

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. We found that organizational structure, team composition, power dynamics, and responsibilities differ between organizations, but we also found common *organizational patterns* at specific collaboration points and challenges associated with those.

Overall, our observations suggest four *themes* that would benefit from more attention when building ML-enabled systems. 👥: Invest in supporting *interdisciplinary teams* to work together (including education and avoiding silos). 📄: Pay

more attention to collaboration points and clearly *document* responsibilities and interfaces. ⚙: Consider *engineering work* a key contribution to the project. 🗓: Invest more into *process and planning*.

3.2.1 Related Work

Researchers and practitioners have discussed whether and how ML changes software engineering with the introduction of learned models as components in software systems, e.g., [141, 8, 249, 367, 282, 320, 305, 397]. To lay the foundation for our interview study and inform the questions we ask, we first provide an overview of the related work and existing theories on collaboration in traditional software engineering and discuss how ML may change this.

3.2.1.1 Collaboration in Software Engineering

Most software projects exceed the capacity of a single developer, requiring multiple developers and teams to collaborate ("work together") and coordinate ("align goals"). Collaboration happens *across* teams, often in a more formal and structured form, and *within* teams, where familiarity with other team members and frequent colocation fosters informal communication [219]. At a technical level, to allow multiple developers to work together, *abstraction* and a *divide and conquer* strategy are essential. Dividing software into *components* (modules, functions, subsystems) and hiding internals behind *interfaces* is a key principle of modular software development that allows teams to divide work, and work mostly independently until the final system is integrated [254, 211].

Teams within an organization tend to align with the technical structure of the system, with individuals or teams assigned to components [71], hence the technical structure (interfaces and dependencies between components) influences the points where teams collaborate and coordinate. Coordination challenges are especially observed when teams cannot easily and informally communicate, often studied in the context of distributed teams of global corporations [248, 128] and open source ecosystems [38, 337].

More broadly, *interdisciplinary* collaboration often poses challenges. It has been shown that when team members differ in their academic and professional backgrounds, it leads to communication, cultural or methodical challenges when

working together [48]. Key insights are that successful interdisciplinary collaboration depends on professional role, structural characteristics, personal characteristics, and a history of collaboration; specifically, structural factors such as unclear mission, insufficient time, excessive workload, and lack of administrative support are barriers to collaboration [53].

The component *interface* plays a key role in collaboration as a *negotiation and collaboration point*. It is where teams (re-)negotiate how to divide work and assign responsibilities [44]. Team members often seek information that may not be captured in interface descriptions, as interfaces are rarely fully specified [75]. In an idealized development process, interfaces are defined early based on what is assumed to remain stable [254], because changes to interfaces later are expensive and require the involvement of multiple teams. In addition, interfaces reflect *key architectural decisions* for the system, aimed to achieve desired overall qualities [23].

In practice though, the idealized divide-and-conquer approach following top-down planning does not always work without friction. Not all changes can be anticipated, leading to later modifications and renegotiations of interfaces [73, 38]. It may not be possible to identify how to decompose work and design stable interfaces until substantial experimentation has been performed [24]. To manage, negotiate, and communicate changes of interfaces, developers have adopted a wide range of strategies for communication [38, 341, 76], often relying on informal broadcast mechanisms to share planned or performed changes with other teams.

Software lifecycle models also address this tension of when and how to design stable interfaces. Traditional top-down models (e.g., waterfall [340]) plan software design after careful requirements analysis; the *spiral model* [340] pursues a risk-first approach in which developers iterate to prototype risky parts, which then informs future system design iterations; *agile* approaches [340] deemphasize upfront architectural design for fast iteration on incremental prototypes. The software architecture community has also grappled with the question of how much upfront architectural design is feasible, practical, or desirable [23, 379], showing a tension between the desire for upfront planning on one side and technical risks and unstable requirements on the other. *In this context, our research explores how introducing ML into software projects challenges collaboration.*

3.2.1.2 Software Engineering with ML Components

In a ML-enabled system, ML contributes one or multiple components to a larger system with traditional non-ML components. We refer to the whole system that an end user would use as the *product*. In some systems, the learned *model* may be a relatively small and isolated addition to a large traditional software system (e.g., audit prediction in tax software), in others it may provide the system's essential core with only minimal non-ML code around it (e.g., a sales prediction system sending daily predictions by email). In addition to models, an ML-enabled system typically also has components for training and monitoring the model(s) [141, 175]. Much attention in practice recently focuses on building robust ML *pipelines* for training and deploying models in a scalable fashion, often under names such as "*ML engineering," "SysML,"* and "*MLOps*" [175, 320, 203, 244]. In this work, we focus on the development of the entire ML-enabled system, including both ML and non-ML components.

Compared to traditional software systems, ML-enabled systems require additional expertise in *data science* to build the models and may place additional emphasis on expertise such as data management, safety, and ethics [167, 8]. In this paper, we primarily focus on the roles of *software engineers* and *data scientists*, who typically have different skills and educational backgrounds [161, 397, 307, 167]. Data science education tends to focus more on statistics, ML algorithms, and practical training of models from data (typically given a fixed dataset, not deploying the model, not building a system), whereas software engineering education focuses on engineering tradeoffs with competing qualities, limited information, limited budget, and the construction and deployment of systems. Research shows that software engineers engaging in data science without further education are often naive when building models [397] and that data scientists prefer to focus narrowly on modeling tasks [307] but are frequently faced with engineering work [376]. While there is plenty of work on supporting collaboration among software engineers [76, 59, 309, 406] and more recently on supporting collaboration among data scientists [370, 402], we are not aware of work exploring collaboration challenges between these roles, which we explore in this work.

The software engineering community has recently started to explore *software engineering for ML-enabled systems* as a research field, with many contributions

on bringing software-engineering techniques to ML tasks, such as testing models [15, 47] and ML algorithms [396, 64, 15, 47], deploying models [175, 115, 5, 27, 65], robustness and fairness of models [332, 358, 282], lifecycle for ML models [115, 260, 8, 206], and engineering challenges or best practices for developing ML components [147, 62, 134, 3, 8, 320, 205, 43]. A smaller body of work focuses on the ML-enabled system beyond the model, such as exploring system-level quality attributes [331], requirements engineering [367], architectural design [399], safety mechanisms [42, 305], and user interaction design [55, 6]. *In this paper, we adopt this system-wide scope and explore how data scientists and software engineers work together to build the system with* ML and non-ML components.

3.2.2 Research Design

Due to prior 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. 3.4: (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 [345, 344], 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 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 [345]. 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.

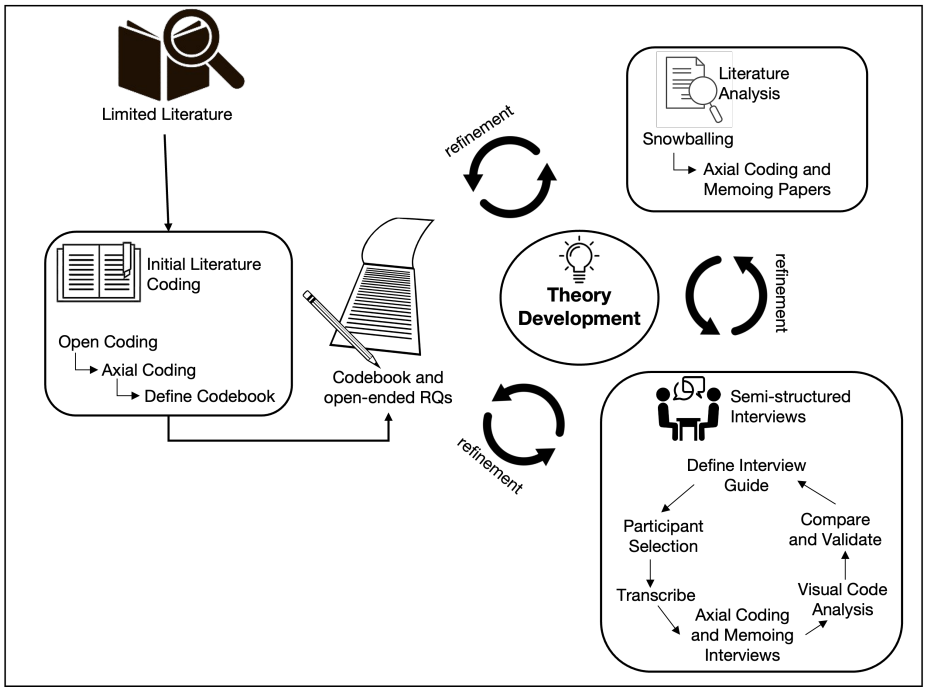


Fig. 3.4 Identification B: Research Design

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 [122]) 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 [134] (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. 3.3) 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.

3.2.3 Findings

3.2.3.1 Diversity of Org. Structures

Throughout our interviews, we found that the number and type of teams that participate in ML-enabled system development differs widely, as do their composition and responsibilities, power dynamics, and the formality of their collaborations. To illustrate these differences, we provide simplified descriptions of teams found in two organizations in Fig. 3.3. We show teams and their members, as well as the

artifacts for which they are *responsible*, such as, who develops the *model*, who builds a repeatable *pipeline*, who operates the model (*inference*), who is responsible for or owns the *data*, and who is responsible for the final *product*. A team often has multiple responsibilities and interfaces with other teams at multiple collaboration points. Where unambiguous, we refer to teams by their primary responsibility as *product team* or *model team*.

Organization 3 (Fig. 3.3, top) develops an ML-enabled system for a government client. The product (health domain), including an ML model and non-ML components, is developed by a single 8-person team. The team focuses on training a model first, before building a product around it. Software engineering and data science tasks are distributed within the team, where members cluster into groups with different responsibilities and roughly equal negotiation power. A single data scientist is part of this team, though they feel somewhat isolated. Data is sourced from public sources. The relationship between the client and development team is somewhat distant and formal. The product is offered as a service, but the team only receives feedback when things go wrong.

Organization 7 (Fig. 3.3, bottom) develops a product for in-house use (quality control for a production process). A small team is developing and using the product, but model development is delegated to an external team (different company) composed of four data scientists, of which two have some software engineering background. The product team interacts with the model team to define and revise model requirements based on product requirements. The product team also provides confidential proprietary data for training. The model team deploys the model and provides a ready-to-use API to the product team. As the relationship between the teams crosses company boundaries, it is rather distant and formal. The product team clearly has the power in negotiations between the teams.

As illustrated by the differences between these two examples, we found no clear global patterns when looking across organizations, but patterns did emerge when focusing on three specific collaboration aspects, as we will discuss in the next sections.

3.2.3.2 Collaboration Point: Requirements and Planning

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 requirments).

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.

Product and Model Requirements We found a constant tension between product and model requirements in our interviews. Functional and nonfunctional product requirements are for the entire product. Model requirements set goals and constraints for the model team, such as expected accuracy and latency, target domain, and available data.

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.

Represents both 🔤 syntactic and 💬 semantic boundaries

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 [244].

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 [305, 397, 270].

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 [167, 278, 197, 397, 368, 330]. Ashmore et al. report mapping product requirements to model requirements as an open challenge [15].

Suspect due to 💬 semantic boundaries

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 [33, 271, 197, 140, 198, 142].

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., [402, 133, 201, 33, 330, 318, 387, 55, 6, 134].

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

Represents both 💬 semantic and 🫂 pragmatic boundaries

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 [367].

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 [147, 318, 367].

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 [220] could be extended to cover more of these model requirements.

Project Planning

ML uncertainty makes effort estimation difficult (👥). Irrespective of trajectory, 19 participants (P3a, P4a, P7a-b, P8a, P14b, P15b-c, P16a, P17a, P18a, P19a-b, P20a, P22a-c, P23a, P25a) mentioned that the uncertainties associated with ML components make it difficult to estimate the timeline for development of ML components and by extension the product. Model development is typically seen as a science-like activity, where iterative experimentation and exploration is needed to identify whether and how a problem can be solved, rather than as an engineering activity that follows a somewhat predictable process. This science-like nature makes it difficult for the model team to set expectations or contracts with clients or the product team regarding effort, cost, or accuracy. While data scientists find effort estimation difficult, lack of ML literacy in managers makes it worse (P15b, P16a, P19b, P20a, P22b). Teams report deploying subpar models when running out of time (P3a, P15b, P19a), or postponing or even canceling deployment (P25a). These findings align with literature mentioning difficulties associated with effort estimation for ML tasks due to uncertainty [14, 206, 376] and planning projects in a structured manner with diverse methodologies, with diverse trajectories, and without practical guidance [42, 206, 376].

Semantic boundary

Generally, participants frequently report that synchronization between teams is challenging because of different team pace, different development processes, and tangled responsibilities (P2a, P11a, P12a, P14a, P14b, P15b, P15c, P19a).

Recommendations. Participants suggested several mitigation strategies: keeping extra buffer times and adding additional time-boxes for R&D in initial phases (P8a, P19a, P22b, P22c, P23a; 🗓), continuously involving clients in every phase so that they can understand the progression of the project and be aware of potential missed deadlines (P6b, P7a, P22a, P23a; 👥). From the interviews, we also observe the benefits of managers who understand both software engineering and machine learning and can align product and model teams toward common goals (P2a, P6a, P8a, P28a; 👥).

3.2.3.3 Collaboration Point: Training Data

Data is essential for machine learning, but disagreements and frustrations around training data were the most common collaboration challenges mentioned in our interviews. In most organizations, the team that is responsible for building the model is not the team that collects, owns, and understands the data, making data a key collaboration point between teams in ML-enabled systems development.

Common Organizational Structures We observed three patterns around data that influence collaboration challenges from the perspective of the model team:

Provided data: The product team has the responsibility of providing data to the model team (org. 6–8, 13, 18, 21, 23). The product team is the initial point of contact for all data-related questions from the model team. The product team may own the data or acquire it from a separate data team (internal or external). Coordination regarding data tends to be distant and formal, and the product team tends to hold more negotiation power.

External data: The product team does not have direct responsibility for providing data, but instead, the model team relies on external data providers. Commonly, the model team (i) uses publicly available resources (e.g., academic datasets, org. 2–4, 6, 19) or (ii) hires a third party for collecting or labeling data (org. 9, 15–17, 22, 23). In the former case, the model team has little to no negotiation power over data, whereas, in the latter, it can set expectations.

In-house data: Product, model, and data teams are all part of the same organization and the model team relies on internal data from that organization (org. 1, 5, 9–12, 14, 20, 24–28). In these cases, both product and model teams often find it challenging to negotiate access to internal data due to differing priorities, internal politics, permissions, and security constraints.

Negotiating Data Quality and Quantity Disagreements and frustrations around training data were the most common collaboration challenges in our interviews. In almost every project, data scientists were unsatisfied with the quality and quantity of data they received at this collaboration point, in line with a recent survey showing data availability and management to be the top-ranked challenge in building ML-enabled systems [8].

Provided and public data is often inadequate (📄, 👥). In organizations where data is provided by the product team (P7a, P8a, P13a, P22a, P22c), the model team commonly states that it is difficult to get sufficient data. The data that they receive is often of low quality, requiring significant investment in data cleaning. Similar to the requirements challenges discussed earlier, they often state that the product team has little knowledge or intuition for the amount and quality of data needed. For example, participant P13a stated that they were given a spreadsheet with only 50 rows to build a model and P7a reported having to spend a lot of time convincing the product team of the importance of data quality. This aligns with past observations that software engineers often have little appreciation for data quality concerns [187, 167, 270] and that training data is often insufficient and incomplete [305, 7, 270, 142, 376, 198, 331].

Semantic boundary

Pragmatic boundary

When the model team uses public data sources, its members also have little influence over data quality and quantity and report significant effort for cleaning low quality and noisy data (P2a, P3a, P4a, P3c, P6b, P19b, P23a). Papers have similarly questioned the representativeness and trustworthiness of public training data [387, 367, 115] as *"nobody gets paid to maintain such data"* [104].

Pragmatic boundary

Training-serving skew is a common challenge when training data is provided to the model team: models show promising results, but do not generalize to production data because it differs from training data (P4a, P8a, P13a, P15a, P15c, P21a, P23a, P22c) [14, 50, 348, 271, 197, 387, 270, 272, 198, 408]. Our interviews show that

this skew often originates from inadequate training data combined with unclear information about production data, and therefore no chance to evaluate whether the training data is representative of production data.

Semantic boundary

Data understanding and access to domain experts is a bottleneck (📄, 📅). Existing data documentation (e.g, data item definitions, semantics, structure, schema) is almost never sufficient for model teams to understand the data (also mentioned in [159]). In the absence of clear documentation, team members often collect information and keep track of unwritten details in their heads (P5a), known as institutional or tribal knowledge [134, 8]. Data understanding and debugging often involve members from different teams and thus cause challenges at this collaboration point.

Represents both semantic and pragmatic boundaries

Model teams receiving their data from the product team report struggling with data understanding and having a difficult time receiving help from the product team (or the data team that the product team works with) (P8a, P7b, P13a). As the model team does not have direct communication with the data team, the data understanding issues often cannot be resolved effectively. For example, P13a reports *"Ideally for us it would be so good to spend maybe a week or two with one person continuously trying to understand the data. It's one of the biggest problems actually, because even if you have the person, if you're not in contact all the time, then you misinterpreted some things and you build on it."* The low negotiation power of the model team hinders access to domain experts.

Model teams using public data similarly struggle with data understanding and receiving help (P3a, P4a, P19a), relying on sparse data documentation or trying to reach any experts on the data.

For in-house projects, in several organizations the model team relies on data in shared databases (org 5,11, 26, 27, 28), collected by instrumenting a production system, but shared by multiple teams. Several teams shared problems with evolving and often poorly documented data sources, as participant P5a illustrates *"[data rows] can have 4,000 features, 10,000 features. And no one really cares. They just dump features there. [...] I just cannot track 10,000 features."* Model teams face challenges in understanding data and identifying a team that can help (P5a, P25a, P20b, P27a), a problem also reported in a prior study on data scientists at Microsoft [167].

Challenges in understanding data and needing domain experts are also frequently mentioned in the literature [167, 270, 159, 27, 140, 134], as is the danger of building models with insufficient understanding of the data [367, 115]. Although we are not aware of literature discussing the challenges of accessing domain experts, papers have shown that even when data scientists have access, effective knowledge transfer is challenging [323, 253].

Ambiguity when hiring a data team (📄). When the model team hires an external data team for collecting or labelling data (org 9, 15, 16, 17, 22, 23), the model team has much more negotiation power over setting data quality and quantity expectations (though Kim et al. report that model teams may have difficulty getting buy-in from the product team for hiring a data team in the first place [167]).. Our interviews did not surface the same frustrations as with provided data and public data, but instead participants from these organizations reported *communication vagueness* and *hidden assumptions* as key challenges at this collaboration point (P9a, P15a, P15c, P16a, P17b, P22a, P23a, P22c). For example, P9a related how different labelling companies given the same specification widely disagreed on labels, when the specification was not clear enough.

Represents both 💬 semantic and 🤝 pragmatic boundaries

We found that expectations between model and data team are often communicated verbally without clear documentation. The data team often does not have sufficient context to understand what data is needed. For example, participant P17b states *"Data collectors can't understand the data requirements all the time. Because, when a questionnaire [for data collection] is designed, the overview of the project is not always described to them. Even if we describe it, they can't always catch it."* Reports about low quality data from hired data teams have been also discussed in the literature [367, 15, 197, 376, 142].

Need to handle evolving data (⚙, 👥). In most projects, models need to be regularly retrained with more data or adapted to changes in the environment (e.g., data drift) [141, 197], which is a challenge for many model teams (P3a, P3c, P5a, P7a, P11a, P7b, P15c, P18a, P19b, P22a). When product teams provide the data, they often have a static view and provide only a single snapshot of data rather than preparing for updates,. where model teams with their limited negotiation power have a difficult time fostering a more dynamic mindset (P7a, P7b, P15c,

Represents both 💬 semantic and 🤝 pragmatic boundaries

P18a, P22a), as expressed by participant P15c: *"People don't understand that for a machine learning project, data has to be provided constantly."* It can be challenging for a model team to convince the product team to invest in continuous model maintenance and evolution (P7a, P15c) [159].

Conversely, if data is provided continuously (most commonly with public data sources, in-house sources, and own data teams), model teams struggle with ensuring consistency over time. Data sources can suddenly change without announcement (e.g., changes to schema, distributions, semantics), surprising model teams that make but do not check assumptions about the data (P3a, P3c, P19b). For example, participants P5a and P11a report similar challenges with in-house data, where their low negotiation power does not allow them to set quality expectations, and face undesired and unannounced changes in data sources made by other teams. Most organizations do not have a monitoring infrastructure to detect changes in data quality or quantity.

In-house priorities and security concerns often obstruct data access (🗖). In in-house projects, we frequently heard about the product or model team struggling to work with another team within the same organization that owns the data.. Often, these in-house projects are local initiatives (e.g., logistics optimization) with more or less buy-in from management and without buy-in from other teams that have their own priorities; sometimes other teams explicitly question the business value of the product. The interviewed model teams usually have little negotiation power to request data (especially if it involves collecting additional data) and almost never a specific agreement to continuously receive data in a certain format, quality, or quantity (P5a, P10a, P11a, P20a, P20b, P27a) (also observed in studies at Microsoft and ING [167, 115]). For example, P10a shared *"we wanted to ask the data warehouse team to [provide data], and it was really hard to get resources. They wouldn't do that because it was hard to measure the impact [our in-house project] had on the bottom line of the business."* Model teams in these settings tend to work with whatever data they can get eventually.

Critical 🤝 pragmatic boundary

Security and privacy concerns can also limit access to data (P7a, P7b, P21a, P21b, P22a, P24a) [270, 159, 197, 198], especially when data is owned by a team in a different organization, causing frustration, lengthy negotiations, and sometimes

expensive data-handling restrictions (e.g., no use of cloud resources) for model teams.

Recommendations. Data quality and quantity is important to model teams, yet they often find themselves in a position of low negotiation power, leading to frustration and collaboration inefficiencies. Model teams that have the freedom to set expectations and hire their own data teams are noticeably more satisfied (P16b). When planning the entire product , it seems important to pay special attention to this collaboration point, and budget for data collection, access to domain experts, or even a dedicated data team (🗓). Explicitly planning to provide substantial access to domain experts early in the project was suggested as important (P25a).

We found it surprising that despite the importance of this collaboration point there is little written agreement on expectations and often limited documentation (🗎), even when hiring a dedicated data team – in stark contrast to more established contracts for traditional software components. Not all organizations allow the more agile, constant close collaboration between model and data teams that some suggest [270, 272]. With a more formal or distant relationship (e.g., across organizations, teams without buy-in), it seems beneficial to adopt a *more formal contract*, specifying data quantity and quality expectations, which are well researched in the database literature [202] and have been repeatedly discussed in the context of ML-enabled systems [323, 167, 159, 142, 198]. This has also been framed as *data requirements* in the software engineering literature [367, 348, 305]. When working with a dedicated data team, participants suggested to invest in making expectations very clear, for example, by providing precise specifications and guidelines (P9a, P6b, P28a), running training sessions for the data collectors and annotators (P17b, P22c), and measuring inter-rater agreement (P6b).

In addition, automated checks are important as data evolves (⚙). For example, participant P13a mentioned proactively setting up data monitoring to detect problems (e.g., schema violations, distribution shifts) at this collaboration point; a practice suggested also in the literature [348, 318, 270, 272, 198] and supported by recent tooling, e.g., [310, 272, 160]. The risks regarding possible unnoticed changes to data make it important to consider *data validation* and *monitoring infrastructure* as a key feature of the product early on (⚙, 🗓), as also emphasized by several participants (P5a, P25a, P26a, P28a).

3.2.3.4 Collaboration Point: Product-Model Integration

As discussed earlier, to build an ML-enabled system both ML components and traditional non-ML components need to be integrated and deployed, requiring data scientists and software engineers, typically across multiple teams, to work together. We found many conflicts at this collaboration point, stemming from unclear processes and responsibilities, as well as differing practices and expectations.

Common Organizational Structures We saw large differences among organizations in how engineering responsibilities were assigned, which is most visible in how responsibility for *model deployment and operation* is assigned, which typically involves significant engineering effort for building reproducible pipelines, API design, or cloud deployment, often with MLOps technologies. We found the following patterns:

Shared model code: In some organizations (2, 6, 23, 25), the model team is responsible only for model development and delivers training code (e.g., in a notebook) or model files to the product team; the product team takes responsibility for deployment and operation of the model, possibly rewriting the training code as a pipeline. Here, the model team has little or no engineering responsibilities.

Model as API: In most organizations (18 out of 28), the model team is responsible for developing and deploying the model. Hence, the model team requires substantial engineering skills in addition to data science expertise. Yet, organizations had different team compositions: some model teams are mostly composed of data scientists with little engineering capabilities (org. 7, 13, 17, 22, 26), some model teams consist mostly of software engineers who have picked up some data science knowledge (org. 4, 15, 16, 18, 19, 21, 24), and others have mixed team members (org. 1, 9, 11, 12, 14, 28). These model teams typically provide an API to the product team, or release individual model predictions (e.g., shared files, email; org. 17, 19, 22) or install models directly on servers (org. 4, 9, 12).

All-in-one: If only few people work on model *and* product, sometimes a single team (or even a single person) shares all responsibilities (org. 3, 5, 10, 20, 27). Team compositions are again different across organizations, where it can be a small team with only data scientists (org. 10, 20, 27; two occasionally solicit temporary

help from other software engineering teams within the organization) or mixed teams with data scientists and software engineers (org. 3, 5).

We also observed two outliers, making different deliberate decisions about engineering responsibilities: One startup (org. 8) had a distinct model deployment team, allowing the model team to focus on data science without much engineering responsibility. In one large organization (org. 28), an engineering-focused model team (model as API) was supported by a dedicated *research team* focused on data-science research with fewer engineering responsibilities.

Responsibility and Culture Clashes Interdisciplinary collaboration is challenging. We observed many conflicts between data science and software engineering culture, made worse by unclear responsibilities and boundaries.

Team responsibilities often do not match capabilities and preferences (⚙). When the model team has responsibilities requiring substantial engineering work (*model as API*, *all-in-one*),. we observed some dissatisfaction when its members were assigned undesired responsibilities. Data scientists preferred technical support rather than needing to do everything themselves (P7a, P7b, 13a), but can find it hard to convince management to hire engineers (P10a, P20a, P20b). For example P10a describes *"I was always struggling to change the mindset of the team lead, convincing him to hire an engineer, because I had to learn a lot by myself. For me, it was fine to learn. I just didn't want this to be my main responsibility."* Especially in small teams, data scientists report struggling with the complexity of the typical ML infrastructure (P7b, P9a, P14a, P26a, P28a).

Represents all levels of knowledge

In contrast, when deployment is the responsibility of software engineers in the product team (*shared model code*) or of dedicated engineers in *all-in-one* teams, some of those engineers report problems integrating the models due to insufficient knowledge on model context or domain, and the model code not being packaged well for deployment (P20b, P23a, P27a). In several organizations, we heard about software engineers performing ML tasks without having enough ML understanding (P5a, P15b, P15c, P16b, 18b, 19b, 20b). Mirroring observations from past research [397], P5a reports *"there are people who are ML engineers at [company] , but they don't really understand ML. They were actually software engineers. The way they*

work on ML is just by using the existing infrastructure. So they don't understand [overfitting, underfitting, ...]. They just copy-paste code."

Siloing data scientists fosters integration problems (👥, 📅). We observed data scientists often working in isolation—known as *siloing*—in all types of organizational structures, even within single small teams and within engineering-focused teams. In such settings, data scientists often work in isolation with weak requirements without understanding the larger context and seriously engaging with others only during integration (P3a, P3c, P6a, P7b, P11a, P13a, P15b, P25a) [140], where problems may surface. For example, participant P11a reported a problem where product and model teams had different assumptions about the expected inputs and the issue could only be identified after a lot of back and forth between teams at a late stage in the project.

Semantic boundary

Technical jargon challenges communication (👥). Participants frequently described communication issues arising from differing terminology used by members from different backgrounds (P1a, P1b, P2a, P3a, P5b, P8a, P12a, P14a, P14b, P16a, P17a, P17b, P18a, P18b, P20a, P22b, P23a), leading to ambiguity, misunderstandings, and inconsistent assumptions (on top of communication challenges with domain experts [159, 266, 368]). Participant P1b reports *"There are a lot of conversations in which disambiguation becomes necessary. We often use different kinds of words that might be ambiguous."* For example, data scientists may refer to prediction accuracy as *performance*, a term many software engineers associate with response time. These challenges can be observed more frequently between teams, but they even occur within a single all-in-one team with members from different backgrounds (P3a, P3b, P3c, P20a).

Syntactic boundary

Code quality, documentation, and versioning expectations differ widely and cause conflicts (👥, ⚙). Many participants reported conflicts around development practices between data scientists and software engineers that arise during integration and deployment. Participants report poor practices that may also be observed in many traditional software projects; but particularly software engineers commonly expressed frustration in our interviews that data scientists do not follow the same development practices or have the same quality standards when it comes

Represents all levels of knowledge boundaries

to writing code. Reported problems relate to poor code quality (P1b, P2a, P3b, P5a, P6a, P6b, P10a, P11a, P14a, P15b, P15c, P17a, P18a, P19a, P20a, P20b, P26a) [265, 125, 315, 14, 376, 115, 62], insufficient documentation (P5a, P5b, P6a, P6b, P10a, P15c, P26a) [159, 402, 220], and not extending version control to data and models (P3c, P7a, P10a, P14a, P20b). In two shared-model-code organizations, our participants report having to rewrite code from the data scientists (P2a, P6a, P6b). Missing documentation for ML code and models is considered the cause for different assumptions that lead to incompatibility between ML and non-ML components (P10a) and for losing knowledge and even the model when faced with turnover (P6a, P6b). Recent papers similarly hold poor documentation responsible for team decisions becoming invisible and inadvertently causing hidden assumptions [142, 115, 266, 134, 402, 159]. Hopkins and Booth called model and data versioning in small companies as desired but "elusive" [134].

Recommendations. Many conflicts relate to boundaries of responsibility (especially for engineering responsibilities) and to different expectations by team members with different backgrounds. Better teams tend to define processes, responsibilities, and boundaries more carefully (), document APIs at collaboration points between teams (), and recruit dedicated engineering support for model deployment (), but also establish a team culture with mutual understanding and exchange (). Big tech companies usually have more established processes, with more teams with clear responsibilities, than smaller organizations and startups that often follow ad-hoc processes or figure out responsibilities as they go.

The need for engineering skills for ML projects has frequently been discussed [8, 244, 339, 315, 320, 408], but our interviewees differ widely in whether all data scientists should have substantial engineering responsibilities or whether engineers should support data scientists so that they can focus on their core expertise (). Especially interviewees from big tech emphasized that they expect engineering skills from all data science hires (P28a). Others emphasized that recruiting software engineers and operations staff with basic data science knowledge can help at many communication and integration tasks, such as converting experimental ML code for deployment (P2a, P3b), fostering communication (P3c, P25a), and monitoring models in production (P5b). Generally, *siloing data scientists is widely recognized as problematic* and many interviewees suggest practices for improving

communication (👥), such as training sessions for establishing common terminology (P11a, P17a, P22a, P22c, P23a), weekly all-hands meetings to present all tasks and synchronize (P2a, P3c, P11a, P6b), and proactive communication to broadcast upcoming changes in data or infrastructure (P11a, P14a, P14b). This mirrors suggestions to invest in interdisciplinary training [167, 8, 266, 249, 161] and proactive communication [187].

Quality Assurance for Model and Product During development and integration, questions of responsibility for quality assurance frequently arise, often requiring coordination and collaboration between multiple teams. This includes evaluating components individually (including the model) as well as their integration and the whole system, often including evaluating and monitoring the system *online* (in production).

Model adequacy goals are difficult to establish (📄, 👥). Offline accuracy evaluation of models is almost always performed by the model team who is responsible for building the model, though often have difficulty deciding locally when the model is good enough (P1a, P3a, P5a, P6a, P7a, P15b, P16b, P23a) [147, 115]. As discussed earlier, model team members often receive little guidance on model adequacy criteria and are unsure about the actual distribution of production data. They also voice concerns about establishing ground truth, for example, needing to support data for different clients, and hence not being able to establish (offline) measures for model quality (P1b, P16b, P18a, P28a). As quality requirements beyond accuracy are rarely provided for models, model teams usually do not feel responsible for testing latency, memory consumption, or fairness (P2a, P3c, P4a, P5a, P6b, P7a, P14a, P15b, P20b). Whereas literature discussed challenges in measuring business impact of a model [167, 15, 30, 142], interviewed data scientists were concerned about this only with regards to convincing clients, managers or product teams to provide resources (P7a, P7b, P10a, P26a, P27a).

Syntactic and 💬 semantic boundaries

🤝 Pragmatic boundary

Limited confidence without transparent model evaluation (📄). Participants in several organizations report that model teams do not prioritize model evaluation and have no systematic evaluation strategy (especially if they do not have established adequacy criteria they try to meet), performing occasional "ad-hoc inspections"

💬 Semantic and 🤝 pragmatic boundaries

instead (P2a, P15b, P16b, P18b, P19b, P20b, P21b, P22a, P22b). Without transparency about their test processes and test results, other teams voiced reduced confidence in the model, leading to skepticism to adopt the model (P7a, P10a, P21b, P22a).

We aim to support this evaluation challenge with intervention A, §4.1

Unclear responsibilities for system testing (📅). Teams often struggle with testing the entire product, integrating ML and non-ML components. Model teams frequently explicitly mentioned that they assume no responsibility for product quality (including integration testing and testing in production) and have not been involved in planning for system testing, but that their responsibilities end with delivering a model evaluated for accuracy (P3a, P14a, P15b, P25a, P26a). However, in several organizations, product teams also did not plan for testing the entire system with the model(s) and, at most, conducted system testing in an ad-hoc way (P2a, P6a, P16a, P18a, P22a). Recent literature has reported a similar lack of focus on system testing in product teams [402, 27], mirroring also a focus in academic research on testing models rather than testing the entire system [15, 47]. Interestingly, some established software development organizations delegated testing to an existing separate quality assurance team, which however had no process or experience for testing ML products (P2a, P8a, P16a, P18b, P19a).

All levels of knowledge boundaries exist

Planning for online testing and monitoring rare (📅, ⚙, 👥). Due to possible training-serving skew and data drift, literature emphasizes the need for online evaluation . With collected telemetry, one can usually approximate both product and model quality, monitor updates, and experiment in production [30]. Online testing usually requires coordination among multiple teams responsible for product, model, and operation . We observed that most organizations do *not* perform monitoring or online testing, as it is considered difficult and participants refer to a lack of standard process, automation, or even test awareness (P2a, P3a, P3b, P4a, P6b, P7a, P10a, P15b, P16b, P18b, P19b, 25a, P27a). Only 11 out of 28 organizations collected any telemetry; it is most established in big tech organizations. When to retrain models is often decided based on intuition or manual inspection, though many aspire to more automation (P1a, P3a, P3c, P5a, P10a, P22a, P25a, P27a). Responsibilities around online evaluation are often neither planned nor assigned upfront as part of the project.

This space often suffers from 🔤 syntactic and 💬 semantic knowledge boundary, Cf. §2.2.1

We also target this challenge in §4.1, intervention A

Most model teams are aware of the possibility of data drift, but many do not have any monitoring infrastructure for detecting and managing drift in production. If telemetry is collected, it is the responsibility of the product or operations team and it is not always accessible to the model team. Four participants report that they rely on manual feedback about problems from the product team (P1a, P3a, P4a, P10a). At the same time, others report that product and operation teams do not necessarily have sufficient data science knowledge to provide meaningful feedback (P3a, P3b, P5b, P18b, P22a) [284].

Recommendations. Quality assurance involves multiple teams and benefits from explicit planning and making it a high priority (🗓). While the product team should likely take responsibility for product quality and system testing, such testing often involves building monitoring and experimentation infrastructure (⚙), which requires planning and coordination with teams responsible for model development, deployment, and operation (if separate) to identify the right measures. Model teams benefit from receiving feedback on their model from production systems, but such support needs to be planned explicitly, with corresponding engineering effort assigned and budgeted, even in organizations following a model-first trajectory. We suspect that education about benefits of testing in production and common infrastructure (often under the label DevOps /MLOps [203]) can increase buy-in from all involved teams (👥). Organizations that have established monitoring and experimentation infrastructure strongly endorse it (P5a, P25a, P26a, P28a).

Defining clear quality requirements for model and product can help all teams to focus their quality assurance activities (📄). Even when it is challenging to define adequacy criteria upfront, teams can together develop a quality assurance plan for model and product. Participants and literature emphasized the importance of human feedback to evaluate model predictions (P11a, P14a) [314], which requires planning to collect such feedback (🗓). System and usability testing may similarly require planning for user studies with prototypes and shadow deployment [387, 318, 348].

3.2.4 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 [48], 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 [359] (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 [220] 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 [2] 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 [161, 315, 249] 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 [49], more research into integrated process life cycles for ML products (covering software engineering and data science steps) in all their different forms is needed.

3.3 Summary and Reflection

In this chapter, we explored the challenges encountered in building ML products. The contributions of this challenge-identification thrust include:

- A comprehensive catalog of challenges in ML product development, derived from an extensive meta-summary of academic literature. This study highlights underlying implicit issues in collaboration, and complements the findings of our subsequent interview study.
- An empirically grounded set of collaboration challenges, identified through a qualitative interview study with 45 practitioners across 28 organizations, providing insights into real-world collaboration issues encountered when developing ML products.

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.

Note: As large language models (LLMs) have rapidly become central to many ML-enabled products, it was important to also examine the challenges practitioners face when working with these systems. Our analysis revealed that many of these challenges center around how LLM outputs are evaluated in practice. We therefore examine these challenges in the following chapter (§4.1) in conjunction with practitioner evaluation practices for assessing LLM outputs in real-world systems, where these practices serve as interventions to address these challenges.

3.3.1 What Has Changed Since These Studies?

Since these studies were conducted, several systematic literature reviews and survey papers have examined different aspects of engineering ML-enabled software systems [417, 418, 424, 434], including requirements engineering for AI systems [419–421], data engineering and lifecycle management [423, 422], development and operational practices such as MLOps and continuous ML practices [425–427, 429], and several system qualities such as robustness, security, fairness, maintainability, scalability, and environmental cost [430–433]. Alongside these reviews, a growing body of empirical studies–including interviews, surveys, and case studies–has examined concrete engineering practices in areas such as requirements specification [437, 438], software architecture [439, 441], model integration [440], deployment, and monitoring of ML-enabled systems [435, 436]. Across these studies, a consistent theme emerges: Many of the challenges identified in the earlier work presented in this dissertation remain persistent, such as difficulties specifying requirements for probabilistic ML systems, managing evolving datasets and pipelines, integrating models into complex software architectures, and monitoring model behavior reliably in production. While the technical capabilities of machine learning systems have advanced rapidly—with recent advances such as agentic AI systems, and AI coding assistants—the engineering practices, organizational structures, and development processes required to support them have evolved more slowly and struggle to keep pace. In this sense, the challenges identified in this dissertation are not isolated observations but part of a broader pattern now widely recognized in the literature.

At the same time, the technological landscape has evolved, shifting from deploying individual task-specific ML models within products toward a broader ecosystem that includes foundation models, LLM-based applications, and increasingly agentic development workflows. Within this evolving landscape, evaluation, observability, and operational control are emerging as central engineering concerns for modern AI systems [442, 446, 445]. In our meta-summary and collaboration studies in this chapter, evaluation already appeared as a difficult collaboarion point for ML teams. Recent work suggests that this issue has become even more pronounced with the rise of LLM-based applications [442, 444, 445]. Empirical studies of LLM application development and research on LLM observability report that practitioners struggle to design reliable evaluation strategies, identify meaningful telemetry signals, and

diagnose unexpected system behavior in deployment [233, 442, 446]. Practitioner studies of production LLM systems report that teams often focus heavily on architectural patterns such as retrieval-augmented generation (RAG), yet continue to struggle with evaluating model capabilities, monitoring system behavior, and managing risks in deployment [418]. Similarly, research on foundation-model engineering emphasizes that moving from research prototypes to production systems requires addressing challenges not only in prompt design and model alignment, but also in testing, observability, feedback integration, orchestration, and compliance [442]. Emerging work on LLMOps further argues that traditional MLOps pipelines are insufficient for LLM-based systems, as developers must manage nondeterministic outputs, open-ended behaviors, and complex runtime interactions [443]. Taken together, this literature suggests that the engineering focus in AI systems is increasingly expanding from building models toward evaluating, observing, and governing model behavior in complex applications—a transition that our LLM evaluation study in Chapter 4.1 captures at an early stage.

Beyond changes in ML systems themselves, such as the incorporation of foundation models, and agentic AI systems, recent work also highlights transformations in the development workflows used to build these systems. A particularly notable development is the emergence of AI agents—such as coding assistants and autonomous development tools—that increasingly participate in software engineering activities such as writing code, generating tests, and assisting with design decisions. Rather than focusing solely on collaboration among software engineers, data scientists, and other stakeholders, recent research increasingly examines how humans collaborate with AI coding assistants, software engineering agents, and multi-agent development systems. Empirical studies of developer–agent collaboration show that developers are generally more successful when interacting with agents through iterative collaboration and feedback loops, rather than delegating tasks in a one-shot manner [494–496]; however, these studies also highlight persistent challenges related to trust, debugging, verification, and testing of agent-generated code [447]. In parallel, early investigations of agentic development pipelines report that organizations are beginning to rethink workflows, roles, and tooling ecosystems as agents participate in tasks such as code generation, system integration, and testing, while also identifying emerging challenges around orchestration, governance, and sustainable adoption of these systems [448, 450]. More broadly, recent work on

GenAI-augmented software engineering frames this shift as a transformation of software engineering processes rather than merely a tooling upgrade, raising new questions about coordination, accountability, and quality assurance in AI-supported development [449]. From the perspective of this dissertation, these developments extend the collaboration challenges identified earlier: the collaboration problem in ML-enabled systems is **no longer limited to human–human collaboration across knowledge boundaries**, but increasingly involves **human–AI and AI–AI coordination** within complex development ecosystems.

3.3.2 Connecting Identified Challenges to Interventions

While many of the identified challenges require further research attention, we begin by designing three interventions to address three specific collaboration problems. One recurring issue identified in our challenge-identification studies is the difficulty of evaluating machine learning products, particularly those incorporating large language models. In response, my first intervention study surfaces empirically grounded, naturally occurring evaluation practices developed by industry practitioners. These practices help address *syntactic* (🔤), *semantic* (💬), and *pragmatic (🔀) knowledge boundaries* that arise during the evaluation of LLM-based systems.

For the second intervention, our focus extends to a specific aspect of responsible AI: Explainable AI (XAI). Due to the novelty of explainability in ML systems, collaboration around XAI often encounters *syntactic* (🔤) and *semantic (💬) knowledge boundaries*. Effective development therefore requires establishing a shared lexicon and common interpretations of what explanations should contain and how they should be communicated to stakeholders (details on Chapter 4). While our meta-summary study highlighted the importance of explainability and transparency [456, 68], especially in regulated domains [55, 11], our interview study revealed that practitioners often overlook the need for meaningful model explanations in practice. This contrast suggests gaps in how explainability is understood and operationalized across stakeholders. To address this, we design a series of interventions aimed at fostering a shared understanding of explainability and guiding practitioners toward producing explanations that are meaningful to end users.

A third 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 [290, 220], 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.

Chapter 4

Designing Interventions

In this chapter, to demonstrate how we can overcome the identified challenges (*Chapter 3*), we present three studies that design interventions aimed at enhancing collaboration within different spheres, as highlighted in my thesis statement: "*(b) I identify and propose interventions to promote effective collaboration among these development team members.*"

Through these interventions, I articulate and demonstrate principles for addressing collaboration challenges, instantiated across three studies. The first study examines naturally occurring interventions developed by industry practitioners. As industry teams have been at the forefront of building software applications with large language models (LLMs) and faced persistent challenges in evaluating LLM outputs, they have developed a diverse set of scattered practices to address these issues. I identify and synthesize these emerging practices through interviews and validate their relevance and adoption at scale through a survey, resulting in a consolidated set of empirically validated practices for evaluating LLM-based products.

In the second and third studies, I design two interventions to address two distinct collaboration challenges. To analyze these challenges, I draw on Carlile's framework for managing *knowledge boundaries* [58] (introduced in Chapter 2) to design interventions 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 design the intervention that can effectively address and resolve this gap.

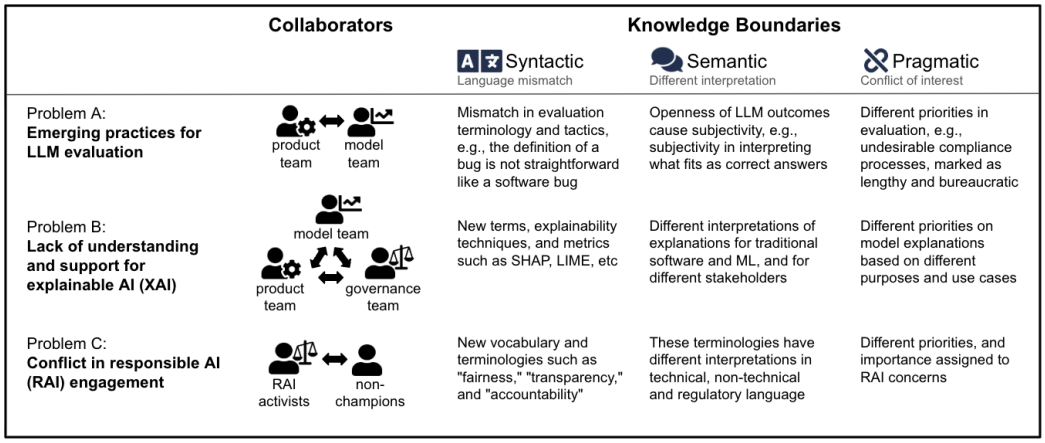


Fig. 4.1 Targetted Problems Across Different Knowledge Boundaries

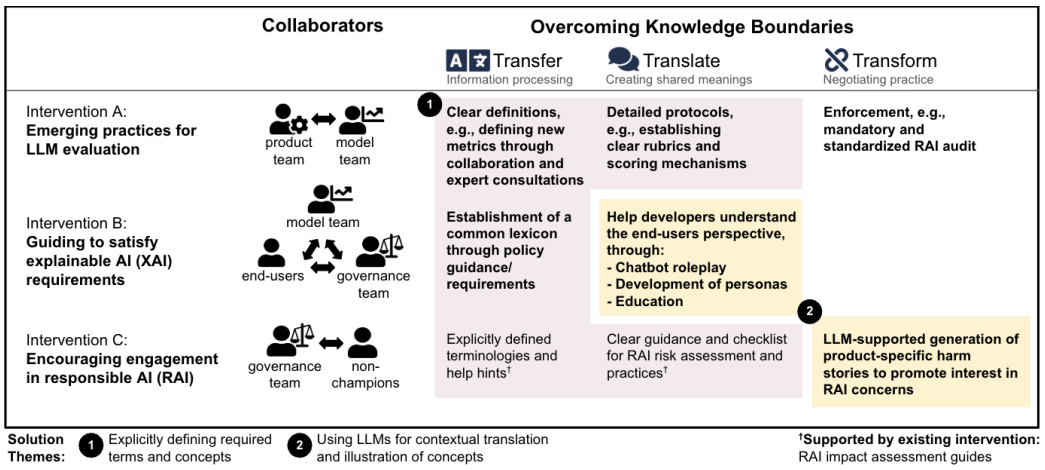


Fig. 4.2 Interventions Facilitating Collaboration to Overcome Knowledge Boundaries

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 help understand the other perspectives. 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. Broadly, we focus on the following two broad themes for our plans of action to mitigate the knowledge gaps in our solutions (depicted in Fig. 4.2):

❶ Explicitly defining required terms and concepts (helps with 🔤 *syntactic* and 💬 *semantic boundaries*)

❷ Using LLMs for contextual translation and illustration of concepts (helps with 💬 *semantic* and 🤝 *pragmatic boundaries*)

We also take a collaborative approach with interdisciplinary stakeholders to co-design guiding artifacts. 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 chapter presents naturally occurring interventions developed by practitioners in response to the challenges they face in evaluating LLM-based products. These challenges and the corresponding practices can be understood through the lens of the three levels of knowledge boundaries. Although the original study did not explicitly categorize the practices according to these boundaries, we provide several illustrative examples here for clarity. A *syntactic boundary* (🔤) arises when established software evaluation terminology does not readily transfer to LLM systems. For instance, the definition of a "bug" is often unclear for LLM-generated outputs. *Semantic boundaries* (💬) emerge due to the inherent subjectivity of LLM outcomes, where determining what constitutes a correct or incorrect response can vary depending on interpretation and context. *Pragmatic boundaries* (🤝) also arise from conflicts in priorities across teams, for example, when software development teams and governance teams differ in the importance they assign to qualities such as responsible AI. To overcome these knowledge boundaries, practitioners have developed a range of practices. For example, teams

In line with the identified evaluation challenges, §3.1.3, and §3.2.3

collaboratively develop new evaluation metrics to address syntactic gaps (🔤 *transfer*, theme ❶), establish detailed evaluation protocols such as clear rubrics to align interpretations (💬 *translate*, theme ❶), and enforce organizational requirements for specific quality attributes, such as responsible AI, through leadership mandates to resolve cross-team conflicts and align organizational priorities (🤝 *transform*).

In line with the lack of standardized processes for XAI, §3.1.3

Our second intervention aims to establish common ground for explainable AI between development teams and stakeholders who require explanations. In ML systems, the notion of explainability differs substantially from its traditional meaning in software engineering, where explanations often rely on deterministic logic and traceable execution paths. In contrast, ML explanations require interpreting complex models whose behavior is learned from data, making it difficult for practitioners to determine what explanations should be provided and how they should be communicated to non-technical stakeholders. As a result, collaboration challenges arise at both the *syntactic* (🔤) level, where expectations and terminology around explanations remain unclear, and the *semantic* (💬) level, where practitioners and end users interpret explanations differently. To address these challenges, I design a series of interventions. First, I introduce an explainable AI policy to guide practitioners on what explanations to generate, for whom, and for what purposes. Experimental results show that the policy helps reduce syntactic ambiguity by clarifying expectations and terminology (theme ❶ in Fig. 4.2). However, it fails to resolve the semantic boundary: the resulting explanations remain overly technical and do not align with end-user perspectives. To address this gap, later interventions incorporate persona development and role-playing chatbots (theme ❷ in Fig. 4.2) that simulate end-user perspectives, helping practitioners better understand stakeholder needs and generate explanations that are meaningful to their intended audiences.

In line with the rare consideration of RAI, §3.2.3

Our third 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 - theme ❶ in Fig. 4.2*)* that 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 [282, 230], who we refer to as *non-champions*. To address this gap, I introduce *sticky stories*: narrative descriptions of potential harms that are concrete, relevant, severe, surprising, and diverse. By presenting tailored harm scenarios to practitioners, our LLM-generated *sticky stories* (theme ❷ in Fig. 4.2) make potential risks more salient and encourage reflection on downstream impacts, thereby fostering awareness and helping realign practitioner priorities with organizational responsible AI goals.

4.1 Intervention A: Identifying Emerging Practices for LLM Evaluation

> The findings of this section have been published in ICSE-SEIP'25*: "Beyond the Comfort Zone: Emerging Solutions to Overcome Challenges in Integrating LLMs into Software Products"* [233].

As discussed in Section 3.2 and 3.3, our investigations reveal substantial collaboration challenges during the evaluation of ML product development. These challenges are further exacerbated in the context of LLM-based products, where the absence of well-established testing processes and the high degree of subjectivity complicate evaluation. As one interviewee noted, *"The hardest thing has been [answering] 'What is a bug?' Like we have gotten into so many arguments [...]."*

Find the details on vague model adequacy goals, and rare online testing and monitoring in §3.2.3 (p. 61-62)

While researchers have made significant efforts to comprehend the *challenges* associated with building LLM-based products [255, 123, 85], – whether in the form of *tools*, *techniques*, and *(best) practices* – have been fragmented. There are many lists collecting various *LLMOps* tools, with many startups competing in this field to aiding prompt engineering, prompt optimization, monitoring, evaluation, and other development, maintenance, and operations tasks [120, 410]. There are also academic papers proposing diverse methods and techniques as solutions to support various aspects of development and evaluation of LLM-based products, e.g., [170, 218, 347, 325, 283]. At the same time, there are increasing concerns about some common practices, such as using LLMs to validate the output of LLMs [228, 326]. Despite the proliferation of these tools and methodologies and lots of (often controversial) opinions about them, their adoption and effectiveness in real-world settings is not well-documented.

Discussed in §3.3.1

Instead of designing a new intervention, we aim to uncover the emerging interventions that practitioners have developed, recognizing that industry is currently ahead in this space, but that these practices remain fragmented and lack systematic consolidation. Given the broad portfolio of products and large community of software practitioners at Microsoft, in this work we uncover the emerging solutions that practitioners across many product teams at Microsoft have adopted for common LLM-related challenges, as well as solutions they are currently exploring. Thus, we adopt a mixed-method study combining 26 interviews across 12 product teams and a survey with 182 responses. Overall, we aim to answer the following research question: **What emerging solutions (tools, techniques, and practices) are the practitioners adopting to overcome LLM-related challenges?**

4.1.1 Related Work

4.1.1.1 LLM-based Products

In recent years, machine learning generally and LLMs specifically have attracted widespread attention, enabling developers to build products around these models [268, 255, 85]. With pre-trained LLMs from vendors like OpenAI or Meta, typically accessed via APIs but possibly also deploying "open" models locally, developers can customize LLMs for specific tasks by developing dedicated prompts. The integration of LLMs can be simple as prompt engineering lends itself to rapid prototyping [153, 394, 192], but can also use sophisticated, multi-step pipelines and integrations with information retrieval systems [183, 411, 152]. In this section, we refer to products incorporating these features as *LLM-based products*.

4.1.1.2 Known Challenges in Building LLM-Enabled Products

After researchers have extensively studied challenges in developing machine-learning solutions and integrating them into software products, see a recent survey [232], more recently, researchers have also explored challenges specifically for integrating LLMs into software products by engaging with industry practitioners and freelancers, e.g. [255, 123, 85, 184]. Focusing on quality assurance problems, we summarize the key challenges identified in these studies in Table 4.2. As part of our own interviews and survey in this research, we found largely the same chal-

lenges, which both replicates and confirms prior work and additionally provides confidence that our study of solutions is conducted in an environment that faces the same challenges as the larger community.

4.1.1.3 Proposed and Emerging Solutions

There is a vast number of *proposed solutions* – tools, techniques, and practices – suggested by researchers and practitioners. These might be suggested in academic papers (e.g., user interfaces to surface ethical concerns during prompt development [375]), popular blog posts (e.g., suggesting testing strategies for LLMs grounded in property based testing [352]), and startups promoting their services (e.g., tooling to validate LLM outputs [274]). Many of these solutions are presented under the label *LLMOps*.

Given the vast number of solutions suggested through academic papers across many venues, all kinds of open-source projects and proprietary tools and services, many of which may have received little evaluation or adoption, we do not attempt a comprehensive survey of *proposed solutions* but refer the interested reader to lists of LLMOps tools [120, 410] and recent surveys of research on prompt engineering [304, 63, 364] and various examples of research automatically or interactively optimizing prompts [165, 329, 276] and evaluating prompts [170, 153, 347, 84, 393, 264]. Instead of a comprehensive survey, our research focuses on identifying *emerging solutions* that are increasingly adopted and sometimes shared as best practices within Microsoft.

4.1.2 Research Design

We employed the established sequential exploratory mixed-methods research design [69] that performs research in two phases: In a first qualitative phase we explore challenges and emerging solutions through interviews. Then, in a subsequent second phase, we quantify the observed challenges and emerging solutions through a survey with a much larger sample size. This enables an in-depth open-ended exploration of emerging themes beyond a predefined list but also allows us to provide quantitative insights about the prevalence of these emerging solutions in the studied teams at Microsoft.

4.1.2.1 Interviews

In the first research phase, we created an interview guide to broadly explore the *current state*, *challenges*, and *emerging solutions* across the entire lifecycle of LLM-enabled products and features. After feedback from experts and a pilot test, we refined the guide, emphasizing the *evaluation* phase and adding relevant questions to thoroughly explore this topic.

To recruit a varied and representative group of interviewees aligned with our research goals, we used *purposive sampling* [56], explicitly recruiting interviewees from different teams from different departments within Microsoft that had integrated LLM components into their products for various use cases. In addition, we aimed to recruit multiple team members in different roles (managers, engineers, and data scientists) from each product team to explore different perspectives. In total, we conducted four pilot interviews, followed by 26 interviews we analyzed for this paper (cf. Table 4.1).

Our recruitment strategy involved reaching out through company-internal channels and using snowball sampling [108] to identify suitable interviewees.

We conducted the interviews virtually, recorded them with consent, and transcribed them using an in-house tool. We followed the usual best practices for interviews [316] to establish rapport and encourage open dialogue.

We analyzed the interview data with standard qualitative methods [135, 178], involving open coding and memoing to develop and iteratively refine a codebook for challenges and solutions. We established inter-coder agreement on one interview (percentage agreement between two raters coding independently = 80%).

4.1.2.2 Survey

Drawing from our interview analysis results, we identified the key challenges and emerging solutions and designed our survey to quantify their prevalence within the company, using various rating scales. This included questions about agreement to statements about challenges, ratings of difficulty of select activities (from ‘extremely hard‘ to ‘extremely easy‘) and ratings of whether participants tried and used techniques (from ‘tried and would recommend’ to ‘would like to try’ to ‘tried but did not find useful’). Similar to the focus in our interviews, we again focused the survey on the quality assurance aspects of developing LLM-based products,

Products	**Participants**	**Roles**
G1	P01, P08, P18	Manager, Engineer, Data Scientist
G2	P03, P05, P06	Engineers
G3	P02, P9, P10, P17	Manager, Data Scientist, Engineer
G4	P04	Engineer
G6	P11	Engineer
G7	P12, P14	Managers
G8	P13, P24	Researcher, Data Scientist
G9	P15, P16	Data Scientist, Manager
G10	P19	Manager
G11	P21, P22, P23	Engineers
G12	P25, P26	Manager, Data Scientist

Table 4.1 Interview Participants

allowing us to explore this topic in more depth while keeping the survey to a reasonable length. In addition, we also incorporated a separate section in our survey for non-LLM components to compare and contrast the responses — this is beyond the scope of this paper.

Since it is difficulty to identify who exactly works on LLM features at Microsoft, we oversampled and included anyone who committed to a repository containing LLM code as a potential survey participant. Despite expecting a lower response rate [277], it was for us crucial to reach as many LLM practitioners as possible. To ensure accuracy, we incorporated qualifying questions into the LLM section of the survey. In total, we emailed 12,878 practitioners, received 977 automated out-of-office replies, and 332 responses (response rate < 3%), among which, 182 responses were directed to the LLM component of the survey.

In this paper, we report primarily quantitative results from ratings regarding challenges and emerging solutions.

4.1.2.3 Threats to Validity and Credibility

Our research has the kind of limitations typical for this style of research. Both of our interviews and surveys have a risk of response bias, where the respondents may provide inaccurate responses due to misunderstanding, social desirability, and other factors. While the survey affords some generalizability to the population of developers at Microsoft, given that participation in our survey and interviews was

voluntary, those who chose to participate might be inherently different from those who did not, potentially skewing the results. Results might be influenced by other practices at Microsoft, hence readers should be careful when generalizing results to other organizations. Also despite following standard practices for coding and careful design of our survey and interview protocol, we cannot entirely exclude biases introduced by us researchers.

4.1.3 Findings

We provided an overview of quality-assurance-related challenges identified in the literature and confirmed in our interviews and survey in Table 4.2. In the remainder of this section, we report emerging solutions reported by the interviewed and surveyed practitioners, focusing on quality assurance).

Challenges, disruptions, and emerging solutions: Emerging solutions do not always match perfectly the identified challenges and many solutions implicitly or explicitly address multiple challenges. To organize the emerging solutions, we organize them by themes we call *disruptions*.

Specifically, we refer to *challenge* as inherent difficulty or obstacles associated with and caused by LLMs and use *disruptions* to describe how these challenges disrupt traditional software development practices and cause day-to-day disturbance for practitioners in their established workflows and practices (especially for practitioners new to LLMs) as they attempt to address these challenges within their known established practices and tools. The experienced disruption then drives the exploration and adoption of new solutions to overcome them. In essence, the sequence is: One or more challenges lead to disruptions in development practices, which in turn trigger the adoption of a new solution. For example, the challenge *lack of specifications* (C1, Table 4.2) leads to perceived *insufficiency of objective metrics*, which results in the formulation and adoption of the emerging practice of *combining multiple qualitative and quantitative metrics* (Emerging Solution 2).

For each emerging solution, we include quantitative evidence from our survey (marked with). Based on the practitioners who rated that they (a) *'tried and would recommend'* the solution, (b) *'tried and found it somewhat useful,'* and (c) *'tried but did not find it useful,'* we compute the percentages of respondents who

Known and Confirmed Challenges	**Our Participants' Views on the Challenges**
C1: **Lack of specification**. The definition of a bug is ambiguous and debatable [208].	29.6% P9, P10, P14 P14: *"How do you test these things are doing well or not doing well? What's the bar."*
C2: **Subjectivity**. Determining expectations and what fits as correct answers is challenging [326, 13, 218].	36.7% P2, P9, P10, P14, P16, P17, P24 P17: *"So in these cases it's more subjective to even measure the quality on the output, and figuring out the rubrics."*
C3: **Metrics dilemma**. Developing the right metrics set is complicated [325, 85]. How do we know which metrics to use, how to measure them, and if the chosen metrics are appropriate? (a) The selection of metrics tends to be *ad hoc*. (b) Many teams rely on common known metrics without considering whether their *specific use case fits* to them. (c) Defining *suitable measurements* for metrics is difficult because it's subjective and not straightforward. (d) There is a significant *disparity between offline and online metrics*. Comparable sets of metrics for both offline and online evaluation are necessary, but currently lacking. (e) Online metrics provide *weak signals* and offer little insight into output quality.	P23: *"That's really hard to map back to some of these really core metrics [...] So I think that's sometimes like more art than science."* (a): 46.5% P5, P17, P23, P25 P17: *"People just kind of end up doing what they need to do in an ad hoc way."* (b): 31.2% P5, P7, P20 P5: *"We just pick the ones that we are interested in and we just run the test."* (c): 46.5% P1, P4, P7, P10, P15, P16, P26 P1: *"Measurement of the offer is another big technical challenge."* (d): 49.7% P1, P26 P26: *"Our pain points right now are more around metrics and like representativeness of offline data [...] you get these offline metrics that they will translate to something and online like that's a big gap."* (e): 30.3% P8, P22, P26 P8: *"I don't think we have any kind of signal which will trade that we are doing an awesome job on generating content."*
C4: **LLM properties**. LLMs are made non-deterministic; teams struggle with testing, as outputs are not consistently reproducible. LLMs often hallucinates, making them unreliable [326, 85, 255].	57.7% P1, P2, P3, P4, P5, P7, P9, P10, P12, P13, P14, P15, P17, P18, P22 P9: *"How do we deal with this nondeterminism problem?"* P10: *"It's gonna hallucinate sometimes. It's gonna not do what you ask it to do."*
C5: **Lack of robust evaluation methods and pipeline** [85, 255]. (a) Many teams *lack proper evaluation mechanisms* or have *inconsistent evaluation practices*. Unit testing prompts are difficult and there are no effective methods for evaluating non-deterministic models beyond basic health checks. (b) Teams report the absence of a cohesive *evaluation pipeline*, having separate evaluations on different platforms with no integration. (c) Evaluations using LLMs are unreliable, frequently requiring multiple attempts to determine whether an issue is due to *flakiness* or another factor. (d) The sheer number of *variables* involved in testing complex prompts makes it hard to analyze all dimensions and user inputs confidently. (e) Creating test cases depends heavily on *synthetic data*, often generated using LLMs themselves, as access to real data is limited. (f) Numerous instances of bugs reaching production highlight *inadequate testing* and overlooked *edge cases*.	(a): 36.3% P2, P5, P6, P11, P12, P16, P20 P12: *"We are working on building more complete test suite because current test suite is still very limited."* (b): P1, P2, P6, P15 P2: *"We don't have a strong evaluation pipeline right now."* (c): 65.6% P6, P7, P9, P16 P6: *"It's been really flaky to where it'll fail a lot of the time."* (d): 52.0% P1, P2, P4, P6, P7, P9 P4: *"It's like one prompt. We want to be able to do that as one evaluation, and I'm still looking for where I can run that automatically so that I could get both the breakdown of scores and the single score. There can be pros and cons to it, like maybe we want to run it all at once, or separate it but also giving the context. That's gonna give all these dimensions [that] can help it score and give more. Like more accuracy for each specific score when it knows the other dimensions. Uh. And so we're still like kind of playing around with which one is the right type of evaluation. And there's lots more."* (e): 42.1% P12, P15, P16, P25 P15: *"A big challenge is like getting right data to test these LLMs. Like across everything. So creating synthetic data, [trying] understanding like what they can [look] like."* (f): P2, P3, P4, P5, P6, P11, P14, P15, P16, P19 P5: *"Edge cases where [LLM feauture] just actually blurbs out it's entire command"*
C6: **Manual efforts**. Manual testing is common but labor-intensive and inefficient. Some teams avoid automation to move quickly but later experience drawbacks. [218, 184]	76.6% P1,P2, P9, P10, P11, P12, P14, P15 P9: *"this process is extremely manual, right? It is not possible to fully automate it because it requires us to keep our brain on."*
C7: **Infrastructure constraints**. There is an over-dependence on the current evaluation infrastructure that lacks flexibility for all use cases. The causes of test failures are not clear, documentation is sparse, and communication is a burden. In some instances, responsiveness and support are inadequate [208].	43.1% P1, P2, P4, P6, P7, P16, P25, P26 P6: *"Kind of been challenging because [infrastructure] is owned by another team [...] Nobody on our team really has access to that or fully understands how it's powered and we run into problems with it all the time. So like the data, we'll be looking at the dashboard and like we don't have data for the past two weeks [...]. And then we have to go find someone and poke them and be like, hey, what's going on?"*
C8: **Model migration issues**. Evaluating models' post-migration is problematic; minor tweaks can cause substantial changes, complicating regression testing [184].	40.4% P12, P14, P16 P12: *"If you like migrate the model or any changes in the prompts, you have to go through everything again."*
C9: **Compliance**. Compliance processes are lengthy, manual, and bureaucratic, involving excessive paperwork and manual effort[255].	30.7% P1, P2, P3, P5, P7, P8, P11, P12, P13, P14, P15, P16, P23 P13: *"That's (compliance) like a huge, huge challenge and I think just in general, you know, Microsoft is super careful with customer data, which obviously is good."*

: % of responses from survey indicating that the challenge is 'hard' or 'extremely hard;' : Interview participants reporting the challenge; : Representative interview quotes

Table 4.2 Challenges of evaluating LLM-based products

adopted as *a+b+c,* who was *satisfied* as *a+b*, and who *did not find the solution useful* as *c*.

Disruption A: Evaluation metrics change substantially (C1, C2, C3, C4, C6).

Whereas metrics for test suite quality for traditional code (e.g., coverage) and metrics to evaluate qualities (e.g., response time, error rate) are fairly standardized and often automated, product teams at Microsoft often need to create custom metrics when evaluating LLM features and face substantial manual effort. For example, subjective qualities such as fluency, saliency, consistency, and creativity are considered crucial for generating text based suggestions in some product use cases (P1, P3, P5).

Represents both syntactic and semantic boundaries

There is usually no clear oracle (not even the labels of traditional ML model testing), but many outputs may be equally correct or acceptable for a given input.

Many more subjective measures are open to interpretation and teams routinely rely heavily on manual human judgement, making the process time-consuming and laborious ((76.6% adoption) survey respondents mentioned spending manual effort). Also, even with human evaluators, ensuring objectivity and reliability can be difficult, and may introduce inconsistencies and biases [13].

Overall, developers often need to creatively explore new metrics rather than relying on established ones – for example P4 mentioned, *"I just created this. It's called the [-] metric, which looks at like 10 dimensions. You can score across each dimension and then a final score."* Interviewees find metric creation challenging, as P25 pointed out, *"it's really a social science problem more than a science problem."* The process of defining custom metrics lacks a systematic foundation, and teams frequently encounter difficulty in defining the right metric for their specific use case, as P7 rightfully mentioned *"it's just frustrating to come up with some scoring criteria."*

Emerging solution 1: Defining custom metrics through iterative collaboration and expert consultations (57% adopted, 55% satisfied, 2% did not find useful).

Product teams have started to use initial brainstorming sessions aimed at figuring out the various dimensions of quality related to their specific LLM use cases (P1, P4,

P15, P16, P17, P22, P25, P26). Instead of approaching this as a typical engineering task, they treat it more as a research phase. Many teams engage with domain experts (e.g., *linguists*, P1) and researchers (P15, P16, P17) to explore specific use cases and identify metrics for their desired response qualities. P16 also mentioned involving LLMs as a judge to evaluate the appropriateness of such metrics, and human judgement to validate it further: *"Like asking the LLM as a judge metric to evaluate how good those types of [metrics] are. [...] We have a data scientist who works on this. There's quite a bit of iteration, and then also we worked with [the] PM and the feature crew to explain what we've done and then ask for feedback as well to iterate on."* Teams also mentioned iterating over it multiple times to confirm they have the right metrics, such as *"You have to run through this couple of times to make sure you have the right set of metrics"* (P1).

⚙ Emerging solution 2: Combining qualitative and quantitative metrics to evaluate the multifaceted outputs effectively (📊 54.4% adopted, 53.7% satisfied, 0.7% did not find useful).

Most interviewed teams (P2, P4, P8, P9, P15, P16, P17, P18, P22, P26) have started combining subjective metrics (e.g., fluency) with 'objective,' more mechanical, more easily quantifiable metrics (e.g., evaluating the syntax of a generated formula) to make evaluation more comprehensive. For example, with reference to evaluating LLM-generated conclusion slides, engineering manager mentioned *"We do both structured objective metrics and subjective metrics about how good is this slide,"* where quantitative criteria like bullet point count and sentence length are assessed alongside qualitative elements such as content-groundedness. This hybrid approach is echoed by other teams: *"We kind of have a merge of two things. One is still this correctness [...] You could find different ways to sum the two columns, but there's generally a functionally correct answer. [...] There's on the other side [...] We don't have a spec [...] So in these cases it's more subjective to even measure the quality on the output"*.

⚙ Emerging solutions 3 & 4: Evaluating subjective metrics using LLM validators (📊 50.6% adopted, 47.3% satisfied, 3.3% did not find useful). Establishing clear rubrics and scoring mechanisms (📊 33.8% adopted, 33.1% satisfied, 0.7% did not find useful).

The usage of LLMs as validators for evaluating LLM responses has gained attention in the literature [405, 157, 228, 326] and has become a common practice for

many teams at Microsoft (P1, P2, P3, P4, P6, P9, P10, P11, P15, P16, P17). These teams use rubrics to measure certain qualitative factors such as fluency (smoothness and proficiency of language), salience (contextual appropriateness), and consistency (uniformity and lack of contradictions) that signal the expected response quality of the model. Microsoft has developed frameworks (📊 15.4% adoption) to facilitate this technique, garnering positive feedback from many, P1:*"Today you can actually use large language models to evaluate the outcome of large language models, which is fantastic."* Of the twelve product teams we engaged with, nine incorporate this approach in at least one feature team, highlighting its effectiveness in reducing manual efforts.

Despite these benefits, the approach comes with its potential pitfalls. One key reported issue among our participants appears to be the perceived unreliability of the validator LLMs. Some participants describe the validator LLMs as *"flaky,"* suggesting that there are inconsistencies in the judgement delivered by these models for similar responses. This inconsistency makes it challenging to rely solely on these LLM validators for response evaluation, thereby necessitating the involvement of human validators. However, teams still found it beneficial to *use LLMs for an initial validation phase, followed by a secondary review by human validators* to verify the initial pass/fail test results.

One big concern of this approach research has found is that using LLM-as-a-judge to directly evaluate a model's output can result in a model assigning a high score if it was likely to produce that output and not necessarily based on criteria [228]. As a result, an excessive dependency on these LLM validators could potentially lead to oversights of such issues by human validators, thereby instilling a misleading sense of validation accuracy. Our observations have also identified an unwarranted over-reliance on this subjective method across various teams. Notably, we found teams using LLM validators even in instances where adopting more objective measures could be better suited, such as for measuring the syntactical correctness of generated code. This finding underscores the need for thoughtfulness and careful consideration in selecting validation techniques (see *emerging solution 2*).

⚙ Emerging solution 5: Build a validator allowing a range of acceptable outputs instead of conducting single-valued unit tests for objective metrics (📊 43.3% adopted, 42.6% satisfied, 0.7% did not find useful).

To improve the robustness of their objective evaluation, many Microsoft teams adopted test designs that accept a range of multiple acceptable answers or entirely switch to test general criteria instead of expecting one acceptable value (P2, P9 P12) – which mirrors ideas from property-based testing [67, 352]. They systematically implement this by using techniques such as regular expressions, search patterns, similarity algorithms, or simply by matching with a pre-approved set of cases. P12 mentioned their strategy of using a similarity algorithm: *"We know what's a good answer; we can calculate the similarity of the generated answer to the good answer [using a comparison algorithm], and if the similarity is close, although there could be some variation, but if the similarity to the known good answer is high, then we feel the quality is high."* Similarly, P2 mentioned using a formula evaluation pipeline for formula verification: *"So whenever there's a formula that shows up, we make sure that the formula goes through an evaluation pipeline before it comes back."* The notable benefit of such systematic evaluation techniques is that they can be largely automated, which substantially reduces the need for labor-intensive manual work.

Disruption B: Common assumptions about test processes and environments break (⊞ C5, C6, C9).

Model and prompt evaluations use different infrastructure than traditional unit testing and rely often on large amounts of data that needs to be handled separately. And while developers are used to continuous integration for code tests, they do not necessarily set up similar infrastructure to (automatically) re-run offline evaluations whenever the model or prompt pipeline changes.

Semantic boundary

Also, given all the differences and new challenges when it comes to LLMs (including the ones previously discussed), practitioners often do not have experience or templates of how to approach model and prompt testing, resulting in many ad-hoc processes. These ad-hoc processes tend to be inefficient and can result in practitioners dealing with repetitive, redundant tasks, causing frustration and suboptimal outcomes.

⚙ Emerging solution 6: Automating offline evaluation to run periodically on a schedule (📊 29.6% adopted, 28.9% satisfied, 0.7% did not find useful).

A few teams (P4, P6, P22) we spoke to value the practice of periodically scheduled, automated offline evaluations. For example, P22 mentioned, *"Most of the offline eval is heavily automated. I mean basically you wanna have a scheduled run every day just to keep track of the general metrics."* P6 also voiced her interest in adopting such a routine, given she saw other teams benefiting from it: *"They have [tests] just running on a schedule [..], like every hour or something. Evaluating sort of their offline response quality and they have a dashboard for that too, and I've asked some folks how can we do that because I think it's also valuable to have the offline evaluation happening periodically."*

⚙ Emerging solution 7: Establishing internal team standards for evaluation processes and pipelines (📊 48.7% adopted, 48% satisfied, 0.7% did not find useful).

To overcome this, several teams try to establish internal *standards* for the evaluation process or pipeline (P2, P4, P9, P15, P16, P22). As P15 expressed: *"So as we were like developing these, we started facing challenges of like, OK, we can do this in an ad hoc way and we'll have to do this many times. So we need to start building things where we can do it at scale. [...] [Do not want to] just put data scientists on evaluating the same prompt on different models every time because that is kind of redundant. So that's where I think the thought for like standardization and like building standard pipelines and tools that came in."* Beyond individual teams, there are efforts within Microsoft to provide default processes supported by standardized tools to benefit all product teams, as P11 stated: *"So we're using fairly standardized tools that Microsoft other teams have provided us. [..] So what we need to do is hook up those services to our back end, which is fairly standard."*

Disruption C: Engineers require new skills to handle LLM evaluations (⊞ C4).

Traditional software engineering education did not teach the skills necessary for evaluating LLMs (which did not exist even a few years ago). Such evaluations often require adopting a research-focused mentality where developers need to explore hypotheses, define custom metrics and measurements, and iterate over variations – these are not commonly taught and practices in traditional software engineering. We observed that this shift can be challenging for engineers. For example, P9 explained

💬 Semantic boundary

how engineers lack the understanding that LLM evaluations are not similar to unit tests where all the test cases need to pass: *"People who are trained doing research usually have the background of understanding that looking at the analysis resources is important. It's not a unit test, right? We often run into engineers thinking about these as unit tests. Say for example, this is a benchmark, right? It is OK that there is 63 failures. It just shows me that limits of the system [...] And engineers tend to think about it as ohh [...] I need to make the tests of unit test quality which is 100% pass rate."*

⚙ Emerging solution 8: Involving data scientists in authoring tests, as they understand the limitations of the model better (📊 33.5% adopted, 32.1% satisfied, 1.4% did not find useful).

At Microsoft, teams (P7, P9, P10, P14, P15, P18, P26) often recruit explicit help from data scientists to author test cases, as P14 mentioned, *"We have probably like 1 dedicated person from the data science team who's helping us kind of complete some of that testing."* Interviewees find that involving data scientists in authoring prompts and test cases improves the testing process. With their understanding of the model's capabilities and limitations, data scientists can more rigorously assess the data involved to generate the test cases, as P9 mentioned, *"That's actually the place where I think the domain of data science is important. [...] really carefully thinking through the data."*. They are often well-versed in spotting trends, anomalies, and potential pitfalls that may be overlooked by others. Interviewees believe that data scientists are better positioned to preempt potential issues and create test cases that cover a wider range of scenarios, leading to a more comprehensive LLM evaluation. Conversely, the absence of a data scientist in the team can prove to be a roadblock, as P5 explained: *"We don't know what's going on and we don't know how to remedy this and we don't have a data scientist on the crew to under to basically explain what was going on."*

Disruption D: Even with extensive evaluations, LLM solutions remain unreliable and developers have difficulty establishing trust (⊞ C4, C5).

Even with substantial evaluations and more systematic and automated processes, teams often still doubt the quality and reliability of their features, as P10 explained: *"It's easy to just generate something, right? It's easy to have a solution and get up and running with it, [...] It's a lot harder to really know, like where the quality is at [...] lot harder to kind of have confidence."* The practitioners lack of confidence in LLM responses are not unfounded, as they have recounted numerous incidents that validate their concerns, as P5 mentioned: *"There have been [...] multiple times where we've been caught off guard [...] like any non deterministic system, there would always be edge cases where [the LLM] just blurbs out it's entire command, which is a huge risk. [..] It's been on the news."*

Also, mentioned in §3.2.3 (p. 61)

As a general theme, we see much more emphasis on testing and monitoring in production and on providing guardrails beyond the model.

⚙ Emerging solution 9: Employing canary release strategy to enhance confidence in the LLM outputs (standardized practice, no adoption statistics available).

All interviewed teams use canary releases. A canary release strategy is the idea to expose a new feature or update initially to only few users and monitor the behavior of the system, before rolling it out to more users, or rolling it back if problems are discovered [227].

Microsoft employs several stages of audience groups, called 'rings,' through which a new release passes before becoming available to the public. As P3 explained: *"We'll do [RING-2]. We'll keep it in internally for a while [...] before we actually go beyond. So there's kind of like a breaking in, burning in, stabilization period."* This helps catch any unforeseen bugs or issues that might not have been apparent during development or initial testing stages. For instance, P10 discovered from the earliest release ring that the LLM was not performing up to par: *"We had a [RING-1]. We had like an early release of the [LLM] experience, and it was not good. [...] It was pretty buggy."* Beyond detecting bug, canary release also allowed interviewees to gather early user feedback on the new feature or update, that are used to make improvements or adjustments before a full deployment,

P4: *"So then once we get into Microsoft rings in production, we can look through [customer] feedback [...] and then figure out which were the right comments we wanna address."* Beyond internal rings, Microsoft also has customers under non-disclosure agreements (NDAs) who participate in early access programs, use the product before others, and provide valuable feedback, as explained by P14: *''then we have our EAP, our early access program. And these are like a set of customers that we work very closely with. Those customers are very open about, this is the scenario we tried and it didn't work, and then we can use that feedback."*

⚙ Emerging solution 10: Running A/B testing for tracking changes in different versions (e.g., prompt updates, and model migration) (📊 36.7% adopted, 36.7% satisfied, 0% did not find useful).

P22 appreciated the utility of online A/B testing frameworks, stating, *"I think the online AB test framework, I think it's pretty nice and it's pretty detailed."* Several teams (P8, P12, P16, P18, P21, P22, P24, P26) also mentioned their consistent use of A/B testing for different types of changes such as change of underlying model, and updates in the prompt engineering pipeline, such as P18: *"during the migration we run AB testing like 50-50% and try to compare the two models directly."* P16 also mentioned running audits on the A/B testing to make sure its serving its purpose: *"On our the AB testing and metrics we're doing this audit to understand if the telemetry is implemented correctly, because that's one area that we want to make sure things are done well in too."*

⚙ Emerging solutions 11 & 12: Establishing extensive guardrails (📊 44.2% adopted, 41.3% satisfied, 2.9% did not find useful). Monitor systems to trigger alerts automatically if something goes wrong (📊 36.4% adopted, 34.2% satisfied, 2.2% did not find useful).

Some teams (P1, P2, P3, P4, P8, P10, P11, P16, P17, P22) at Microsoft have implemented robust guardrails, including API format checks, response structure validations, specific pattern regular expressions, and other rule-based checks. This has improved practitioners' confidence in the overall system, P8: *"I dont see a risk like the content would be able to mess up with the engineering system. Reason being we have put a lot of good guardrails to make sure that the reliabilities are good enough [...] all like prompt injection, check block list and like safety harm, we call that content safety, and couple of mores like jailbreak classifiers."*

However, there can occasionally be a false sense of security from these guardrails. For instance, P11 mentioned the three-level guardrail system, falsely considering the trained model itself as the first guardrail, *"LLM is usually a already trained or fine tuned to to to not answer to an appropriate questions So it's already what you call censored"*, instructions to the model as the second guardrail, *"the second thing we have a control over is the metaprompt. So you we can say please answer politely. Please don't answer any legal or medical questions and so on"*, and finally the real guardrail, *"The third one is [tool] policy filters. So the the input and the output are run through a filter. They will look to see if there's any bad words and so on and so forth."*

Despite these misconceptions, there's an impression that guardrails have improved over time. Practitioners (P3, P7) also highlighted the development of systems to automatically alert them if issues arise, assuring improved protection and monitoring. As P3 pointed out, *"It's gotten better, guardrails over the years. There are systems now that know the prompt you sent in and watch for the prompts coming back out in the result and can can automatically trigger or recycle or can tell you that it couldn't get an answer or can do things like that."*

Microsoft also maintains a framework with universal guard lists, which set uniform standards across all its products while also reducing the individual teams' workload in establishing guardrails. However, one concern we found among a few practitioners is how these guardrails can sometimes be too sensitive and flag innocuous interactions with customers, which can lead to customer frustration, as P15 mentioned *"we are not able to understand the sensitivity of the guardrails that we are building [...] I remember in January or December of this year we got our feedback [...] somebody said that yeah, [feature] was overblocking. [...] So Power user wrote on Reddit that like I mean actually they wrote a blog like that."*

Disruption E: Existing approaches to telemetry and monitoring need to be revised (⊞ C3, C9).

Many of the interview participants mentioned that the current telemetry methods, originally developed for non-ML software, are not strong indicators when it comes to assessing the quality of LLM-generated responses. As P8 elaborated: *"I don't think we have any kind of signal which indicate that we are doing an awesome job*

Also, mentioned in §3.2.3 (p. 62)

on generating content.". Many teams depend on the classic telemetry metrics, such as direct customer feedback, thumbs up, and thumbs down, which typically have a very low response rate.

A particularly frequently mentioned problem is no-eyes debugging, that is, to gain insights into system behavior at runtime without accessing user data, which should not be revealed to developers due to compliance and privacy regulation. This problem exists in debugging traditional software systems too [401], but is reported as even more severe with the open-ended nature of interacting with LLMs where users can input any text and the models can produce any response. Developers struggle to find better signals about when problems occur private in LLM outputs (e.g., as opposed to crash logs).

For example, P13 explained, *"All of the commercial data like [LLM responses] are eyes off. So we cannot actually see it. So how you go and like understand what people are doing and what's working and how to make it all better if I can't actually even see it [...] that's like a huge, huge challenge and I think just in general, you know, Microsoft is super careful with customer data, [...] so you just have to do the best you can while keeping you know sort of customer data privacy."*

⚙ Emerging solution 13: Developing new and multiple types of telemetry metrics that may better suit LLM solutions (📊 58.1% adopted, 57.4% satisfied, 0.7% did not find useful).

Practitioners employ several types of telemetry to assess the performance of their LLM features. For example, some practitioners (P7, P17, P20, P24, P25, P26) track *apology rates* from LLMs, an indicator of the models' confidence in their responses. There is another set of metrics that almost all Microsoft practitioners use predominantly, called *seen-tried-kept*, that measures whether customers noticed the feature, tried it, and accepted or used the generated response. Moreover, Microsoft have a custom telemetry metric called ASHA (Aggregated Session HAppiness) to estimate customer satisfaction, referred as the *happiness index*. This measures the success rate of each user sessions, such as, if a user employs the LLM feature ten times without failure, they have a successful ASHA session. P17 elaborated the metrics they use: *"We have standard KPIs, [...] we use Asha, which sort of this is for aggregated session happiness scores. So we use that pretty heavily and we rely a lot on the customer feedback thumbs up and thumbs down. [...] beyond that, there's like seen tried and kept, of course [..] also say an engagement measurement*

of do they continue to engage with the [feature]. And we also have a measure of apologies like how often we're apologizing in a response, right. So that's another indication that we're probably not giving high confidence answers." Apart from all these, they still have other traditional metrics such as monthly active users (P15).

Research teams in Microsoft are also actively engaged in establishing a robust set of quality metrics for the product teams, largely based on the analysis of system logs, as P13 mentioned: *"the sort of mission in terms of coming up with a set of measures based on people's naturalistic interactions with [LLMs] in order to give guidance to the product team. [...] They track of course a bunch of key metrics, one of which is percent of conversations that were successful. And so how we created that measure with them was what indicator that metric [...] we call inferred SAT, inferred user satisfaction. Every single [LLM] interaction gets a score"*

⚙ Emerging solution 14: Use the LLM-as-a-validator strategy to gain granular insight in production behavior without revealing private data (📊 24.8% adopted, 24.1% satisfied, 0.7% did not find useful).

Instead of solely relying on established but weaker signals like *seen* or *kept rates*, a few teams are trying to identify and design stronger signals for their use cases. For instance, P17 mentioned an approach where validator LLMs are used to review the eyes-off user conversations and report back with an assessment of the quality:*"So there's one path we have which compliant eyes off chat analysis [..] where an LLM is able to look over [user-LLM conversation] in an anonymized fashion and basically report back if it's done well, if the conversations are good or not [...] So that's again relying on LLM evaluators and eyes off chat analysis. It's one way we try to get signal."* P6 also mentioned using a similar approach to track whether the quality of responses is improving or deteriorating over time: *"We have online [LLM-as-judge] evaluation, which is sort of like evaluating the actual conversations [...] eyes off, so it doesn't actually save the evaluation, but it's sort of evaluates it at the time it's happening. And then it saves that, and we have this sort of daily chart that shows us how is the quality of these real responses happening or changing over time."*

P15's team went a further step up to validate whether the metric delivers accurate signals by showing the LLM's evaluation to the user and asking for their agreement or divergence: *"So we had built in an evaluation within the product itself. The first few versions of the product where the user was able to look at the rating that was*

given by the LLM and then select if they agree or disagree with that rating and how much rating they would give it. And then submit logs. [...] those logs are shared back. You don't read the customer content, but you know a certain [response]'s model score and user score, and if there is a difference, it means like the prompt engineering is not [working well]." Additionally, P15's team is running a similarity analysis algorithm to track the variation between the text generated and the text ultimately used by the user: *"What was the output from the LLM? And then what was the final [text] that was [used]? So the plan is to do text distance analysis on the output of the LLM to the final [text] that was [used], because then you can actually understand how much of a variation was coming in terms of like the final [text] content.*"

Disruption F: Lack of focus on system-wide evaluation of LLM-based products (⊞ C5).

The new challenges with evaluating LLMs tend to draw attention to model and prompt evaluation approaches (forms of unit testing) and away from more holistic evaluations of the entire system (integration, system, and acceptance testing).

Also, mentioned in §3.1.3 (p. 33)

⚙ Emerging solution 15: Setting up an end-to-end test automation infrastructure (📊 38.2% adopted, 37.6% satisfied, 0.6% did not find useful).

Recognizing the need for comprehensive system testing to assess the full functionality and efficiency of LLMs in the system, Microsoft's infrastructure teams have developed an end-to-end testing framework. Being a consumer of the framework, P17 explained the need for it: *"We're evaluating the LLM; we likely wanna evaluate the full system, the end to end system, because the client [program] is a part of our system that does non trivial computation as a part of our conversation with our LLM. There are plenty of great tools to evaluate a single LLM call or maybe like a couple of tightly interconnected LLM calls [...] but as soon as we execute an [client program] and that gets fed into prompt, the tooling support doesn't exist. So [infrastructure team] having to create our own.*" Although this platform is relatively new, many Microsoft product teams ((📊 39.4% adoption)) have already adopted it.

⚙ Emerging solution 16: Conducting comprehensive tests beyond unit tests, including tests for reliability and availability (📊 61% adopted, 60.3% satisfied, 0.7% did not find useful).

Beyond evaluating LLMs, Microsoft teams also prioritize regular software testing. This includes conducting unit and regression tests, among others, which are integral in identifying and rectifying bugs throughout the software system. For instance, P2 noted the importance of unit tests in preventing issues: *"the app teams have a very extensive like unit test pipeline. [...] As long as our tests aren't failing, then we're not breaking anything else in the product."* Similarly, P11 referred to regression testing: *"if we change the [filters], if we change our meta prompt, you know anything that's substantial or the large language model itself, if you switch from 3.5 to four, then you will have to rerun the test."*, P7 mentioned UX testing: *"We also have some UX tests or end to end tests on validate that the the answers are coming as expected."*, and P3 discussed reliability testing: *"We can measure [reliability]. We wanna get 99%. Then you go until you hit 99%. It's a very clear goal."*

Disruption G: Responsible AI, being a relatively new concept for engineers, has a steep learning curve and might come across as bureaucratic (⊞ C9).

Practitioners often consider Responsible AI as a secondary priority [232, 282], particularly when they are already burdened with a substantial workload to develop, integrate, and evaluate such products with models. However, improper prioritization and a lack of standardized Responsible AI (RAI) practices can lead to legal, ethical, and reputational risks for organizations. Microsoft places a high priority on responsible AI practices [216], incorporating ethical considerations, transparency, and human-centric values into their development processes to ensure the responsible deployment of AI technologies, including LLMs.

Pragmatic boundary

⚙ Emerging solution 17: Standardizing Responsible AI (RAI) evaluation and practices – bias and fairness evaluation (📊 39.7% adoption), safety evaluation (📊 55.9% adoption), robustness evaluation (📊 33.5% adoption), transparency and explainability (📊 27.9% adoption), privacy compliance (📊 55.8%

adoption), security and vulnerability evaluation (59.8% adoption), other (11.7% adoption) (). As P7 highlighted, *"It feels like right now it's we are doing heavy [RAI] testing around that area just because we are new, we don't know what's gonna happen. We just want to play safe."* This statement emphasizes that Microsoft is placing a significant emphasis on RAI assessment. The company has assembled dedicated teams and specialists who focus on Responsible AI practices, particularly for large language models these days. They employ different types of RAI checks, P21: *"We basically what we call responsible AI filtering where we do some sort of separate checks to make sure that we're not outputting sort of racist or sexist or other comment things that shouldn't, that totally inappropriate to be saying."*, and at different phases, P8: *"We do post filter and prefilter on the input content and the output content for responsible AI"*. These experts also establish the necessary metrics, measurements, and tool support required for evaluating Responsible AI, which all teams adhere to, As noted by P3, *"So there's a RAI responsible AI stack that catches if it's anything harmful."* Now that they have established and standardized these practices, they are focusing on automation and aiming to increase the frequency of RAI activities, P4: *"We're trying to do that more frequently. So we're looking at how do we run RAI just weekly and there's some new tools that many other teams are building to make that happen automatically."* The process and the automation are complicated as P17 described, and so the experts are helping the feature teams go through and use them effectively, *"There are many redundant tools all running and trying to catch inappropriate content at different layers, so helping teams integrate that into their system. The other side would be like understanding the requirements and consulting I guess with feature crews, [...] making sure we understand the prompts, the metaprompts, the mitigations that are in place"*

Emerging solution 18: Applying RAI red teaming strategies (48% adoption).

Red teaming, in general, is a technique where experts mimic cyber-attacks to identify vulnerabilities in a company's security system. Responsible AI (RAI) red teaming is focused on detecting and rectify any flaws related to Responsible AI within AI systems, to ensure responsible AI practices are enforced. P1 elaborates on how Microsoft has extensively enhanced its RAI red teaming process, transforming it from random tests to a more sophisticated system, *"red teaming used to be very random, right? Just to try ourself and give it a few prompt and see what triggers*

something offensive. Now it is more sophisticated [...] sometimes we also invite people from the bigger org to participate [...] so we can get more diverse inputs and observations." P17 builds on this by detailing the layered structure of this technique in the company: *"There's a few different layers that does red teaming, so at the [upper team] side, there are some security red teaming that's done and they've also dipped their toes in some like, RAI harms red teaming as well. At that, the high level and the lower down on the [team], there is an [product] specific we call it kind of a purple team: but it is the red team that also have internal knowledge of the system."* P22 also shared about an existing effort that combines offline evaluation with red teaming within one tool and a single codebase to simplify the process: *"So actually one of the things we've done recently is we moved all of our red teaming from [local repo] into our offline evaluation tooling. So it's all the same code base, making it easier for when you have offline eval you get red team, because both are generally needed to get to any kind of release."*

⚙ **Emerging solution 19: Following a robust RAI audit process (mandatory for adoption).** At Microsoft, it is mandatory for all teams to gain approval from an organizational entity known as the Deployment Safety Board (DSB) prior to releasing any feature related to LLMs. The DSB's primary role is to reaffirm that all teams have adhered to standard Responsible AI checks and practices. As P12 elaborated: *"Yeah, we run DSB, Privacy, we do all sort of review and validation before we release. So that's also why the release cycle has to be impacted. It just a lot of testing, a lot of evaluation and signoff has to happen before we release each update or each feature. But yeah, we don't want customers to experience harmful content."* In addition to initial approval, teams are also required to re-submit to the DSB for each significant change such as model migration. This constitutes a comprehensive process as P21 explained, *"Every time we change the model we're using [...] have to rerun all that offline evaluation and then do red teaming and get DSB approval and then we can do A/B and to see how that looks."*

4.1.4 Discussion

This study provides insights into how practitioners are addressing the disruptions they face when integrating LLMs into software products. By identifying 19 emerging solutions, we have highlighted strategies that are already being widely adopted

to manage the complexities of LLMs, particularly in the areas of quality assurance. These practices—such as defining custom evaluation metrics, combining qualitative and quantitative measures, and automating offline evaluations—offer practical solutions for handling LLM-specific disruptions.

Our findings have important practical implications. For development teams, adopting practices such as automating offline evaluations, involving data scientists in test creation, and using LLMs as validators can improve the development and evaluation of LLM-based features. These solutions not only help teams manage the unpredictable and subjective nature of LLMs but also provide a roadmap for integrating these new processes into existing workflows.

Beyond influencing engineering practices, the study also had organizational impact within the participating company. Our findings revealed significant fragmentation across teams developing LLM-based features and AI copilots, where different groups were independently experimenting with evaluation approaches and tooling with limited coordination. In follow-up discussions with leadership several months after the study, the evidence of these challenges helped highlight the need for a more coordinated effort. As noted by a collaborator involved in the study, "*one of the impacts of your work was the creation of a new team [...] focused on helping with many of the challenges that you found in your research.*" While organizational changes are rarely attributable to a single study, this example illustrates how systematically documenting practitioner challenges can surface structural issues and motivate institutional responses.

While the widespread adoption of many of these solutions by teams at Microsoft, and the frequent explicit endorsement in the survey, implicitly suggests effectiveness, we have not conducted a rigorous evaluation of how well they work in different contexts or compared them to potential alternatives. A logical next step in this line of research (for us or others) is to carry out such a study to assess the effectiveness of these solutions in various settings. As LLMs become more prevalent, development teams will continue to face these, and likely even more, disruptions. Our aim is to provide a set of validated best practices that teams can confidently follow to ensure that these features will behave as intended, without going off track or causing unintended outcomes.

We encourage other researchers and practitioners to share their experiences and insights as they adopt solutions, so that the community of software professionals

collectively can learn how to safely and responsibly develop LLM-enabled software products.

4.2 Intervention B: Guiding to Satisfy Explainable AI (XAI) Requirements

> This section shares materials with FAccT'24 paper: *"Regulating Explainability in Machine Learning Applications – Observations from a Policy Design Experiment"* [413], and arXiv'26 paper: "*Policy Alone is Probably not the Solution: A Large-scale Experiment on How Developers Struggle to Design Meaningful End-user Explanations"* [451].

Explainable AI (XAI), a component of responsible AI, has received comparatively less attention in both industry and academia than other widely discussed aspects such as fairness and privacy. Nevertheless, explainability and transparency are increasingly emphasized in regulatory and policy discussions [456, 68], highlighting their growing importance, particularly in regulated domains [55, 11]. Unlike fairness and privacy, which have relatively stable interpretations across technical domains, explainability in machine learning takes on meanings that differ substantially from its traditional interpretation in software systems, and its interpretation can also vary across disciplines such as sociology, and law. In software engineering, explainability typically refers to the clarity with which the software code and its architecture can be understood and reasoned about. While many software systems can be complex and difficult to interpret in practice (e.g., distributed systems or large-scale codebases), their behavior can fundamentally be inspected, traced, and probed through tools such as debugging, logging, and step-by-step execution. Debugging tools and techniques like breakpoint setting, step-by-step execution, and variable state examination lend further support to maintaining this form of inspectability. In contrast, ML explainability refers to deciphering the "opaque box" nature of complex algorithms, where the internal decision-making processes are learned from data and not explicitly written in code. Even with access to the model, understanding why a particular decision was made is often indirect and approximate rather than directly traceable. Thus, it does not simply mean tracing outputs back

See §3.1 (p.28) for findings on regulatory constraints and requirements

to inputs but understanding the layers of abstraction and approximation that occur within the algorithm.

This shift leaves considerable room for interpretation regarding what constitutes a meaningful explanation. Such ambiguity can arise both among technical practitioners and between technical and non-technical stakeholders who consume or rely on these explanations. Consequently, collaboration around explanations often occurs across knowledge boundaries [58]. At the 🔤 *syntactic level*, stakeholders may lack a shared vocabulary for describing what explanations in ML systems should contain. At the 💬 *semantic level*, stakeholders may interpret explanations differently; for example, whether explanations should focus on model internals, highlight influential features, or communicate implications for end users. At the 🤝 *pragmatic level*, stakeholders may also have conflicting incentives; for example, product teams may prioritize usability and rapid deployment, while regulators, domain experts, or affected users may demand deeper transparency or accountability. These differences complicate collaboration when designing and communicating explanations for ML-based products.

To mitigate this challenge, we begin by developing a series of interventions aimed at bridging these 🔤 *syntactic* and 💬 *semantic boundaries*. Rather than relying solely on broad educational efforts, we frame these as *microinterventions*—lightweight, targeted supports that can be integrated into practitioners' workflows to assist in generating meaningful explanations. First, we provide explicit guidance for ML practitioners through an explainable AI policy, positioned as a structured prompt to make key considerations more explicit during explanation generation, designed via an experimental study involving an interdisciplinary team of ML and policy researchers. We then assess this policy through a large-scale controlled experiment in an educational setting. The results show limited impact in the classroom. While the policy helps mitigate 🔤 *syntactic boundaries*, it falls short in addressing 💬 *semantic boundaries*, as participants struggle to determine what explanations would be appropriate from the perspective of end users. Motivated by this limitation, we introduce additional microinterventions that provide more contextual and interactive support. Based on the finding that policy alone is insufficient, we introduce a second intervention using LLM-based chatbots to role-play end users receiving model-generated explanations. These role-playing mechanisms act as contextual scaffolds, helping practitioners translate abstract

guidance into concrete, user-centered explanations. Although this approach shows moderate improvements, we further explore complementary supports, including structured guidance and persona development, to strengthen practitioners' ability to reason from end-user perspectives. This supports the development of empathy and helps practitioners better understand end-user perspectives. Together, these findings address the following research question: **How can different micro-interventions—such as structured guidance and LLM-based role-playing—support practitioners in generating explanations that are meaningful to end users?**

4.2.1 Related Work

Machine learning components (from traditional ML to LLMs to agents) are increasingly integrated into software products, where they produce outputs, suggest decisions, or even automate actions in the real world [8, 141, 268]; this includes medical devices and diagnostic tools promising lower costs and improved health outcomes [452]. Modern ML models are usually complex and inscrutable, even to their creators, where developers cannot simply inspect model internals to understand how exactly the model works. Software engineers who want to ensure the quality of the overall software product (including the safety of a medical device) hence need to understand how to integrate ML components and how to compensate for their inaccuracies, possibly through safeguards around the model [141, 268, 453, 454] and human-computer-interactions design [6, 262, 455].

4.2.1.1 Explainability

Explainability and *interpretability* usually refer to specific tools that extract insights from otherwise inscrutable models [224], 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 [33, 132], but there is also extensive research about how to make explanations useful to non-experts under the label of *human-centered explainable AI* [299], 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 [224]. 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) [195, 33, 224], 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 [224, 300]. End users often ask for descriptions of the data used by the product and fear that they would not understand more specific explanations [196]. Research has shown that study participants often misinterpret or place too much trust in explanations [343, 466, 365], raising concerns that explanations could be used to manipulate users.

4.2.1.2 Human-centered explainable AI and the purpose of explanations

Explanations are usually intended to serve a purpose, whether functional, social, economic, or normative [68, 234, 456, 299, 456, 457], but that purpose is rarely articulated clearly in discussions, requirements, or even regulation. Beyond debugging, purposes include (1) *auditing*, especially for fairness issues [456, 458], (2) *human-AI collaboration* for effective use and calibrating trust [55, 262, 460], (3) *oversight* and *contestation* of wrong data and decisions [456, 459], and (4) assuring the *dignity* and agency of individuals [68, 456]. For many of these purposes, explanations must be aimed at end users or external parties, not just developers.

Critiques of a lack of end-user focus go back to the early days of explainability research [461] and lead to the emergence of the *human-centered explainable AI* community [55, 262, 365, 462, 463], which explores designing effective explanations for *end users*, e.g., to improve human-AI collaboration. However, end-user explanations are generally less studied and less deployed than technical explanations for developers, and evidence for effectiveness is mixed [463]. Many studies highlight risks for manipulation of user behavior through explanations, e.g., [464–

466], and recognize that explainability needs for end users are context-dependent beyond one-size-fits-all solutions [467].

4.2.1.3 Explainability and policy

Regulation provides a form of societal infrastructure for coordinating social welfare and distributing risks, and establishing paths toward standards of practice [468]. Sociological scholarship often makes a distinction between two types of relevant law in medical contexts [469]: "Hard law" is typically passed by legislative bodies and enforceable by action of regulatory agencies or in court, backed by the authority of the government; the EU AI Act is one relevant example [470]. "Soft law," by contrast, includes rules and guidelines that are written by a range of non-legislative bodies, as well as the guidance to implement and interpret hard law (still emerging in the case of the EU AI Act). Policy can have effects at two levels: (1) at a legalistic level, motivating changes to behavior to avoid a pre-specified sanction, like fines, and (2) at a normative level, indicating what is symbolically valued or desired and setting expectations for what constitutes good professional behavior. Even when it is not enforced or enforceable, policy can signal values and provide a basis for professionals with less power to challenge or shift the status quo [469]. Against a recent turn toward de-regulation for AI-powered in the U.S., it is all the more important to assess what is possible for "softer" policy guidance to achieve, particularly in an early phase where developers move fast with emerging technologies before norms emerge about responsible engineering practices. To date, policymakers have sketched out broad policy principles aimed at shaping developer behavior on transparency for ML systems, such as White House *Blueprint for an AI Bill of Rights* [471] and former Executive Order 14110 [472]. Additionally, many companies have/are investing in in-house responsible AI internal guidelines and practices [217, 473].

However, law in any form has important limitations. Evidence from social science suggests that organizations, institutions, and professional norms, in addition to law, play roles in changing the actions of professionals like developers [474–476]. At its most effective, policy provides clear guardrails that enable innovation [477] by balancing between competing demands: It must ensure an even playing field for technological development and commercial exchange while not creating so onerous

a burden that innovation is stifled [478]. In practice, in high-risk domains (e.g., healthcare, aviation), regulation and policy are heavyweight and sometimes cumbersome, entailing substantial infrastructure, guidance, and consultants [479, 480]. By contrast, many AI guidelines aim to cover applications across many domains, taking a more lightweight approach. It remains a question what developers (without dedicated training or access to experts) make of this kind of policy guidance, and what other support might be necessary to ensure compliance, given context-specific and application-specific explanation needs [467] and competing demands on developers' attention, like time pressure, conflicts of interest, and regulatory capture [481–485]. Research on how developers receive, interpret, and enact guidelines–how, in short, they navigate between the legalistic and normative levels of law–is necessary to help better align policy and developers.

4.2.2 Research Design

We began with a 10-week collaborative policy design study with an interdisciplinary team of ML and policy researchers to develop an explainable AI policy. We then evaluated this policy through a series of controlled experiments conducted across four semesters in a graduate course. In the first two semesters, we examined how developers design end-user explanations and how different policy language and enforcement mechanisms influence compliance and explanation quality. In the following two semesters, we extended the intervention by introducing persona-based role-playing chatbots, followed by a self-persona development task and educational support. Across all four semesters, the experiments involved 409 participants.

4.2.2.1 Collaborative and Iterative Policy Design

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—

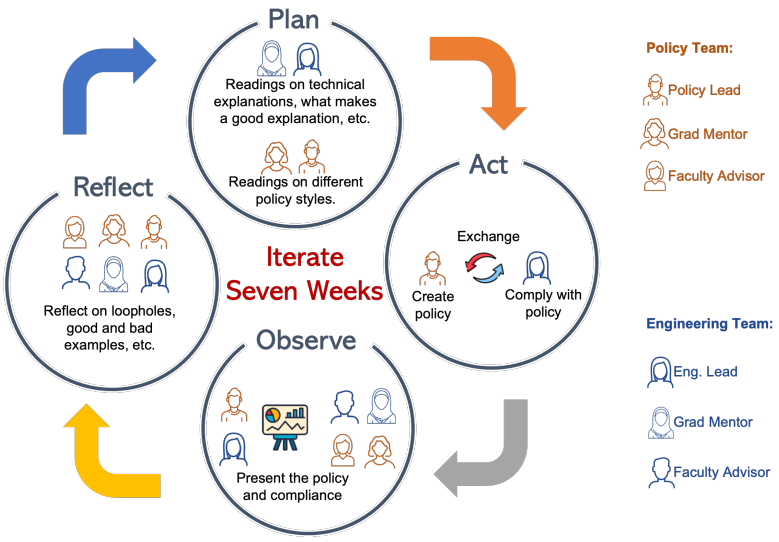


Fig. 4.3 Iterative and Collaborative Policy Design Process

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 [346, 302], 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. 4.3). 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 4.3).

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 arrived at several reasonable policy drafts for different contexts and purposes (more details can be found in the policy design paper [413]), among which we used the policy in Figure 4.4 for our experiments.

4.2.2.2 Experimental Design in an Educational Setting

Having developed the policy, we next examine how developers design end-user explanations when supported by different forms of guidance: the policy itself, LLM assistance, and educational scaffolding. To study this systematically, we conducted a controlled experiment in a classroom setting, which allowed us to collect a large number of responses while maintaining control over the task and experimental conditions. In the following sections, we describe the study scenario, the task given to participants, recruitment, and the data analysis procedure.

Purpose of Policy: To preserve the dignity of individuals | To enable effective human-AI collaboration | None
Policy Requirements: Designers, developers, and deployers of automated systems should provide generally accessible plain language documentation including clear descriptions of the overall system functioning and the role automation plays ①, notice that such systems are in use, the individual or organization responsible for the system ②, and explanations of outcomes that are clear, timely, and accessible ③.
Specifically: [comprehensive policy version only]
INTENDED USE

- Describe the automated system's intended use and the role of the automation (model) ①.
- Provide evidence that the automation (model) functions accurately, consistently, and effectively in the intended use case ④.

HOW IT WORKS

- Describe how the automation (model) works generally. Provide evidence that the documentation is effective for the policy purpose.
- Provide a mechanism to describe how the automation (model) worked with regard to an instance of use to all intended users and subjects affected by the automated system in a form that is accessible to them ③. Descriptions must include (1) that automation was used, (2) a short explanation of how the automation works, (3) what additional actors are involved in decisions, (4) what significant personal data was used for the decision ⑤, (5) what decisions were reached in a specific case. Provide evidence that the documentation is effective for the policy purpose.

CONCERNS

- Describe limitations and misuse potential ⑥ of the automated system beyond its intended purpose and any provided mitigations ⑦.
- Describe the data used by the automated system. Justify the use of personal identifiable information.
- Describe how to report misuse ⑧ or harm from the automated system.

LANGUAGE REQUIREMENTS

- Provide all documentation in language appropriate for the intended audience. All documentation for untrained users must use nontechnical language at an eighth grade reading level ⑨.

Fig. 4.4 Our policy, highlighting the policy requirements selected for analysis (①–⑨)

Observation 1: Over the course of seven weeks of iterations, it was possible to draft policies that addressed the concerns of involved partie s 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 worldviews 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 4.3 Summary of the Observations on Policy Design Exercise

Scenario: Diabetic retinopathy screening. Participants were asked to provide explanations for a hypothetical low-cost ML-powered medical device to screen for diabetic retinopathy. The device detects diabetic retinopathy on a scale of 0 to 4 (none to severe) using images of the eye and the patient's age and gender, comparable to existing commercial screening tools. The device would be used by trained users (e.g., nurses or volunteers) to perform screenings at mobile clinics or in patients' homes, with the potential, as stated in the scenario, to *"drastically reduce*

screening costs and make screenings much more available, especially in under-resourced regions of the world." Related (more costly) devices are commercially available [239, 238, 118]; in Fig. 1.1, we show the limited explanations for/by one of them. Existing rates of compliance with annual screening recommendations for diabetic retinopathy among diabetics in the U.S. range from 25 to 60% [334].

We chose this scenario for its real-world application, current relevance, and readily available data and models. In preparation, we conducted interviews with regulators of medical devices, medical professionals, and diabetes patients, asking how they approached understanding screening device predictions, complying with clinical norms and regulations, and integrating tools into clinical practice. Over two years, we attended large diabetes conferences, where we interacted with representatives of companies (both startups and established firms) marketing ML-powered diabetic retinopathy screening devices and observed how screening tools were introduced to physicians. This preparation provided us with more background knowledge than most the average non-clinician researcher to evaluate participant solutions from the perspective of clinical practitioners and patients.

Assessment tasks. We provided a dataset (from a public dataset used for a Kaggle competition [10]) and a pre-trained ResNet50 model. We augmented the data with synthetically generated gender and age data to enable participants to perform segmented analysis of subpopulations and describe the use of potentially sensitive information.

The task was to create explanations for the system that comply with a provided policy (see below), creating (HTML) pages that present: (a) *Global explanations:* What external stakeholders might want to know about the product, the model, or the data. This might be information found on the product web page, training materials, or a handbook. (b) *Individual explanations:* Information about a specific diagnosis. This might be shown on the device, recorded in the patient's medical records, or provided as a printed handout.

In the first experiment, we asked participants to identify targeted stakeholders themselves; in others, we specified that the handbook was intended for nurses/volunteers and the handout for patients. In addition, we asked participants (a) to describe their solution, (b) to self-assess their compliance with their assigned policy

and provide evidence of their compliance, and (c) to write a reflection about the challenges they faced.

Participants were given basic training in explainability techniques and transparency as part of their coursework prior to completing the task (160 minutes of lectures, two readings [300] [262], and an 80 minute lab session); instructions briefly covered the pitfalls of explanations and the diverse needs of different stakeholders (using the "Hello AI" case study [55]), but mostly focused on technical post-hoc explainability techniques like *LIME* and *Anchors* [224]. Participants were not given instruction about diabetic retinopathy or clinical communication. We designed the task to be about 8 hours of work per participant, not including prior training.

Recruitment and participants. To assess the interventions (policy and other), participants were recruited from a large graduate course on software engineering, machine learning, and MLOps in four consecutive semesters. In the course, most students already had substantial prior experience as software engineers or data scientists: according to the first two semesters, 63% had prior internship, research, or work experience as a data scientist, and 51% had internship, research, or work experience as a software engineer, including 29% of students who had previously worked in industry as a data scientist or software engineer (or both). Only 6% and 5% of students indicated having no prior data science or software engineering experience respectively. The students' background is reflective of many early-career practitioners in industry teams, who usually have experience in their field and basic awareness of explainability tools, but limited exposure to human-centered explainable AI. While they likely have personal experience with medical devices as patients, our participants were unlikely to have the domain expertise or the access to domain experts that would come with working in an industry team on a commercial product.

The IRB approved study was designed as a secondary analysis of a homework assignment. All students in the course had to complete the homework assignment and were graded based on a standard rubric. In the first semester, the rubric did not require policy compliance and was orthogonal to the six experimental conditions. In the second semester, all students were given the same policy and graded uniformly on compliance. In the third semester, students were assigned to one of two groups,

Study	Policy variation	LLM intervention	*n*
Experiment 1: Semester 1 *Not enforced; participants self-selected stakeholder*	No purpose specified; length: short	-	17
	No purpose specified; length: comprehensive	-	20
	Human–AI collaboration; length: short	-	24
	Human–AI collaboration; length: comprehensive	-	17
	Preserving dignity; length: short	-	26
	Preserving dignity; length: comprehensive	-	20
Experiment 1: Semester 2 *Enforced; stakeholder provided*	Policy conditions combined	-	70
Experiment 2: Semester 3 *Chatbots provided and assigned*	Policy conditions combined	Role-play	61
	Policy conditions combined	Writing-coach	57
Experiment 2: Semester 4 *Guided for persona development*	No policy	self-created	97

Table 4.4 Experimental Conditions and Participant Counts (n)

with access to an LLM either as a role-playing chatbot or as a writing coach. In the fourth semester, all students were tasked with developing personas themselves and were guided on common pitfalls in producing explanations. All participant assignments were random (see Table 4.4). Students could opt to allow us researchers to perform an analysis of the anonymized assignment after the submission of final grades at the end of the semester.

While we know the demographics of students in the course generally, we intentionally did not collect individual background information of participants due to research ethics considerations and to avoid raising barriers to participation. Random assignment of large experimental groups makes substantial experience/demographic differences among the groups unlikely. Demographics and experience were similar across all semesters.

Data analysis. We analyzed all solutions using *qualitative content analysis* [313], where researchers create coding rubrics for one or more dimensions and systematically assign one code per dimension to each chunk of analysis (here each participant's solution is considered as one chunk). Qualitative content analysis uses

qualitative research methods for interpreting meanings, themes, and patterns within content through inductive reasoning and contextual understanding for systematic coding that *produces frequency counts that can be analyzed quantitatively*.

We analyzed the solutions from multiple angles: First, we identified elements of explanations in terms of what form the explanations have (e.g., text, visuals), what data is presented (e.g., confusion matrix), and what post-hoc explanation tooling was used (e.g., SHAP). Then, we judged policy compliance of each solution for nine specific policy requirements highlighted in Table 4.4. We purposefully selected a subset of policy requirements to scope the analysis, including requirements related to global (e.g., ①, ②, ⑥) and individual (e.g., ⑨, ⑤) explanations, requirements that require deep design (e.g., ④) and requirements that are met with fact statements (e.g., ②, ⑧). We assessed compliance with the requirement to write explanations at an 8th-grade reading level automatically through the common/standard/validated measure of *Flesch-Kincaid (FK) Grade Level* [388, 60, 350]. Finally, independent of compliance, we coded for four common failure modes and corresponding symptoms that resulted in poor quality explanations discussed in detail below. For the first iteration where we left the choice of stakeholder to the participants, we only analyzed those solutions that targeted nurses for global explanations (n) and patients for individual explanations (n) to enable more meaningful comparisons.

As standard for this method [313], the codebook was developed based on domain knowledge and an analysis of a subset of the solutions, before applying it to all solutions. Especially for failure modes, we first analyzed common problems in the solutions in an open-ended way (mirroring thematic analysis [178, 246]), settling on the coding frame only after many discussions, once we reached saturation. We share the codebook in the appendix of the original paper [409]. To increase reliability, after our initial manual coding, we repeated the coding with an LLM, and investigated every disagreement between the model and the original labeler (6% to 31% of labels per dimension), and corrected 92 labels out of 868. We found LLMs to be unreliable and we instead assessed inter-rater reliability on 40 solutions (20 individual, 20 global explanations) with two raters, yielding Cohen's k values between 0.83 and 1, indicating almost perfect agreement. For the resulting quantitative data, we report descriptive statistics and refer the interested reader to the appendix for (often negative) statistical results from chi-squared tests and logistic regressions.

4.2.2.3 Limitations and threats to validity

As with every study, ours also has limitations from tradeoff decisions in the research design, and the results should be interpreted accordingly. First, we are an interdisciplinary research team from four US-based universities with distributed expertise in machine learning, software engineering, and social science. We have interacted with and interviewed manufacturers and users of diabetic retinopathy screening tools (see above), giving us more domain knowledge than the participants. Despite carefully calibrated rubrics and assessed inter-coder reliability, our backgrounds may bias us towards assessing explanations more critically than the average population of users. Second, conducting a study with graduate students has recognized benefits and drawbacks [306, 97, 99]. The classroom setting allowed us to conduct the study at a scale (number of participants and task length/depths) that would be infeasible with professional developers. With a majority of our participants having prior internship, research, or work experience, we regard them as representative of early-career professionals about to (re-)enter technology careers upon graduation. As their education is more recent and they were introduced to explainable AI through course content, participants might be more primed for responsible AI engineering than most practitioners. Participants may be biased to use techniques explicitly introduced in the course, but this is orthogonal to our findings. In contrast, the typical practitioner would likely have more domain knowledge about healthcare. Readers should exercise care when generalizing results beyond our population.

4.2.3 Experiment I: Policy and Enforcement Variations

We analyze the first two semesters separately from the latter two because the interventions differ. In the first two semesters, we experimented with variations in policy language, length and enforcement, whereas in the later semesters we introduced persona-based interventions. In this section, we report results from the first two semesters. We begin with general observations across all policy conditions before examining differences among the experimental groups.

In the first semester, we provided policy as guidance and required self-assessment but varied the *comprehensiveness* of the policy and its provided purpose (6 conditions): We either provided a one-sentence policy extracted from the *Blueprint for an AI Bill of Rights* [36] or a more comprehensive version that *additionally* included a

prescriptive list of requirements, inspired by recent research on policy design [234]; for each, we provided one of three stated *purposes* of the policy as either (a) *"to enable effective human-AI collaboration,"* (b) *"to preserve the dignity of individuals,"* or (c) no purpose was stated. In Table 4.4, we show the text of all policy versions. After learning that policy differences had little influence in our first semester, in the second semester we assigned the same policy (the comprehensive policy, without the initial sentence, with the purpose of effective human-AI collaboration for nurses and preserving dignity for patients) to all participants, but enforcing the policy through a grading rubric rather than asking for self-assessment only.

4.2.3.1 Participants provide mostly technical explanations with off-the-shelf tools

While the policy provided high-level guidance or requirements, the assignment did not prescribe *how* to provide explanations. See Fig. 1.1 for illustrative excerpts of some solutions. In semester 1, the majority of participants provided technical information about model evaluation and training for global explanations (e.g., SP16 in Fig. 1.1), with over half reporting cohen kappa scores, confusion matrices, and description of training data distributions. Many participants (21%) provided disaggregated evaluation results for subpopulations by age, gender, or severity. A few participants provided technical details of the model architecture (e.g., SP19 in Fig. 1.1). About half of the solutions provide a description of the purpose of the model in the system. For individual explanations, almost all solutions (98%) included a visual explanation highlighting pixels or overlaying boxes on the input image (63% used anchors, 19% LIME, 8% SHAP, 10% others; as in Fig. 1.1), however often without any description on how to interpret the image. Generally, participants used explainability techniques that are readily available from libraries. Explanations in the second semester were similar, with participants again relying primarily on readily available explainability techniques and providing visualizations with limited interpretation.

4.2.3.2 Policy language barely influenced compliance, but enforcement did

We show compliance results across experimental conditions in Table 4.5. Contrary to our initial expectations, the specific policy language (comprehensiveness and

Policy requirement	% Compliance
① System description & model role	70.6% 65.4% 50.0% 70.0% 65.0% 70.6% 95.7%
② Organization responsible	0.0% 3.8% 4.2% 0.0% 0.0% 0.0% 4.3%
③ Clear and accessible for end users	0.0% 4.3% 5.0% 5.0% 5.6% 0.0% 1.4%
④ Model accuracy	76.5% 80.8% 79.2% 90.0% 85.0% 82.4% 97.1%
⑤ Personal data used in prediction	5.9% 0.0% 0.0% 5.0% 0.0% 11.8% 20.0%
⑥ Limitations and misuse	41.2% 30.8% 16.7% 50.0% 50.0% 47.1% 87.1%
⑦ Mitigations of limitations	0.0% 3.8% 12.5% 15.0% 35.0% 29.4% 71.4%
⑧ Reporting of misuse and harm	0.0% 3.8% 8.3% 20.0% 25.0% 29.4% 47.1%
⑨ Eight grade reading level	0.0% 0.0% 0.0% 0.0% 0.0% 0.0% 0.0%

Compliance in all six experimental conditions in experiment 1, from top to bottom: No purpose/short, dignity/short, human-AI col./short, no purpose/comprehensive, dignity/comprehensive, human-AI col./comprehensive. The vertical line indicates the average across all conditions. The blue line is for compliance in experiment 2.
①–③ are included in the short policy; ④–⑧ are included only in the comprehensive policy, ①, ③–⑨ are enforced through grading rubrics in experiment 2.

Table 4.5 Compliance with Selected Policy Requirements

purpose) had little influence on compliance in semester 1, where participants were asked to comply with the policy but compliance was not enforced through the grading rubric. A few results are instructive though: Participants across all policy conditions were likely to share model accuracy (④), even though it was only required in the comprehensive policy. For other requirements only stated in the comprehensive policy, such as identifying model limitations (⑥), we saw slightly higher compliance when the requirement was actually stated, but only marginally so. In contrast, participants rarely identified the responsible organization to contact in case of harm (②) even though this was stated in the first sentence of the policy. This suggests that participants largely wrote what they understand and what is intuitive to them, often ignoring (or failing to comply with) other parts of the policy. Regarding policy purpose, we did not detect any differences, quantitatively nor qualitatively. This lack of differences across policy comprehensiveness and purpose is why we did not vary policy language in the second semester.

In semester 2, we tightened policy enforcement with a stricter grading rubric. We found that observed compliance increased for almost every policy requirement (statistically significant except ②,[1] ③, and ⑨; see appendix [409]), such as including limitations of use (⑥), and reporting of misuse(⑧). Still, compliance was still fairly low for several requirements that did not align easily with technical explainability tooling, such as reporting personal data used (⑤, 20%), and reporting path for misuse and harm (⑧, 47.1%). Low compliance, even despite enforcement, suggests that participants did not understand the requirements or were not able to comply. For example, participants sometimes encouraged users to report issues, but without providing a concrete reporting process or contact point. This explains also the lack of difference in complying with plain language requirements③ and ⑨), as almost all participants failed at this; only 4 individual explanations passed our assessment of 8th grade reading level, and not a single solution passed this for both global and individual explanation.

Analyzing participants' (often cursory) self-assessments of compliance in the first semester, we found that participants often claimed that they complied even though they quite obviously did not. A notable only exception was that many partic-

[1] The "organization responsible" requirement ② was not included in the second semester. The lack of change here supports the finding that the other changes are due to the treatment effect (enforcement).

ipants acknowledged that they did not know how to comply with writing accessible explanations (⑨). In reflections, almost every participant described difficulty writing clear and accessible explanations [234]. Some participants described this task as potentially insurmountable, like SP12: "*The requirement to use plain language can be at odds with the complexity inherent in automated systems, particularly in AI and machine learning models.*" The necessity and trickiness of balancing was a common theme, and some participants thought they had done acceptably given resource constraints. For example, SP113 argued, *"Fully complying with the policy can also take up a lot of extra time and cause stress. Engineers should be spending more time working on actual systems than writing up documentation [. . .] perfect English and documentation skills aren't typically required of software experts*." Ultimately, some participants recognized that they were falling short in the requirement to write clearly but were unable to come up with a good solution.

4.2.3.3 Explanations were not meaningful for their intended end users

Failure mode	Individual explanations	Global explanations
1: Inscrutable	7.0% 7.1%	12.0% 12.9%
2: Requires ML expertise	87.7% 85.7%	93.0% 84.3%
3: Requires medical expertise	17.5% 27.1%	*Not applicable*
4: Model-centric	91.2% 78.6%	70.0% 24.3%

For each plot, the top bars correspond to semester 1 and the bottom bars to semester 2.

Table 4.6 Failure Mode Comparison by Explanation Type

No explanation was fully compliant with policy; in particular, participants failed the requirement of a clear and accessible explanation. Even when participants complied, we often noticed shallow solutions that technically complied with the language of the policy, but did little to further the policy's purpose in our judgment, suggesting a check-the-box approach to compliance rather than careful engagement. We judged almost all explanations as likely incomprehensible to the intended users, and thus as ineffective. To better understand the problems beyond compliance, we thus analyzed common problems in an open-ended fashion (cf. Sec. 3), resulting

in four failure modes we discuss here. In Table 4.6, we report the failure modes for global and individual explanations across both semesters (as policy conditions in the first semester made no meaningful difference, we group them together here; details in the appendix [409]).

Failure mode 1: Inscrutable or technically incorrect explanations. Some explanations failed at basic intelligibility such that even experts cannot reasonably interpret what is shown. Several of these cases reflected misunderstandings of how explainability tools should be applied or interpreted. Common errors included presenting raw or opaque artifacts (e.g., arrays of pixel values, unlabeled SHAP outputs, or internal CNN activations), omitting essential context (e.g., feature names, scales, or reference points), or producing technically incorrect explanations due to misuse of explainability libraries (see appendix [409]). For example, SP2 printed the numerical SHAP values of the pixels of only the top row of the image as an array of numbers (see appendix [409]), offering no visual or semantic grounding and FP7 presented a SHAP waterfall plot attributing a prediction to individual image embedding dimensions (e.g., *image_embedding_973*). This failure mode was relatively uncommon (7% of individual explanations and 12% of global explanations), and stricter enforcement in the second iteration did not meaningfully change these rates.

Failure mode 2: Understanding explanations requires ML expertise. We judged 88% of all solutions as likely inscrutable to end users without ML expertise. Participants frequently presented explanations in a disjointed, fragmented, check-the-box manner, lacking a coherent form aligned with the designated stakeholders. Very commonly, explanations resembled internal documentation intended for technically competent peers (e.g., machine learning experts). For instance, in almost half of the individual explanations, participants provided images that highlight areas without describing the significance of those areas (see Fig. 1.1). Furthermore, explanations commonly included jargon-heavy language, such as *kappa*, *confusion matrix*, or *train/test data* instead of domain-appropriate medical language such as sensitivity, specificity, or efficacy or plain language descriptions for lay users. These explanations make sense to the developer, but are difficult to follow for anyone not immersed in the same exercise or knowledge base. Even when participants sometimes attempted to translate technical concepts into plain language, they pro-

vided lengthy descriptions of *surrogate models* (FP14) or the concept of *feature importance* (FP7), that are likely not relevant to patients' information needs. Even though plain and accessible language was required by the policy, even stronger enforcement did not improve these problems.

Failure mode 3: Understanding explanations requires medical expertise. In 22% of the individual explanations intended for patients, the explanation relies on medical terminology (e.g., *neovascularization*, *microaneurysms*, or the *peripapillary region*) without additional context that is unlikely to be intelligible to patients without medical training. This was again largely unaffected by enforcement (not statistically significant; see appendix [409]). Since global explanations were intended for nurses, we accepted such language there.

Failure mode 4: Explanations failed to consider the larger context and purpose of the AI system. Many participants (66%) failed to embed explanations in the context of a larger system or use where it is used as part of a workflow. For example, some global explanations reported model accuracy by subpopulation, but did not highlight those subpopulations as ones that should be approached with care in the text for healthcare professionals. Only a few explanations for patients included information about "what does this mean for me" or "what are next steps." These solutions did not consider explainability as one contribution to a larger sociotechnical system aimed at reducing patients' risk of blindness. Embedding explanations in system context and purpose makes the tool more useful, and is especially critical in healthcare settings [373, 179]. Notably, tightening policy enforcement was associated with an improvement regarding this failure mode, especially in global explanations (the only statistically significant result in failure modes analysis). Here, even check-the-box compliance required some engagement with harms, mitigations, and reporting, that go beyond a narrow focus on the model.

Explanations suitable for end users. While the vast majority of explanations were not plausibly targeted to patients or nurses, some participants did offer explanations that we thought were plausibly targeted to those end users. "Good" global explanations contained information presented in a clinically useful way (e.g., in terms of false positives and false negatives), showcasing the limitations and biases

of the model to spur humans to challenge the model's results when it would matter the most for patient outcomes. We did not necessarily expect training data information or technical model details to be included, which is required for regulatory U.S. Food and Drug Administration clearance of medical devices, but usually not included in practitioner handbooks.

Properly targeted individual explanations used clear and accessible language, employing visuals and describing what they showed. They clearly marked the predictive result and posed and answered the question of "what does this mean for me?" For instance, after listing the patient name/ID, gender, age, and diagnosis on separate lines, the FP01 offered the following summary text: *"Your eye scan shows proliferative diabetic retinopathy, a serious condition. This involves the growth of new, abnormal blood vessels in your retina, which can lead to severe vision impairment or blindness. Please seek urgent medical attention from an eye care specialist."* Generally though all solutions that were tailored to patients and avoided the failure modes above still included way more information than clinical professionals that we spoke to preferred – for example, explaining how to read the annotated image from an explainability tool, rather than omitting such visualization or merely providing reference images of diabetic retinopathy at different stages for the individual to compare to their own image. Norms of clinical communication [210], that include only the prediction, the personal data used (to comply with the policy requirement), what the patient should do next, and directing the patient to a number or organization if they had questions or were concerned about the accuracy of the result.

4.2.4 Experiment II: Role-playing Chatbots and Persona Development

We next analyze the latter two semesters separately because these experiments introduce persona-based interventions aimed at addressing the shortcomings observed in Experiment I: While the policy helped clarify expectations for explanations, many explanations remained overly technical and were not meaningful to end users. To better align explanations with end-user perspectives, we introduced interventions designed to foster empathy and perspective-taking.

Explanation Type	Condition	N	1:	2:	3:	4:
Global (Nurse Handbook)	Writing coach	61	1 (1.6%)	25 (41.0%)	9 (14.8%)	16 (26.2%)
	Roleplay nurse	57	0 (0.0%)	18 (31.6%)	3 (5.3%)	13 (22.8%)
Local (Patient Handout)	Writing coach	61	0 (0.0%)	36 (59.0%)	–	1 (1.6%)
	Roleplay patient	56	0 (0.0%)	37 (66.1%)	–	3 (5.4%)

1: Inscrutable, **2:** Requires ML expertise to understand, **3:** Requires medical expertise to understand, **4:** Model-centric explanation.

Table 4.7 Distribution of failure modes across chatbot conditions in semester 3 for global and local explanations. Values show counts with percentages in parentheses.

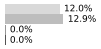
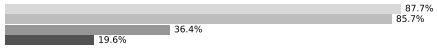
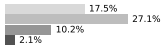
In the third semester, we introduced role-playing chatbots to simulate end-user personas. Participants were assigned to one of two conditions. In the first condition, the LLM acted as a writing coach informed by cognitive process theory [488, 489], prompting participants to reflect on how their explanations might be understood by their intended audience and encouraging revisions to improve clarity and accessibility. In the second condition, the LLM role-played end users (i.e., nurses and patients) interacting with the system, allowing participants to iteratively refine their explanations through simulated user interactions, a technique inspired by perspective-taking and experiential learning approaches used in design and education [490, 492]. Both groups had access to the same policy guidance and were tasked with generating explanations for the same ML system scenario.

In the fourth semester, we extended the intervention by explicitly drawing participants' attention to the perspective-taking challenges that arise when designing explanations for end users. Rather than prescribing a specific method, we provided educational guidance on several approaches for identifying user needs—such as developing personas [491], interviewing potential users, or conducting hazard analysis [493]—to help participants reason about explanations from the user's perspective. This condition was intended to elicit the "best-case" explanations participants could produce when given stronger educational scaffolding, allowing us to approximate what may be achievable in practice. Unlike earlier conditions, this setup did not include the policy intervention, which could otherwise act as a distraction.

4.2.4.1 The writing-coach and role-play chatbot conditions did not produce significant differences in explanation quality and showed limited engagement with participants, though both prompted revisions to obvious clarity issues in participants' explanations

In the third semester, we expected the *role-play chatbot* to produce stronger outcomes than the *writing-coach chatbot* because it simulated interactions with a stakeholder and was intended to prompt participants to reason more explicitly about audience needs and communication goals. However, we did not find evidence supporting this expectation. Across both global explanations (nurse handbooks) and individual explanations (patient handouts), the number of common mistakes did not differ significantly between conditions (as shown in Table 4.7). For global explanations, a one-way ANOVA showed no significant effect of condition, $F(1,119) = 0.19$, $p = .66$. Similarly, for individual explanations, we found no significant difference between the writing-coach and role-play conditions, $F(1,115) = 0.69$, $p = .41$.

Qualitative analysis of participant-chatbot interactions provides insight into these results. Neither chatbot appeared to generate substantial engagement; participants typically interacted with the tools at a surface level and rarely iterated extensively on their explanations. Despite this limited engagement, the overall quality of explanations still improved (as seen in Table 4.8). In both conditions, the chatbots successfully highlighted obvious issues in the explanations, which participants were able to correct. Common issues identified by the chatbots included: excessive technical jargon for the intended audience, overly detailed model descriptions, misalignment between the explanation and its intended goal (e.g., building trust), and structural clarity problems. Addressing these issues helped participants improve the structural flow, goal alignment, and appropriateness of language in their explanations. As a result, even relatively shallow engagement with the chatbot feedback appeared sufficient to guide participants toward clearer and more audience-appropriate explanations.

Failure mode	Explanation type	Failure percentage
1: Inscrutable	Patient handout	
	Nurse handbook	
2: Requires ML expertise	Patient handout	
	Nurse handbook	
3: Requires medical expertise	Patient handout	
	Nurse handbook	*Not applicable*
4: Model-centric	Patient handout	
	Nurse handbook	

For each plot, the top to bottom bars respectively correspond to semester 1, semester 2, semester 3, and semester 4.

Table 4.8 Failure Mode Comparison by Explanation Type

4.2.4.2 Educational guidance on understanding end-user needs (e.g., personas, interviews, hazard analysis) was associated with greater participant buy-in and more actionable explanations

Participants in the perspective-taking educational condition (semester 4) showed stronger engagement in refining their explanations. Across both global explanations (nurse handbooks) and local explanations (patient handouts), we observed a reduction in common mistakes after the persona step (Table 4.8). More importantly, participants moved beyond surface-level revisions and began reasoning more explicitly about the *information needs* of different stakeholders, tailoring their explanations to support how those stakeholders would interpret and act on the model's outputs.

The information needs were often identified through *persona analysis*, where participants considered the knowledge level, responsibilities, and decision-making context of nurses, screening operators, or patients. Participants also drew on clinical resources (e.g., WHO guidelines or medical literature), interviews with clinicians and patients, interactions with chatbots, and hazard analysis to identify potential failure cases and operational risks. Together, these activities—and the broader

process of explicitly reasoning about end-user information needs—appeared to shift participants' focus from describing the model itself toward identifying practical information related to system use, although we cannot determine which specific strategy contributed most to this effect.

For global explanations, participants increasingly reframed the document as operational guidance for practitioners. Rather than focusing solely on how the model works, many explanations introduced guidance on how to operate the screening system, including image quality requirements, image-capture checklists, screening workflows, and best practices for using the tool. Participants also expanded sections explaining how to interpret model outputs, incorporating reliability metrics such as sensitivity, specificity, or accuracy by severity level, along with confidence score interpretations, retinal feature breakdowns, and representative example cases. In addition, many explanations included safety and oversight guidance, such as escalation protocols, safety checks, referral guidelines, and instructions on when clinicians should trust or question the model's predictions. Together, these additions made the explanations resemble practical guidebooks that support clinicians in integrating the model into the screening workflow.

Improvements were also visible in the local explanations (patient handouts). Compared to earlier drafts, participants more often included information that helped patients interpret the screening results, such as simplified descriptions of disease severity and short summaries explaining what the screening outcome indicates. Majority of the explanations also provided clear guidance on next steps, including recommendations for follow-up care, when to seek medical attention, and practical considerations such as insurance coverage. In addition, participants increasingly anticipated common patient questions, incorporating FAQ-style sections and example phrasing for discussing results. These additions helped address common patient concerns about the screening process, the reliability of the results, and the implications of the outcome for their health. This made the explanations easier for patients to understand and act on.

4.2.5 Discussion

Given the minimal training and guidance provided in the first experiment, we did not expect participants to deliver high-quality explanations appropriate for patients

or nurses. However, we anticipated that the policy might have some influence on the explanations by providing symbolic guidance and nudging developers—who were new to these ideas—toward clearer explanations with a defined purpose and audience. We also explored whether stronger enforcement mechanisms might push participants toward at least minimal compliance with policy expectations. More broadly, we hoped that participants would recognize how challenging it is to produce explanations that are meaningful for end users.

Some participants were indeed able to produce explanations that were oriented toward end users, suggesting that such expectations are not inherently unrealistic. However, overall we found little evidence that policy guidance or increased enforcement substantively improved explanation quality. Most participants did not appear to recognize how misaligned their explanations were with the needs of end users. Instead, they primarily focused on simplifying technical descriptions, often reporting difficulty in translating technical language into "eighth-grade language" rather than reconsidering what information end users actually needed. Participants rarely discussed ambiguity in the policy itself or difficulties interpreting terms such as "dignity." Nor did they frequently acknowledge the challenge of anticipating the informational needs of patients or nurses. Although we expected that the stated purpose of the policy might guide participants in deciding what information to provide, it had no observable influence. In short, the symbolic and normative level of policy alone had limited impact.

When we increased enforcement by incorporating the policy into grading in the second semester, participants made only incremental changes to their explanations. While compliance increased, explanation quality remained relatively low. Participants occasionally addressed concerns beyond the model, for example, suggesting next steps for patients or providing instructions for nurses on how to capture appropriate input images, but the explanations still contained excessive technical detail and jargon. Enforcement mechanisms therefore improved formal compliance but did little to foster deeper engagement with the underlying goals of explainability.

These results pointed to a deeper issue: Most participants lacked a clear understanding of user needs and struggled with perspective-taking. This finding echoes a broader trope in engineering practice: Engineers often have difficulty designing experiences for audiences different from themselves [72]. However, **practitioners lack lightweight, in-situ support** to help them reason about user needs at

the moment of generating explanations. Although the policy mandated clear and accessible explanations, participants lacked strategies for achieving this goal. Like many engineers in practice, our participants had strong technical training but little preparation in writing, communication, or user-experience design. As a result, they relied largely on their own intuition and focused on aspects of explanations that were already familiar to them, such as technical description of models or the use of standard explainability tools. The policy articulated high-level goals and outcome requirements but did not provide concrete guidance for how to achieve them. In particular, it did not offer mechanisms to help practitioners shift perspectives, for example, through personas, interviews, or other user-centered design methods [119]. Without such guidance, participants struggled to move beyond their own technical viewpoint. Expecting developers to independently bridge this gap without training or tools may therefore be unrealistic. If lightweight interventions are intended to shift responsible engineering practices, addressing this challenge will be essential. Expecting practitioners to independently bridge this gap without embedded support may therefore be unrealistic in real-world settings.

These observations motivated the second set of interventions in our study. In the third semester, we introduced LLM-based assistance through role-playing chatbots and writing-coach chatbots. These tools helped participants identify some obvious problems in their explanations, such as missing descriptions or unclear phrasing. However, the overall impact remained limited. Participants engaged with the chatbots only moderately, and improvements to explanations were largely superficial. The tools were effective at flagging blatant issues but did not substantially change how participants reasoned about the needs of end users.

In the fourth semester, we provided educational guidance that encouraged participants to explicitly identify end-user information needs before producing explanations. Participants were introduced to several possible approaches for doing so, such as developing personas, interviewing potential users, or conducting hazard analysis to reason about risks and information requirements. This condition was associated with noticeably stronger engagement and buy-in from participants. Compared with earlier interventions, participants showed greater willingness to reconsider how explanations should be structured and what information should be prioritized for end users. We intentionally designed this condition to elicit the strongest explanations participants could produce, and the results indicate that

participants were indeed able to generate higher-quality explanations under such guidance. Rather than suggesting that extensive education is required in practice, these results highlight the importance of providing structured, task-specific prompts that guide practitioners to explicitly reason about user needs. These results suggest that prompting participants to explicitly reason about user information needs through perspective-taking approaches may be more effective than policy guidance alone, although the study design does not allow us to determine which specific strategy contributed most to this effect.

Overall, the sequence of experiments provides insight into how different interventions support collaboration across the *knowledge boundaries* that arise when designing explanations for ML systems. *Policy guidance* can help establish shared expectations and terminology around explainability, thereby addressing aspects of *syntactic boundaries* (🔤), but on its own it does little to change how developers reason about what information explanations should convey. Lightweight tools such as *chatbots* may help identify surface-level clarity issues, yet they appear insufficient to substantially shift how participants interpret the informational needs of end users. In contrast, interventions that explicitly prompt perspective-taking encourage participants to engage with the *semantic boundary* (💬)—namely, the differences in how developers and end users interpret what explanations should communicate and why they matter. More broadly, these findings suggest that effective interventions take the form of microinterventions—lightweight scaffolds embedded within workflows that support practitioners in reasoning about explanation needs, rather than relying on standalone policies or extensive training. Moreover, improving explainability practices requires mechanisms that help practitioners surface and negotiate different interpretations of explanation needs. In this sense, explanation design ultimately becomes a process of enabling meaningful communication across knowledge boundaries.

We also acknowledge *pragmatic boundaries* (🫂) in this scenario, though addressing this requires broader organizational interventions beyond the scope of this study.

4.3 Intervention C: Encouraging Engagement in Responsible AI (RAI)

The findings of this section have been published in CHI'26*: ""I Don't Think RAI Applies to My Model" – Engaging Non-champions with*

Sticky Stories for Responsible AI Work," which was recognized with the 🏆 **Best Paper Award** [412].

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 [282, 134]. While existing guidelines for RAI assessment, documentation, and tooling [290, 220, 54] 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.

Cf. §3.1 (p.26) and §3.2 (p.50, p.56) for RAI discussions

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 intervention designed to encourage data scientists and software engineers to engage with the principles of RAI. Our intervention specifically targets this *pragmatic 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 can we design interventions that foster deep engagement with responsible AI in ML product development among practitioners?**"

4.3.1 Related Work

Early work seeking to understand industry RAI practices suggested that RAI efforts were often driven by "individual advocates" who are self-motivated to pursue RAI work [201, 282]. In recent days, many practitioners are now formally tasked by their organizations with considering RAI issues [296, 200, 369]. In this paper, we use "*RAI champion*," a term used by organizations such as Microsoft as a role title [288], to refer to both self-motivated advocates and formally designated and trained RAI roles. While there is an abundance of prior work focusing on challenges RAI champions face and how to support them, it remains unclear whether such approaches transfer to the broader group of practitioners that we refer to as *non-champions*, namely those who are not already motivated to lead RAI work but nonetheless encounter RAI concerns in their everyday roles.

4.3.1.1 State of Responsible AI in Industry

Prior research has extensively examined industry RAI practices and challenges. Within the CHI and broader HCI communities, there has been a strong push to better understand industry RAI practices, challenges, and needs [133, 338, 282, 257]. For instance, through interviews and surveys, Holstein et al. [133] identified challenges in fairness-aware data collection and introduced proactive auditing processes. Focusing on UX professionals, Liao et al. [185] and Wang et al. [372] emphasized the need for improved tools and prototyping methods to facilitate communication and collaboration with technical teams when addressing RAI concerns.

Prior research highlights organizational challenges and risks that can limit the meaningful implementation of responsible AI in industry. A large body of HCI and RAI research has shown that individuals frequently encounter push-back from leadership when advocating for more responsible technologies [385,

156, 338, 17, 389, 31]. In addition, despite the many RAI principles, guidelines, and frameworks published by technology companies, organizational studies of industry RAI practices have consistently highlighted how the profit-driven and fast-paced nature of industry work often demotivates practitioners from engaging in meaningful RAI efforts [200, 257, 307, 384]. As a result, multiple studies warn that RAI processes risk becoming bureaucratic "check-the-box" exercises rather than reflective, substantive practices [201, 361, 176, 362, 45]. For instance, RAI documentation is often reduced to a compliance task [201, 86, 61, 390], while fairness and explainability evaluations can become performative practices, sometimes criticized as ethics washing [294, 199, 81, 18].

4.3.1.2 Supports for RAI Harm Identification

An abundance of structured templates and frameworks exist to support practitioners in Responsible AI impact assessment. To support practitioners in identifying potential harms and ethical risks, a wide range of RAI assessment approaches have been introduced, often in the form of structured checklists or impact assessments. For example, Microsoft's *RAI Impact Assessment Template* provides guidelines for conducting impact reviews prior to deploying AI products [217]; Bogucka et al. [39] co-designed and evaluated an AI impact assessment template with practitioners and compliance experts grounded it in regulatory requirements; Deng et al. [80] developed a *Societal Impact Assessment* template focused on design considerations for effective adoption and adaptation; and Rismani et al. [295] argue for the use of established hazard engineering techniques to structure the analysis

More broadly, several tools aim to promote broader reflection on the consequences of technology. Nathan et al. [236] developed a tool to help practitioners envision long-term effects of interactive systems. Elsayed-Ali et al. [90] introduced *Responsible & Inclusive Cards*, an online card-based tool designed to encourage critical reflection on project impacts. Ehsan et al. [89] introduced *Seamful XAI* to allow stakeholders identify mistakes and enhance AI explainability. Documentation frameworks such as *Datasheets for Datasets* [31], originally intended to improve transparency in data collection, have also been shown to help surface ethical concerns [11].

This line of work emphasizes structured processes and tooling as a means to support developers in anticipating harms and fostering reflection.

Recent research has begun leveraging large language models (LLMs) to help AI practitioners reflect on potential risks and harms in their systems. Building on prior HCI work that demonstrated the potential of large language models (LLMs) to support brainstorming and reflexivity, researchers have begun exploring how to incorporate LLMs into RAI tools to support reflection around RAI concerns [54, 251, 375]. Approaches either (a) use LLMs to generate examples of possible harms for a system, following structured reasoning internally to create diverse harms, such as the vignettes generated by AHA! *[54]* and our own work in this paper, or (b) identify and present real-world reports about related systems from news stories as in Farsight [375] and BLIP [251]. Both strategies aim to guide analysis with realistic examples and broaden the range of consequences considered.

We find these tools promising and build on similar ideas. While their design may not explicitly target RAI champions, we suspect that they are more effective for developers already motivated for RAI work. Since motivated RAI practitioners are more likely to volunteer for evaluations of RAI tools (self-selection bias) and the participants' prior motivation was not controlled for in prior studies, we are curious about how effective such tools are for a broad range of practitioners.

4.3.1.3 Engaging Non-Champions

Research suggests that existing fairness and interpretability tools often fail to foster genuine engagement, instead encouraging superficial compliance and limiting meaningful understanding. As some organizations make RAI steps mandatory, either through explicit RAI audit gates [286, 287, 289] or by attaching RAI considerations to existing required privacy assessment steps [79], it remains to be seen whether non-champions engage in depth or just do the minimum amount of work to complete the necessary steps ("check-the-box compliance")

Research has found evidence of such a check-the-box culture: For instance, a participant in Balayn et al.'s study noted, "*Fairness for many companies is just a small checkbox, and sometimes people put their mark without any question...*" [18]. Similarly, Kaur et al. [163] found that interpretability tools, while designed to

improve understanding of machine learning models, can sometimes impair it, with strategies aimed at promoting deliberation and engagement frequently failing to overcome this, and Omar et al. [245] found that structured policy guidance was not effective at engaging with user needs for explanation designs.

Recent research has begun exploring strategies to involve non-champions in responsible AI work, though significant gaps remain. Common strategies to attempt to engage practitioners for RAI work involve nudging [32], gamification [19, 169, 353, 354], and reframing fairness work in familiar quantitative terms [79]. For example, Bhat et al. introduced a JupyterLab extension to nudge data scientists to complete and update model card documentation, particularly the ethics-related sections [32]. Ballard et al. proposed *Judgment Call*, which helps product teams surface ethical concerns using value-sensitive design and design fiction [19], and Kim et al. developed *The Desk: Dilemmas in AI Ethics*, a digital game-based approach to enhance learning about AI ethics [169].

While these strategies can capture short-term attention, it is not clear that they are sustainable to keep non-champions engaged. Nudges may increase initial actions, but cause longer-term behavior changes—and may even reduce follow-through in some cases [143, 414, 269]. Similarly, gamification can spike early engagement, yet motivation drops as the novelty wears off, and in some contexts may impair deeper learning or distract from authentic engagement [298, 116]. For example, Widder et al. [384] argued when evaluating one of these games, that hypothetical contexts created in the game are unlikely to be a viable mechanism for real world change.

In summary, prior research has explored barriers to RAI work and provided many processes and tools to support and engage practitioners in RAI work. However, as we have experienced and will describe next, this support is not equally effective for all practitioners and may fall short for non-champions who are not motivated to go beyond minimally required steps, if any. To the best of our knowledge, prior research focuses mostly (deliberately or not) on RAI champions and has not explored the differences between champions and non-champions. In this paper, we aim to fill this gap, by focusing specifically on how to engage non-champions for RAI activities.

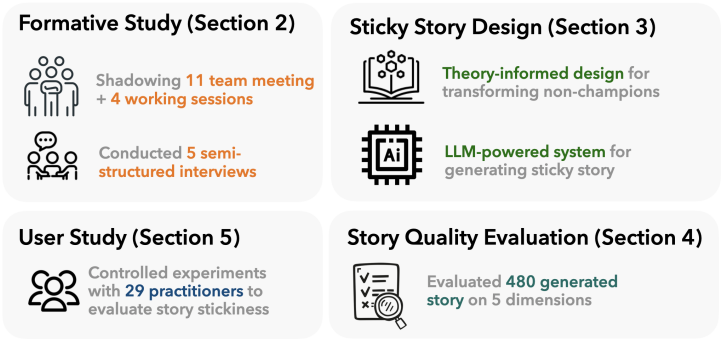


Fig. 4.5 Research overview. The work spans four parts: (1) a formative study, (2) a theory-informed, LLM-powered design for generating "sticky" stories, (3) a quality evaluation of sticky stories, and (4) a controlled user study.

4.3.2 Formative Study

We start by conducting a formative study in close collaboration with a partner technology organization seeking the adoption of Responsible AI (RAI) processes organization-wide (see the study overview in Fig. 4.5). Our initial goal was to explore how best practices could be introduced to strengthen existing RAI governance structures—best practices, such as templates, checklists, and audit processes. However, as the study unfolded, we found that the more pressing challenge was not the refinement of these tools, but the underlying disconnect and resistance among practitioners in engaging with them.

4.3.2.1 Research Method

Data Collection. Our data collection followed a qualitative, ethnographically informed approach, aiming to capture both organizational-level discussions and on-the-ground technical practices. We combined two complementary strategies: (1) shadowing internal governance and project team meetings, and (2) follow-up conversations and semi-structured interviews with team members. Shadowing involved three governance team meetings (April–May 2023), eight project team meetings (July–October 2023), and four one-on-one working sessions (August–November 2023) between a governance team member and a data scientist. To supplement these observations, we conducted two informal one-on-one conversations with developers and five semi-structured interviews: two with governance team champions and three

with data scientists from different project teams. Our documentation relied on contemporaneous note-taking, which foregrounded participants' language, emergent themes, and key tensions, while necessarily prioritizing synthesis over verbatim capture. Across these sessions, we accumulated roughly 22 hours of observation and produced over 50 pages of detailed field notes documenting interactions and discussions.

Data Analysis. Since our goal was to produce rich, context-sensitive insights that directly informs the design of interventions, rather than coding every interaction in a traditional thematic analysis, we adopted an interpretive approach [486, 487]: We examined patterns, recurring tensions, and illustrative examples across meetings, interviews, and conversations to understand how RAI practices were enacted and experienced.

Limitations. Observations were limited to meetings and working sessions, which constrained our view of day-to-day development practices and informal conversations where RAI may have been discussed differently. Second, because we were not permitted to record, our analysis relied on contemporaneous notes that emphasize synthesis rather than verbatim detail, which may have filtered nuance. Third, as with any qualitative analysis, researchers' interpretation shaped how themes were identified and articulated; despite best efforts at collaborative review and discussion among multiple researchers, we cannot entirely exclude the possibility of bias. Finally, our findings are based on a single organizational setting, and while we anecdotally heard similar themes from colleagues in other organizations, the readers should take care when generalizing the findings beyond our setting.

4.3.2.2 Key Finding: Despite Substantial Governance Efforts, RAI Had Minimal Impact, with Practitioners Largely Ignoring or Dismissing RAI Concerns

Even with well-structured governance processes in place, non-champion practitioners largely failed to engage meaningfully with RAI practices. Our observations, working sessions, and interviews revealed three interconnected patterns that explain this lack of engagement:

Consistent with prior research findings [230, 232, 282], practitioners in the organization perceived RAI as abstract and peripheral. While the governance team invested significant effort into artifacts (e.g., templates, workflows, and training modules) with the intention of making RAI practices actionable (such as providing instructions and a step-by-step breakdown of tasks), practitioners often were not sure why these practices mattered for their work. For example, a data scientist completing a system design template skipped sections on fairness and validation, focusing only on tracing data sources that supported her immediate task. In project meetings where the governance team was invited, RAI topics were virtually absent, mentioned only once in passing, and consistently treated as low-risk compared to client deadlines. RAI assessment templates were largely ignored. Interviews reinforced these patterns across teams: One developer explicitly mentioned that she would only use the RAI assessment templates if mandated. Another team completed the templates retrospectively solely to demonstrate compliance to clients. Across these instances, RAI was seen as extra paperwork rather than a tool for identifying or managing ethical risks.

Practitioners outside the dedicated governance *team were largely dismissive of RAI concerns, even when directly prompted.* While practitioners rarely engaged with RAI practices on their own, explicit inquiries also led to dismissive responses. For example, after several project meetings, when we asked a data scientist about potential RAI concerns for their project, they immediately responded, "*there are no responsible AI concerns for this project*," without further reflection. We conjecture that this dismissiveness is partly driven by narrow media narratives, which frame fairness primarily around gender and race. Consequently, this may have reinforced a mindset that RAI concerns are limited to these areas, so when a project did not directly involve gender or race, practitioners overlooked and dismissed other important risks, such as privacy violations, safety issues, algorithmic bias in other demographic groups, entirely without deliberation.

Governance team efforts, though substantial, fell short of effectively motivating practitioner participation in RAI. Governance team members were highly supportive, providing active coaching, clearly explaining RAI goals and processes, and guiding practitioners through training materials grounded in academic and industry best practices. They proactively monitored progress, clarified doubts, and encouraged reflection on potential RAI risks. The organization even enlisted external

support—including our team—to further enhance these interventions. Despite this intensive effort, however, data scientists remained disengaged, completing only minimal tasks to "check-the-box" when directly asked by their managers and resisting prescribed processes. This underscores how carefully designed and actively supported governance mechanisms still require practitioner buy-in.

These observations led us to rethink our approach, considering how to foster genuine engagement with governance practices, rather than simply designing more mechanisms or forced compliance.

4.3.3 Research Design

Our goal is to motivate "non-champions", that is stakeholders who may be indifferent or skeptical, to meaningfully engage in RAI efforts by making the consequences of neglecting its principles feel real and relevant. In this section, we discuss established theories about how practitioners' attitudes and engagement with RAI can be shifted or transformed, and how these theories inform the design decisions behind our intervention.

4.3.3.1 Theoretical Background and Design Principles

Transforming Non-Champions. Our initial aim was to motivate non-champions to care about RAI, and *transform* into champions. We drew on *Transformative Learning Theory* [213], which explains how people change underlying beliefs through *critical reflection*. Central to this, is the concept of *disorienting dilemma*—an event, challenge, or scenario that disrupts assumptions and compels self-examination. In practice, transformation is difficult and slow, requiring interventions that actively provoke reflection rather than passive exposure [380, 214]. *Cognitive Dissonance Theory [101]* complements *Transformative Learning Theory* by describing how psychological discomfort (*dissonance*) from conflicting beliefs can drive a new perspective.

Together, these theories suggest that effective interventions must present challenges to existing assumptions and support structured reflection. Prior research in domains such as ethics education, clinical training, and diversity initiatives demonstrates that reflective prompts, scenario-based exercises, and facilitated discussions can trigger these processes [204]. For example, in health professional education,

curricula designed around transformative learning principles have been shown to improve practitioners' ethical reasoning, and critical reflection [301]. Similarly, in higher education, faculty engaged in action research grounded in transformative learning reported meaningful changes in their teaching practices [112]. However, lasting mindset change requires more than a single exposure—it depends on repeated reinforcement, supportive contexts, and engaging formats [380, 261]. The central question is how to design interventions that provoke these mechanisms to trigger such a transformation among RAI non-champions.

Our aim is to trigger such a transformation, not just "tricking" professionals to engage in short term (e.g., with nudging or gamification). Therefore, we sought to create moments of *disorienting dilemmas* to cause *cognitive dissonance*—points of discomfort strong enough to disrupt assumptions. However, not all disruptions are equally effective. Our formative findings revealed that practitioners had become desensitized to widely circulated media narratives of bias framed around gender and race, often dismissing as irrelevant to their projects. To overcome this resistance, we surfaced dilemmas that were likely unfamiliar to the practitioners and directly consequential within their work contexts.

Sticky Stories as Intervention. One way to capture practitioners' attention is by showing them stories of harm caused by their own ML systems. Prior work has demonstrated that vignettes and fictional scenarios can be effective for RAI champions [54], but we expect non-champions may require more than generic stories: They need something to disrupt and capture their attention, something that differs from common well-known media stories they already dismissed as irrelevant to their work. We aim to create stories (see Fig. 4.6) that not only illustrate harms in the moment but also resonate deeply and are remembered later to seed an actual transformation.

To achieve such impact, we turned to marketing theory, grounding our approach in the principles of *Made to Stick* [126], which summarizes characteristics of memorable and persuasive ideas. While *Transformative Learning Theory* and *Cognitive Dissonance Theory* describe the stages of transformation and the need to create reflective moments, they do not specify how to craft materials that consistently capture attention and sustain engagement. *Made to Stick* offers a practical framework, suggesting strategies that can trigger these mechanisms. We operationalized these

Fig. 4.6 Examples of different types of stories: (Bottom) Baseline story, which tends to be generic and straightforward but less memorable, and (Top) Sticky story, which incorporates elements such as surprise and emotional resonance to encourage deeper engagement and recall. We design (Section 4) and validate (Section 5) the sticky stories to be concrete, surprising, and severe. Our user evaluation (Section 6) with RAI practitioners demonstrates that sticky stories can better engage non-champions and lead them to more critical reflections.

principles into five key story qualities to guide the construction and evaluation of our **sticky stories**:

- 😮 **Surprisingness.** Stories should present harms in ways that disrupt default assumptions, that are counterintuitive or non-obvious. This quality captures attention, which is a prerequisite for reflection and potential attitude change. Evidence from cognitive science shows that unexpected stimuli attract attention and trigger deeper processing [149]. By making harms surprising, practitioners are more likely to notice risks that would otherwise be overlooked and experience disorienting dilemmas.

- **Concreteness.** Stories should include tangible details, such as named roles, real-world analogues, or observable system behaviors. Concreteness helps audiences visualize harms and form emotional connections, making abstract principles more understandable and memorable. Dual Coding Theory [88] supports this approach, showing that concrete information is encoded both verbally and visually, improving recall compared to abstract descriptions.
- **Severity.** Severity emphasizes the magnitude and scope of potential harm, highlighting why certain outcomes demand attention and action. Perceived severity motivates engagement, as individuals are more likely to respond to risks they judge serious. Research on the affect heuristic [335] demonstrates that people's risk judgments are strongly influenced by the emotional weight of outcomes, with severe consequences eliciting stronger reactions and prompting protective or corrective behaviors.
- **Relevance.** Stories should align with domain-specific experiences and stakeholder concerns, ensuring that messages feel applicable to practitioners' work. Relevance increases the likelihood that examples are processed deeply and influence attitudes or behaviors [207]. By situating lessons in authentic professional contexts, practitioners can connect the scenarios to real decisions, enhancing engagement and reflection.
- **Diversity.** A broad range of stakeholders, harm types, and system behaviors can avoid narrow or stereotypical portrayals. Evidence from narrative transportation research indicates that encountering multiple perspectives enhances engagement and supports attitude change [113]. Diverse stories help practitioners anticipate harms across contexts rather than focusing on isolated examples.

Capturing Early Signs of Change: Critical Reflection. Our intervention used *sticky stories* to create *disorienting dilemmas*, aiming to prompt reflective thinking. While our ultimate goal is *transformation*, genuine perspective shifts are difficult to observe without extended observation windows over multiple years; immediate responses may not indicate lasting change. To keep the scope of our research manageable, we focus on *early indicators of transformation*, observing moments when *disorienting dilemmas* triggered *critical reflection*.

To observe signs of critical reflection, beyond just short-term measures of engagement, we focused on concrete behavioral signs that participants were moving beyond surface-level reactions. Prior work defines *critical reflection* as moving beyond descriptive or casual reflection to actively examine one's assumptions, beliefs, and actions, evaluating their validity and potential consequences [213]. In the context of responsible AI, this would involve questioning default practices, recognizing ethical risks, and considering how one's work may contribute to harm. To systematically detect these moments of *critical reflection*, we identified *concrete behavioral indicators,* summarized in Table 4.9. We will use these indicators in our evaluation in Sec. 6, tracing how these moments manifested during participants' engagement with the stories.

Sign	Description	Indicators	Example
Challenges assumptions [213]	Practitioners critically examine or dispute underlying beliefs, premises, or taken-for-granted assumptions that might otherwise go unquestioned.	Identifies own beliefs that may not hold; contrasts story assumptions with their own understanding, experience, or evidence; expresses skepticism toward "default" ways of thinking or doing things.	***P23***: *"I think human evaluators are really important – it's something that I am kind of understanding now. We haven't done this. But looking at the stories, I think, it's a really important for us"*
Explores multiple perspectives [52]	Practitioners consider scenarios from multiple angles or stakeholders, compare alternatives, or expand the scope beyond their immediate perspective.	Weighing different interpretations; comparing different parts of a story; acknowledging multiple voices.	***P15***: *"As a data scientist, I can retrain the model, and I can test it. But what can a government or the person in FDA do? They don't know this."*
Connects to wider systems or past incidents [52]	Practitioners connect the story to broader organizational, social, or technical systems, often moving beyond the immediate prompt.	Linking story implications to other domains, contexts, or systemic risks; recalling real-world events.	***P4***: *"[...] like scholarship distribution and funding distribution specifically, whatever was coming to any charity. How are they distribute it amongst like schools or old age homes and other organizations where the funding has to go to."*
Expresses surprise [213]	Practitioners show surprise, novelty, or realization about aspects of the scenario, signaling a shift in understanding.	Expressions of being surprised, not having thought of it before, finding something new, unexpected, or revealing.	***P19***: *"oh, that's mind blowing [...] I didn't know this can happen."*
Engages in iterative thinking [312]	Practitioners demonstrate a back-and-forth reasoning process, revising or expanding their views as they talk.	Evidence of reconsideration, self-correction, extended reflection, or stepwise elaboration.	***P26***: *"Now that I'm looking at the word like demeaning. I feel like I can probably think of a couple more examples."*
Plans intentional change [212]	Practitioners translate reflection into concrete plans for action or acknowledge the need to modify future behavior, processes, or designs.	Commitment to follow-up action; expressing intent to discuss with others; specific takeaways for their own projects.	***P30***: *"I think, filtration of the training data is pretty simple. It could be done with a simple Regex based filtration, for at least getting rid of such harmful potential comments. And post filtration for the same should be simple, too. Yeah. So I think that's pretty painless to deploy. I would be motivated to do that."*

Table 4.9 Signs of Critical Reflection in Non-Champions

4.3.3.2 Generating Sticky Stories

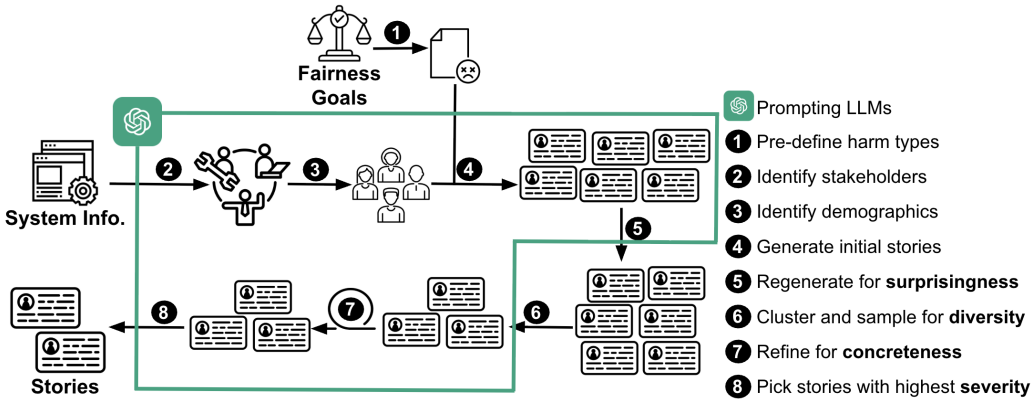


Fig. 4.7 Pipeline for *Sticky Story* Generation

Generating high-quality *sticky stories* is non-trivial. We found that single zero-shot or few-shot prompts with LLMs were not very effective in generating stories that meet all five of our criteria, as they often produce outputs that are overly generic, repeat common tropes (e.g., race and gender only), and fail to capture surprising or contextually relevant harms. Without structured guidance, LLMs struggle to systematically combine diverse harm types, non-obvious stakeholders, and generate stories that effectively provoke reflection. This motivated our design of a systematic and scalable compound AI system [356] that combined prompt engineering techniques with programmatic control to ensure that each story consistently embodied the five desired qualities. We broke down the task into a coherent sequence of logical steps and iterated through multiple design cycles to refine each component. This resulted in the following eight-step pipeline (cf. Fig.4.7):

- **Input (🎯).** To make the stories relevant to non-champions, we grounded them in the specific projects participants were working on. As inputs, we collected the ML system's description, its intended purpose, and a representative stakeholder use case. These inputs seeded the rest of the pipeline, ensuring that the generated stories were *relevant*—directly tied to the participant's own ML system rather than abstract or generic examples.
- **Step 1: Pre-define Harm Types (🎨).** To ensure diversity and coverage of edge cases, we began by specifying a set of harm categories. Pre-defining these categories grounds story generation in well-theorized frameworks of

harm rather than ad hoc examples. For this, we drew on the taxonomy of fairness-related harms from prior studies of harm categories [54, 327] and fairness goals from Microsoft's RAI assessment guide [217]. The set included cultural misrepresentation, reinforcement of biases, unequal access to opportunities, and erasure of minorities.

- **Step 2: Identify Stakeholders (,).** To generate stories that capture diverse harms, it is also essential to identify stakeholders whom practitioners might otherwise overlook. For example, in a movie recommendation system, practitioners often focus on obvious stakeholders such as movie watchers, but may miss stakeholders such as movie producers, whose livelihoods depend on whether their films are surfaced. To account for these cases, we go beyond conventional direct and indirect stakeholders by introducing *direct-surprising* and *indirect-surprising* categories. These highlight marginalized or non-obvious groups, following the principles of *Design Justice* [74], which emphasizes attention to those often overlooked in design processes. We prompt an LLM to generate these different sets of stakeholders based on the product description. By surfacing unexpected stakeholders, the stories are more likely to create *disorienting dilemmas* that challenge practitioners' default assumptions.
- **Step 3: Identify Demographics (,).** To ground stories in concrete contexts, and relevant to the user, we then generate possible demographic attributes for each stakeholder (e.g., age, gender, ethnicity). This step facilitates subsequent steps in tailoring harms to marginalized or contextually relevant groups.
- **Step 4: Generate Initial Stories ().** To increase diversity in our stories, for each harm–stakeholder combination, we generate an initial set of harm stories, forming a matrix of harms and affected users. This matrix-based approach, inspired by prior work [54], ensures broad coverage and combinatorial richness. By systematically exploring combinations, we reduce the risk of narrow or stereotypical examples, and instead highlight harms that may not surface through ad-hoc brainstorming.
- **Step 5: Regenerate for Surprisingness ().** Given that the LLM might default to producing stories that align with patterns it frequently encounters, to avoid bland or generic outputs, we prompt the model to regenerate stories

using earlier outputs as counterexamples—encouraging less typical, more striking, and surprising narratives.

- **Step 6: Cluster and Sample for Diversity (🎨).** To further increase diversity and avoid redundancy, we transform the stories into sentence embeddings, apply K-means clustering (k=10), and sample from the five least-populated clusters—those most likely to contain unique narratives.
- **Step 7: Refine for Concreteness and Severity (🪪, ☠).** Concreteness makes harms tangible and easier to visualize, increasing practitioners' emotional engagement. To make sure the stories are concrete and severe, we employ a two-stage refinement loop: one model refines stories for concreteness and severity, and a second evaluates the output. Stories lacking specificity or clarity are iteratively revised (up to three times) to meet our concreteness standard.
- **Step 8: Pick Stories with the Highest Severity (☠).** Rather than enumerate every harm comprehensively (as other tools [54, 375] and hazard analysis [295, 180] pursue), we aim to provoke and persuade with a small number of high-impact stories. Therefore, in a final step, we prompt an LLM to select the two stories with the greatest magnitude and scope of harm, while ensuring they satisfy all five qualities.

4.3.3.3 Sticky Story Integration in a Tool

To demonstrate how sticky stories could be presented to practitioners and to run our evaluation study, we integrated the pipeline into an interactive tool. The tool is designed to replicate Microsoft's RAI assessment guide [217], which is typically completed as a static, text-based template.

Users begin by entering a brief description of the ML system, its intended purpose, a user story, and system stakeholders (Fig. 4.8-A (1, 2)). Subsequently, during the fairness assessment step (Fig. 4.8-B and C), users can request brainstorming assistance, which presents the previously generated sticky stories (Fig. 4.8-D) for the current assessment step.

To reduce potential delays and maintain a smooth user experience, these stories are often generated in earlier screens of the tool—based on the user's description of the ML system—and stored for retrieval in the subsequent fairness brainstorming

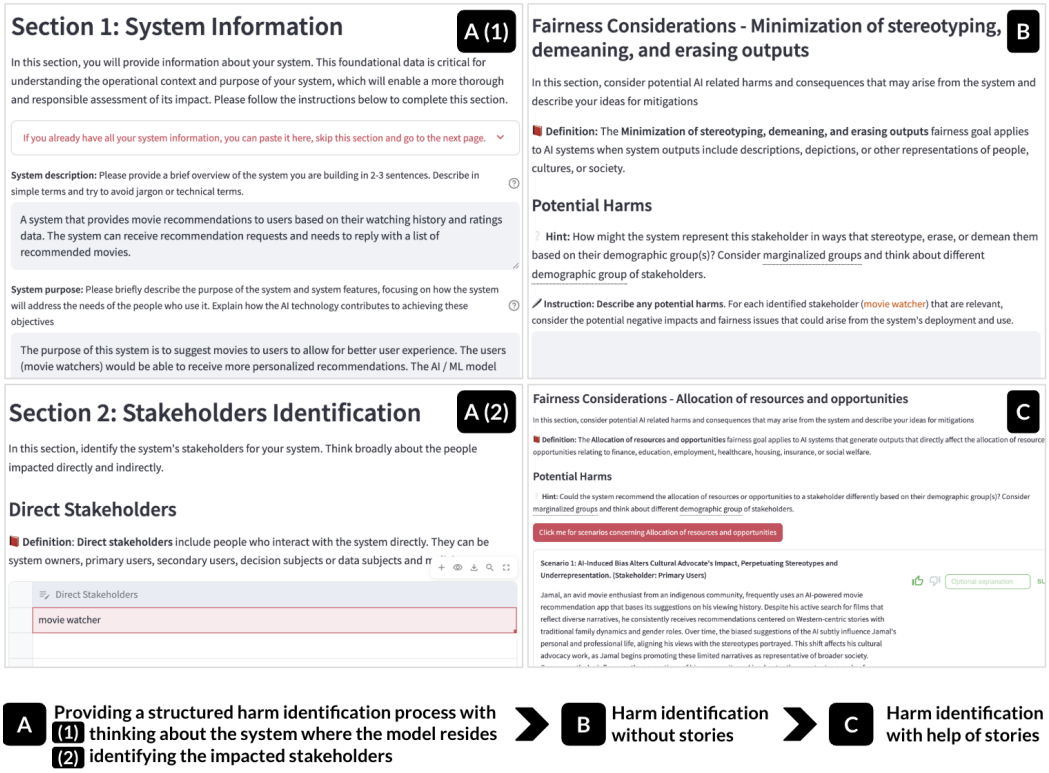


Fig. 4.8 Snapshots of the Tool Integrating the Sticky Story Generation Pipeline

step. This approach can help ensure that stories appear quickly when requested (story generation with the GPT-4o model typically takes 3–4 minutes) helping users focus on the task without waiting—though in practice the timing may vary depending on system load and context. The tool shows two stories by default, but users can request up to three more stories, provide feedback, and regenerate stories.

4.3.4 Evaluation I: Evaluating Stickiness of Harm Stories

We first conducted an offline evaluation to understand the quality and cost of generating sticky stories with our designed pipeline. In the evaluation, we curate diverse AI application scenarios and run our pipeline and an ablated version to generate the harm stories. We then measure the quality of the generated stories with the five desired qualities of sticky stories: **concrete (), severe (), surprising (), diverse (), and relevant (),** as well as the cost of generating these stories in terms of token usage and time elapsed.

4.3.4.1 Experiment Setups

Data. We collected diverse AI application scenarios from the Internet (e.g., [275, 273, 222]) and randomly sampled 15 scenarios (e.g., *voice assistants*, *image search*, *email monitoring*, and *demand forecasting*) for our evaluation.

We used an LLM (`gpt-4o`) to process the searched content into more detailed descriptions, similar to what a user would have input to our system, and we manually verified that these generated descriptions are valid.

Methods. For evaluating the generation capabilities, we implement most of our pipeline with `gpt-4o`, as it demonstrates strong writing capabilities [247]. In step 7, however, we use a smaller model `gpt-4o-mini` as the evaluator to reduce cost, as validation usually requires less capabilities than generation. We use `mxbai-embed-large-v1` to produce sentence embeddings for K-means clustering.

Baseline. We compare our pipeline approach to a zero-shot prompting baseline, in line with prior work [54]. The prompt directly instructs an LLM (`gpt-4o`) to generate scenarios that illustrate harm to relevant stakeholders, and we instruct that the baseline prompts generate stories around 175 words, which is the average length we observe from the stories generated by our pipeline. We share all prompts used in our supplementary material.

Story generation. For each AI application scenario, we generate two stories for each of the two fairness goals (*Quality of Service*, and *Allocation of Resources and Opportunities*). In total, we curated 120 sticky stories and 120 baseline stories. This sample size allows us to draw conclusions with 90% confidence level with 8% margin of error.

Metrics. We evaluate the quality of the sticky stories and baseline stories and the cost of the generation method. For cost, we measure the time it takes to run the pipeline and the number of tokens it costs. For quality, we evaluate the stories on the five desired qualities of *sticky stories*. Four quality metrics (*concrete*, *severe*, *surprising*, *relevant*) are evaluated with a two-phase evaluation involving both human raters and LLMs, with diversity evaluated by a separate established distance metric. The final evaluation covered **240 stories** produced by both methods.

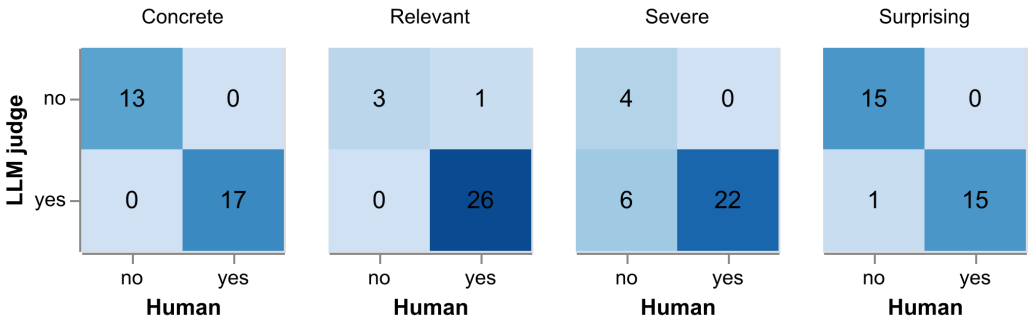


Fig. 4.9 Humans agree well with LLM judges on whether a story is concrete, relevant, or surprising. While there is more disagreement on severity, we found they all stemmed from LLM being overly generous in rating baseline stories as severe.

Human Annotation and Inter-Rater Reliability. Three researchers independently evaluated a set of 20 harm stories using binary (yes/no) judgments across the four qualities of stickiness. We conducted multiple calibration rounds, during which we refined the definitions of the qualities and clarified edge cases based on observed disagreements. After we reached consensus, we developed the finalized rubric (see supplementary material) for a larger-scale evaluation.

Scalable Evaluation Using LLM-as-a-Judge. To scale the evaluation to the full set of generated stories, we adapted our rubric for an automated evaluation using `gpt-4o`. The prompt included the story text, plain-language definitions of each of the qualities, and a binary decision task for each dimension (0 = no, 1 = yes). Four of the five dimensions were rated using binary LLM judgments.

To validate the reliability of the LLM judgments, one researcher independently annotated 30 additional stories. The Cohen's Kappa values for agreement between human and GPT-4 annotations were: Concrete: 1.000, Severity: 0.4783, Surprising: 0.9356, and Relevant: 0.8387. All disagreements on Severity stemmed from the LLM being overly generous in rating baseline stories as severe, thus overinflating the severity results of the baseline (see Figure 4.9) – hence, results regarding severity likely underestimate the real effect.

Measuring Diversity with Embedding Distances. For diversity, we computed cosine distances between semantic embeddings of the story titles (higher distance indicates higher diversity), as it is a well-established distance measurement in the literature [285].

	Quality					Cost	
	Severity	Surprising	Concrete	Relevant	Diversity	Token	Time/s
Pipeline	**0.992**	**0.783**	**1.000**	0.892	**0.156**	56665	50
Baseline	0.683	0.354	0.017	**0.979**	0.098	1232	9

Table 4.10 Offline Evaluation Results of the Generated Stories.

4.3.4.2 Limitations (Threats to Validity)

Despite extensive validation, internal validity may be affected by biases in LLM-based judgements, especially on severity. To mitigate the potential verbosity biases of LLM judges [405], we controlled for story length by generating texts of comparable length across conditions. Our binary ratings for each quality captures only big differences and may not represent more nuanced quality differences. External validity is constrained by the small, curated set of 15 scenarios, which may not represent all AI applications.

4.3.4.3 Results

Overall, we found that our pipeline is able to generate sticky stories that are more concrete (+98.3%), more severe (+30.9%), and more surprising (+42.9%) than baseline stories, demonstrating the effectiveness of our design (see Table 4.10 for more details). The sticky stories are also generally more diverse (+5.8%), due to our clustering approach. However, we do observe a small trade-off in relevance (-8.7%), as sometimes the generated sticky stories are overly dramatic and can be hard to relate. In addition, generating sticky stories requires more resources (5.5x time and 46x token usage) than the zero-shot generation of baseline stories. As we will show next, this cost is likely acceptable, as the sticky stories are indeed more effective at engaging practitioners and inspiring them to think of RAI harms beyond their existing mindsets.

4.3.5 Evaluation II: User Study

After showing that sticky stories indeed embody the five desired qualities (Sec. 5), we now assess their practical value, that is, whether sticky stories actually lead to

greater engagement from (non-champion) practitioners. We conducted a user study that explored how non-champion practitioners engage with harm identification tasks with and without stories. We analyzed differences in terms of (1) the time participants spent identifying harms, (2) the number of new harm categories they surfaced, and (3) the depth and nature of their *critical reflections* on the harms or stories.

4.3.5.1 Study Design

To evaluate the impact of *sticky stories* on practitioner engagement, we conducted a **mixed-design user study** that combined both **within-subject** and **between-subject** elements. Unlike prior work that often focuses only on champions, we deliberately sought to include non-champions, and ended up with a range of practitioners with varied levels of RAI motivation. Each participant completed two harm identification tasks under two of three conditions: *no stories*, *baseline stories*, or *sticky stories*. This design allowed us to disentangle the effect of story presence from the distinct qualities of sticky stories, while also partially controlling for potential learning and ordering effects. We assessed engagement through multiple indicators, including time spent, number and diversity of harms surfaced, and qualitative signs of critical reflection in response to the stories.

Participants and Recruitment. Unlike prior studies that did not account for self-selection bias, which likely led to primarily recruiting participants already motivated by RAI concerns, we sought to recruit *non-champion* practitioners—those less inclined to prioritize RAI in their work. Recruiting this group was inherently difficult, as they are unlikely to volunteer for a study framed around Responsible AI. To overcome this, we strategically oversampled through broad advertisements that emphasized ML evaluation broadly, and probed participants' preferences for different evaluation techniques in a screening survey. This design choice helped attract a broader and more neutral practitioner audience, including individuals who are less inclined toward fairness assessments and thus more representative of non-champions whom our intervention seeks to engage. Recruitment and study protocols were approved by our Institutional Review Board (IRB).

We recruited participants through professional platforms such as LinkedIn, Twitter, and a large Slack community for data scientists. Interested individuals completed the screening survey, which included questions regarding their familiarity with concepts such as model training, model evaluation, model fairness, AI ethics, and MLOps, and how useful they think various evaluation activities are, including in-distribution data evaluation, out-of-distribution data evaluation, model red-teaming, and responsible AI auditing (e.g., fairness). This enabled us to identify practitioners who do not prioritize RAI in their work. We received a total of 291 responses. We excluded submissions that indicated low engagement or fraudulent behavior—such as vague project descriptions, suspicious email addresses, or missing LinkedIn profiles—and filtered participants based on their stated level of high RAI interest. We conducted 5 pilot experiments to test and refine the study protocol. In the pilot experiments, we observed fairly strong effects, suggesting that we could reliably detect true effects with moderate numbers of participants. Afterward, we recruited 31 participants, but due to a tool malfunction in which the baseline stories failed to generate, data from two participants could not be analyzed. That is, we successfully conducted the study with **29 participants**. Each participant received as compensation a $35 gift card.

Experimental Conditions. To evaluate the impact of the sticky story intervention on practitioner engagement, we designed three experimental conditions. Engagement could potentially be influenced either by the presence of any illustrative story or specifically by the *sticky stories*. To disentangle these effects, we implemented two control conditions and one treatment condition.

- **Condition A (No Stories - Control 1):** Participants complete a section of the RAI assessment without being shown any stories.
- **Condition B (Baseline Stories - Control 2):** Participants complete a section of the RAI assessment while being shown two baseline harm stories (see Sec. 5.1.2; zero-shot prompting, matched for length of sticky stories).
- **Condition C (Sticky Stories - Intervention):** Participants complete a section of the RAI assessment while being shown two *sticky stories*.

Tasks. Each participant completed two tasks focused on harm identification and mitigation planning. To ensure ecological validity [166] and help participants en-

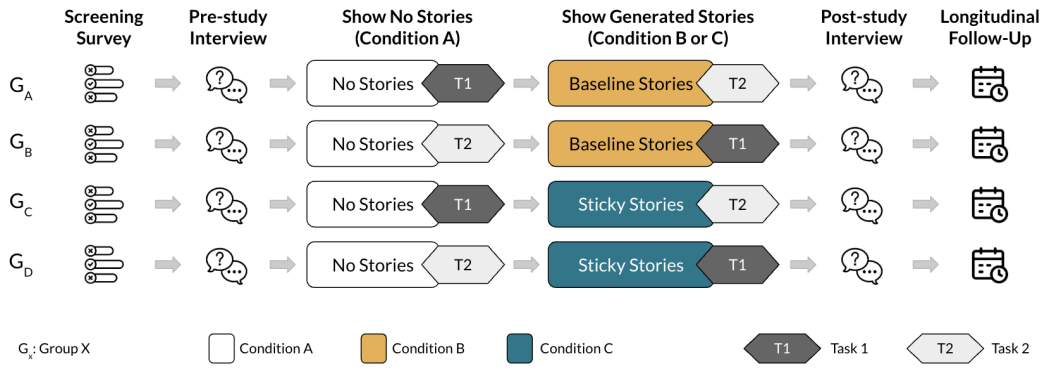


Fig. 4.10 The mixed study design for the user study that combines both within-subject and between-subject elements, to evaluate the effectiveness sticky stories compared to no stories and baseline stories

gage more deeply with the task that is realistic and personally relevant, participants analyze *their own projects.* Each task asks the participant to identify harms and possible mitigations for one of two fairness goals from Microsoft's RAI assessment guide [217]—a well-established framework developed independently of our study:

- **Task 1:** Analyze fairness regarding allocation of resources and opportunities.
- **Task 2:** Analyze fairness regarding stereotyping, demeaning, and erasing outputs.

Each participant worked on both tasks in a random order. Analyzing their own projects ensures that the evaluation reflects realistic, personally relevant contexts.

Groups. We randomly assigned participants to one of four groups. Each group completed their first task in the no story condition (Condition A), followed by the other task in either the baseline (Condition B) or sticky story condition (Condition C), see Fig 4.10. This design enables both **within-subject** comparison (e.g., engagement with vs. without stories) and **between-subject** comparison (baseline vs. sticky stories). In addition, when participants worked on the first task, we could already generate stories for the second task in the background. By randomizing assignment and counterbalancing the order of fairness goals, we reduce confounds related to task complexity and learning effects.

Study Protocol. Each participant completed a pre-study survey, answered a few questions to establish their background, worked on the two tasks with and without

stories, and finally debriefed with the facilitator. This usually took about 60 minutes. Two months after the experiment, we sent a follow up survey.

Pre-Study: Participant Background and RAI Orientation. We collected background information to understand each participant's experience and relationship with RAI practices (e.g., their exposure to RAI), both through a short survey and a brief verbal discussion (see *Interview Guide* in supplementary material). Drawing on stated choice research [194], to increase reliability, we ask questions about behaviors (revealed preferences) rather than preferences (stated preferences). Due to the sensitive nature of responsible AI and non-championship, we intentionally did not collect details about user study participants' demographics, following prior work on responsible AI practices in industry [78, 133, 199, 201].

During the Study: Think-Aloud Harm Identification. While the participants worked on the two tasks, we employed a **think-aloud protocol** to capture participants' real-time thought processes and reasoning while engaging with harm identification tasks, allowing us to understand not just what they identify but how they interpret and respond to different stories. We issued only minimal prompts to preserve the validity of participants' cognitive processes during harm identification tasks [92, 41]. The facilitator maintained a neutral stance and refrained from influencing participants' reasoning—intervening only when they misunderstood the task or deviated significantly from it. In conditions where participants were exposed to stories (either *baseline* or *sticky*), we deliberately avoided leading questions to minimize bias. Instead, we used minimal, open-ended prompts (e.g., "*What are your thoughts as you read this*?" or "*Feel free to keep thinking aloud*") to encourage continued verbalization and self-reflection.

Post-Study: Reflections and Intentions. Immediately after completing the tasks, we invited participants to reflect on their experience by answering open-ended questions about how the task influenced their understanding of fairness-related harms and whether they intended to take any concrete actions or revisit decisions in their own projects. These reflections allowed us to capture participants' intentions to act—serving as a proxy for how compelling, relevant, or actionable they found the experience.

Post-Study Follow-Up (2 Months Later). To assess whether the intervention led to sustained engagement or reflection, we conducted a follow-up survey two

months after the main study. In the survey, participants were asked two open-ended questions:

- *Have you reacted to or done anything based on the findings from our session (e.g., changed anything in your past or new projects directly or indirectly based on concerns raised during our discussion)?*
- *Have you had any discussions—positive or negative—about responsible AI with your peers since the session?*

While insufficient to measure long-term transformations (which may require years), this still captures some concrete **behavioral and social impact over time**, even if modest, indicating whether participants engaged with RAI concepts in their professional communities beyond our study.

4.3.5.2 Data Analysis

To evaluate how practitioners engage differently across study conditions, we defined specific goals and aligned our data sources accordingly. Our primary goal was to understand engagement and reasoning in response to baseline versus sticky stories. We operationalized this goal using targeted questions paired with corresponding metrics (Table 4.11), following the established Goal-Question-Metric (GQM) approach [22]. We combined quantitative metrics (e.g., harm diversity, time spent, user characteristics) with qualitative coding to characterize participants' engagement and reasoning processes (see the finalized variables in Table 4.12).

Quantitative Analysis. To evaluate whether practitioners invest more time when exposed to sticky versus baseline stories—a common proxy for engagement, e.g., [104]—we extracted the total time spent in each section along with the free-text harms from tool logs. To measure harm diversity, each harm was manually assigned to a category and subcategory from an established RAI taxonomy [54]. We assigned each participant a score regarding their prior RAI awareness and championship, based on self-reported experience in the pre-study survey and our assessment of their answers to our initial questions (revealed preferences), which we used as controls in our analysis. Table 4.12 summarizes the variables used in our study. We use ANOVA to analyze the influence of the experimental condition on the dependent variables. Model diagnostics—including checks for normality, homogeneity of

Goal	Question	Metric	Data Source
Understand engagement with stories	Do participants identify more potential harm categories and invest more time when exposed to sticky vs baseline stories?	Number of harms flagged, diversity of harms (coded according to harm taxonomy), time spent per task	Tool interaction logs
Understand reasoning processes	How do participants justify harm identification under baseline vs sticky story conditions?	Comparative coding for reasoning patterns	Think-aloud transcripts
Assess reflection and follow-up	Do participants act on insights or discuss them with peers?	Presence/absence of reported actions, types of discussions with peers	Follow-up survey responses

Table 4.11 Goal-Question–Metric (GQM) alignment for evaluating practitioner engagement and reasoning.

variance, and influential points—were performed to ensure the validity of the analyses.

Qualitative Analysis. To understand *how* participants engaged with harm identification tasks—beyond what quantitative metrics could capture—we conducted *qualitative content analysis* [135, 178], a method that affords quantitative analysis of qualitatively coded data. We used a carefully designed codebook combining theory-driven indicators and patterns emerging from the transcripts (Table 4.9). In particular, deep engagement was coded using the indicators of critical reflection described in Section 4.1, supplemented with additional codes for shallow engagement. Two independent coders applied the codebook to the think-aloud transcripts, and inter-rater reliability was calculated to refine the codes, yielding a Cohen's κ of 0.72, which indicates substantial agreement. Codes were then applied systematically with each participant's transcript as a chunk of analysis, that is, multiple mentions by the same participant were not counted repeatedly. To analyze evolution of participant behaviors and identify trajectories, we used inductive qualitative coding: the first author systematically reviewed session notes and interaction logs, identified recurring patterns, and synthesized these into the practitioner profiles. This data-driven approach enabled us to confidently assign all participants to pro-

Type	Variable	Operationalization	Source
Independent	Story condition	No story (A) vs. baseline story (B) vs. sticky story (C)	–
Dependent (Quant.)	Time spent on harm identification	Duration of engagement, extracted from tool logs	🖩
Dependent (Quant.)	Number of harms	Number of identified harms, extracted and counted from tool logs	🖩
Dependent (Quant.)	Distinct harm categories	Number of unique categories/subcategories identified, coded with RAI harm taxonomy [54]	🏷
Dependent (Qual. & Quant.)	Engagement behaviors	Frequency of coded signs of critical reflection (Table 4.9), from think-aloud transcripts	🏷
Dependent (Qual.)	Self-reported planned actions	Reported and intended actions from follow-up survey responses	✎
Dependent (Qual.)	Transformation trajectories	Shifts in perspective, identified from transcripts and video recordings	🏷
Control	RAI awareness score	0–2 scale, based on in-session remarks: 0 = unaware, 1 = partially aware, 2 = aware	🏷
Control	Championship score	0–5 scale of advocacy for RAI principles in work 0 = unaware, 1 = actively opposed, 2 = acknowledges importance but takes no action, 3 = follows processes reluctantly, 4 = willing to contribute but not advocating, 5 = actively promotes RAI practices among peers	🏷
Control	Prior AI/ML experience	Self-reported in pre-screening survey: 1-2 years, 3-5 years, 6-10 years, more than 10 years	✎
Control	Task/fairness goal order	Randomly assigned	-

Table 4.12 Study variables with type, operationalization, and data source. Icons indicate data source: ✎ = self-reported, 🖩 = derived, 🏷 = coded.

files based on their observed behaviors, even though the process was exploratory rather than codebook-based.

Follow-up survey responses were also examined to capture participants' self-reported takeaways and any indications—planned or already undertaken—of applying these insights in their own projects.

4.3.5.3 Limitations (Threats to Validity/Credibility)

As every study, ours needs to make tradeoffs and has limitations. Our study captures short-term engagement rather than lasting behavior change; even with a two-month follow-up, we cannot establish long-term effects. While our sample size is sufficiently powered to detect large effects, it is not large enough to explore differences across domains and organizations. Biases are possible at several levels: social desirability (despite neutral prompts, avoidance of "like/dislike" questions, and focusing on revealed preferences), bias in participant selection (even with broad, neutral framing, oversampling, screening for non-champions, and limited snowballing), and researcher experiences (including our formative study) shaping questions we ask and how we code data . Internally, the mixed design places the no-story control first for all participants, so any story is confounded with appearing second; learning, priming, fatigue, and carryover from the first fairness goal may influence observed gains. The think-aloud protocol can also alter problem-solving strategies and make engagement appear higher than in ordinary work. Our engagement proxies—time on task and counts of distinct harm categories—are imperfect; time can reflect confusion, and breadth can trade off with depth. Finally, while anchoring tasks in participants' own projects improves realism, it introduces heterogeneity that can mask or mimic effects, and the study examines a single researcher-facilitated exposure rather than repeated use in everyday workflows. We deliberately designed the study, trading off various qualities and accepting some limitations; readers should interpret our results accordingly.

4.3.5.4 Findings

Finding 1: Practitioners spent significantly more time on harm identification tasks when sticky stories were shown. Participants with *sticky stories* for their second task spent substantially more time on harm identification tasks compared

Condition	Task	Time (min)	Harms	Harm cat.	Harm subcat.
Baseline	Task 1 (no stories)	7.5 ± 4.3	1.46 ± 1.05	1.23 ± 0.73	1.38 ± 0.77
	Task 2 (baseline stories)	8.3 ± 3.6	1.31 ± 0.95	0.31 ± 0.48 (↑)	0.54 ± 0.66 (↑)
Sticky	Task 1 (no stories)	5.4 ± 2.7	1.31 ± 1.30	0.81 ± 0.66	1.06 ± 0.93
	Task 2 (sticky stories)	16.6 ± 7.4	2.31 ± 1.01	1.38 ± 0.62 (↑)	1.88 ± 0.62 (↑)

Table 4.13 Descriptive statistics for time on tasks, number of harms identified, and diversity of harms (categories and subcategories) across conditions. Values are reported as mean ± standard deviation (SD). Task 2 "Harm categories" and "Harm subcategories" indicate additional counts relative to Task 1 (↑)

	Relative time (tast 2/task 1)		Task 1 time		Task 2 time (ANCOVA)	
Source	F	p	F	p	F	p
Story condition	31.37***	<0.001	4.39*	0.047	28.47***	<0.001
Task/fairness goal order	0.60	0.445	1.01	0.324	2.48	0.130
RAI awareness score	1.07	0.313	2.13	0.158	5.11*	0.034
Championship score	0.03	0.855	0.20	0.656	2.81	0.108
Prior AI/ML experience	0.24	0.629	0.34	0.568	0.82	0.376
Task 1 time (cov.)	—	—	—	—	18.25***	0.0003

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 4.14 Two-way ANOVA/ANCOVA results for time spent on Task 1 and Task 2. F-values and p-values are reported for all sources. Partial eta squared (η_p^2) for the main Story condition effect was 0.60 for Relative time, 0.08 for Task 1 time, and 0.33 for Task 2 time. Significant effects are indicated with stars.

to those with *baseline stories*, with very large effects observed. Participants with baseline stories increased the time spent modestly over their task in the *no-story condition* (+11%, from 7.5 to 8.3 minutes), whereas participants with *sticky stories* spent nearly triple the time compared to their first task in the *no-stories condition* (+207%, from 5.4 to 16.6 minutes); see Table 4.13, Fig. 4.11.

Statistical analyses controlling for task order, experience, awareness, and championship confirm a very large effect of story type on the relative time increase between the first (no story) and second task (see Table 4.14). An additional analysis suggests that the time participants spend on the first task influences the time they spend on the second task (i.e., some participants are generally slower/more thorough

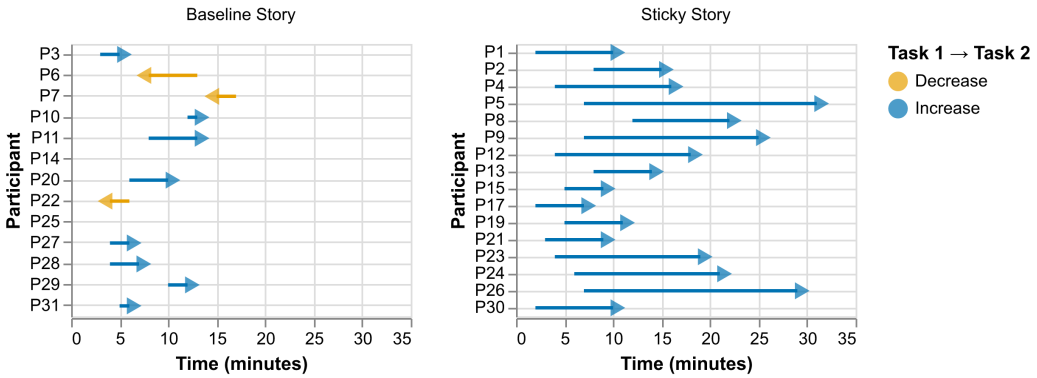


Fig. 4.11 We measured the time participants spent on the harm identification task 1 (no stories) and task 2 (baseline or sticky stories). We found that participants who read sticky stories consistently spent more time on the harm identification task, with much higher increases.

than others), but story type accounted for substantially more variance, confirming that sticky stories associated clearly with more time spent on the second task.

While we anticipated some increase in time for sticky story participants, the magnitude of the increase even over baseline stories was surprising. Qualitative examination of tool logs, think-aloud transcripts, and video recordings revealed distinct patterns of engagement. Participants with baseline stories generally skimmed the stories, often moving quickly through the task without much deliberation. In contrast, participants with sticky stories typically processed each story sequentially, critically evaluating its applicability. Many paused to reflect on past incidents, consider stakeholder perspectives, or connect the story to broader systemic issues, indicating sustained cognitive engagement rather than superficial completion. We discuss this further in *Finding 4 and 5*.

Finding 2: Practitioners identified significantly more diverse harms when sticky stories were shown. Participants generally listed small numbers of harms as result from each task (typically 0 to 5 harms, cf. Table 4.13). While we see an increase in the number of harms identified when sticky stories were provided (but not for baseline stories), we do not find counting the number of harms itself very meaningful, as participants might repeat very similar harms. Instead, we focused on the diversity of harms—measured as the number of distinct harm categories.

Source	Task 1 tax.		Task 2 tax. (ANCOVA)		Task 1 subtax.		Task 2 subtax. (ANCOVA)	
	F	p	F	p	F	p	F	p
Story condition	1.74	0.200	19.43***	0.0002	0.73	0.403	22.39***	0.0001
Task order	0.041	0.841	0.001	0.979	0.174	0.681	0.026	0.873
RAI aware score	0.057	0.813	0.663	0.424	0.000003	0.999	0.004	0.950
Champ. score	0.257	0.617	3.57	0.072	0.005	0.947	0.44	0.514
AI/ML exp.	0.119	0.734	0.000003	0.999	0.074	0.787	0.004	0.952
Task 1 tax.(cov.)	–	–	18.25***	0.0003	–	–	–	–
Task 1 subtax. (cov.)	–	–	–	–	–	–	0.018	0.896

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

Table 4.15 ANOVA and ANCOVA results for taxonomy (tax.) and sub-taxonomy (subtax.) measures. F-values and p-values are reported for all sources. Partial η_p^2 for the main Condition effect are 0.09 (Task 1 tax.), 0.49 (Task 2 tax.), 0.036 (Task 1 subtax.), and 0.54 (Task 2 subtax.).

Participants exposed to *sticky stories* identified substantially more harm categories and subcategories than those in the *baseline stories condition*, corresponding to roughly 4.5× more new categories and 3.5× more new subcategories (see Table 4.13). These differences remained statistically significant after controlling for task order, awareness, prior championship, and experience in an ANOVA, suggesting that the observed increases were robust across participant backgrounds and prior familiarity with fairness concepts.

To understand the reasons behind the differences, we analyzed participants' responses before and after exposure to the stories, alongside their recorded interactions with them. Participants in the *sticky story condition* were more likely to surface new harms and expand across categories: of the six participants (***P4***, ***P15***, ***P17***, ***P21***, ***P25***, ***P30***) who left the harms section blank in the task 1 (*no-story condition)*, all but the *baseline* participant, added harms in task 2. Several also reconsidered fairness goals they had initially dismissed. For instance, ***P1*** began by rejecting the goal of minimizing stereotyping—"To be honest, I don't think this applies to this system"—but after reading a story, exclaimed, *"Oh, man, this is topical! My wife's [country] [...] this could be true*," and added stereotyping-related harms. All participants who expanded into multiple new harm categories (***P1***, ***P2***, ***P4***, ***P13***, ***P17***, ***P21***, ***P23***, ***P30***) were in the *sticky story condition*. In some cases, *sticky stories*

also helped participants who were otherwise "stuck" in a single harm category—for instance, ***P5*** initially listed four harms exclusively under *Allocational Harms*, but in task 2, added one under Quality of Service Harms. By contrast, participants in the *baseline condition* often disengaged in task 2, leaving the section blank (***P25***, ***P27***) or typing perfunctory responses (e.g., ***P22***: "It is as was mentioned in the generated scenarios"). Even many who recorded harms (***P3***, ***P10***, ***P14***, ***P19***, ***P20***, ***P22***, ***P25***, ***P27***, ***P28***, ***P31***) largely repeated categories they had already noted, showing little expansion of perspective.

Finding 3: Practitioners critically reflected on the sticky stories more, but only skimmed the baseline stories. Based on the distribution of critical reflection codes, participants in the sticky story group engaged more deeply and reflected more often than those in the baseline group (Fig. 4.12). Among the indicators of critical reflection (Table 4.9), the most common code was *expressing surprise or enthusiasm*, observed in 17 participants overall—15 of whom were from the *sticky story condition*. Participants expressed surprise with remarks such as *"oh, wow"* (***P15***, ***P19***), *"Oh, my God, I mean, that could have caused an issue"* (***P9***), or *"I didn't really think about the fairness or responsible AI aspect of my system... I was really surprised"* (***P31***).

The second most frequent code was *connecting to the wider system or past incidents*, which also occurred more often in the *sticky story group* (9 participants) than in the baseline group (4 participants). Participants made these connections either by recalling past incidents—e.g., *"If I deploy [the biased model] then [...] it's gonna cost millions of dollars for internal systems. It happened some time back [...] [company] got shut down [...] millions of dollars of impact"* (***P23***) or anticipating wider implications.

Notably, behaviors such as *challenging assumptions* (e.g., ***P26***: *"Now that I'm looking at the word demeaning [...] there's definitely some of this happening*"), *exploring multiple perspectives* (e.g., ***P15***: "*how can a government or the FDA respond?*"), and *iterative thinking* appeared exclusively in the *sticky story group*.

In contrast, signs of shallow engagement—such as skimming or dismissing the stories, or attempting to copy them directly as harms without further reasoning—were only observed in the baseline condition (e.g., ***P7***, ***P10***, ***P25***, ***P27***, ***P28***, ***P29***).

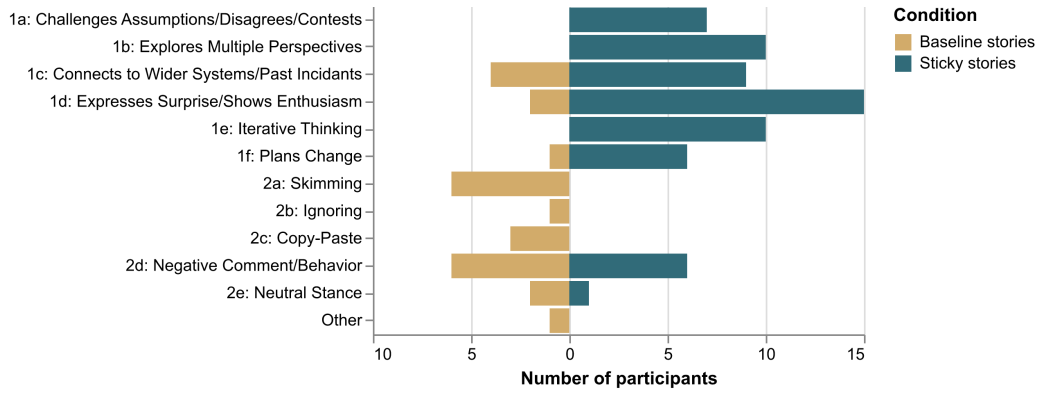


Fig. 4.12 Comparison of code frequencies per participant across baseline and sticky story conditions, highlighting increased reflection in the sticky story condition.

These participants often spent minimal time with the scenarios and focused on completing the task quickly (consistent with *Finding 1*).

Six participants across both groups voiced some negative reactions to at least one story. However, the tone and follow-up differed markedly. Baseline participants tended to be dismissive—e.g., *"I find them very general"* (*P27*)—whereas sticky story participants more often contextualized or justified their critiques. For instance, ***P26*** initially found one story *"a little far-fetched"* but immediately connected it to their own experience of harassment, and ***P5*** expressed doubt but then qualified it with technical reasoning about their system's design.

Finding 4: Follow-up survey revealed more post-study actions than the sticky story group. We sent follow-up emails to all 29 participants. Nine participants responded—six from the sticky story group (***P1***, ***P2***, ***P9***, ***P17***, ***P19***, ***P24***) and three from the baseline group (*P3*, *P14*, *P22*). While these responses are not sufficient to support statistical conclusions, we report them here for completeness and transparency.

Responses from the sticky story group consistently described concrete changes to practice, particularly in data collection and quality processes. These included recruiting more diverse participants for data collection (*P1*), conducting thorough safety checks before adding data for fine-tuning or training models using LLM-as-a-judge [405] (***P17***), enhancing data quality pipelines with expert review (***P19***), and adopting stricter labeling standards and rigorous annotation quality checks

(***P24***). Many also reported ongoing discussions with colleagues about fairness, harm mitigation, and ethical risks.

In contrast, baseline group responses were fewer and generally less action-oriented. One participant described an intention to "*use AI more responsibly*" without a concrete plan (***P3***), another reported having made no changes (***P22***), and a third (***P14***) mentioned future plans but had not yet acted, though they had shared key insights from the study with their organization's responsible AI team.

Finding 5: Practitioners exhibit distinct trajectories in shifting from initial indifference or resistance toward a more engaged stance on RAI. By examining participants' behaviors and narratives using an inductive coding approach, we posit five distinct engagement profiles. These profiles emerged from consistent patterns in how individuals responded to sticky stories, reflected on RAI issues, and engaged with harm identification tasks.

- *Resistors*: Explicitly push back against RAI as irrelevant or hype (***P17***, ***P30***).
- *Indifferents*: Acknowledge RAI is important but show little follow-through (***P2***, ***P3***, ***P5***, ***P6***, ***P21***, ***P26***).
- *Followers*: Follow RAI practices as their company enforces it (***P1***, ***P14***, ***P19***, ***P23***, ***P24***, ***P25***, ***P27***).
- *Learners*: Have limited prior knowledge of RAI, but eager to learn (***P7***, ***P8***, ***P9***, ***P10***, ***P15***, ***P20***, ***P22***, ***P28***, ***P31***).
- *Champions*: Already motivated and advocate of RAI (***P4***, ***P11***, ***P12***, ***P13***, ***P29***)

Resistors. We identified only two participants (***P17***, ***P30***) who were explicitly dismissive of RAI, describing it as hype and irrelevant to their work. This small number was not surprising, given the likelihood of *social desirability bias*—many who held similar sentiments may have instead presented themselves as *Indifferents*. Both resistors were in the sticky story group, meaning we cannot draw conclusions about how baseline participants in this category might have behaved. Notably, however, both individuals demonstrated a notable trajectory:

Outright dismissal ⟶ Skepticism ⟶ Recognition of overlooked risks ⟶ Reframing as RAI-relevant issues (expanded awareness)

This trajectory illustrates how *sticky stories* can move even resistant practitioners toward expanded awareness, with *skepticism* emerging as the stories create *disorienting dilemmas* that challenge participants' existing views.

To illustrate this trajectory, we highlight the case of ***P17***, who initially rejected RAI as unrelated to their domain, saying, "*I don't really believe in it, to be honest. In my area of research, the societal harms are very low. [. . .] I'll give the cliché example—the prison system predicting recidivism across races. But that has nothing to do with code generation.*" This perspective carried into the *no-story condition*, where they spent only two minutes and recorded no harm. The introduction of *sticky stories* shifted this pattern. Although initially skeptical—*"It's hypothetically possible, but very implausible"*—they subsequently acknowledged overlooked risks, such as bias toward non-English codebases: *"Sure, you can overfit to English. This is actually true."* By the end, ***P17*** conceded that what they had considered "just tool issues" could fall under RAI: *"I didn't know that certain things fall under RAI."* This shows how sticky stories can provoke expanded awareness even among practitioners who begin with strong resistance.

Indifferents. Indifferents are participants whose stated preference in responsible AI (RAI) contrasted sharply with their revealed preference (i.e., they rate it as important in a survey, but their described past behaviors do not suggest active engagement). We speculate that they may acknowledge RAI's importance mostly in a general sense or due to social desirability. Within this profile, we observed distinct patterns between the *sticky story* and *baseline groups*. Participants in the *baseline group* (***P3***, ***P6***) demonstrated minimal, superficial engagement, whereas those exposed to *sticky stories* (***P2***, ***P5***, ***P21***, ***P26***) engaged more deeply, reflected on the scenarios, and generated concrete, actionable plans, following the trajectory:

Indifference ⟶ Curiosity sparked ⟶ Critical reflection ⟶ Planned concrete actions

This curiosity may again stem from *disorienting dilemmas*, triggered by *diverse* (🎨) and *surprising* (😮) cases they had not anticipated.

To illustrate this trajectory, we highlight the case of ***P5***. Initially, without stories, ***P5*** spent 7 minutes on the first task, engaging mechanically and offering broad, vague assessments of potential harms, such as unreliable model performance or general stakeholder impacts. Their proposed mitigations were equally generic, such as improving system accuracy. After exposure to *sticky stories*, ***P5***'s engagement deepened substantially, spending 31 minutes carefully working through five stories (two initial and three additional ones). They revisited each story multiple times, expressed surprise and curiosity, and began considering a wider range of potential risks and systemic consequences. Through this process, ***P5*** transitioned from indifference to critical reflection, ultimately generating actionable plans they could realistically implement, showing how *sticky stories* can spark genuine engagement and meaningful planning even among initially indifferent participants.

Followers. These practitioners operate in organizations with established RAI infrastructures—mandatory harm assessment, governance teams, and formal review processes. Their engagement with RAI was often shaped more by institutional requirements than personal motivation, which often made the processes feel bureaucratic and burdensome. In the baseline group (***P14***, ***P25***, ***P27***), this compliance-driven stance generally persisted, with participants identifying only surface-level harms and routine mitigations (e.g., ***P25***: "Okay, is this the end of it?"). By contrast, we saw a noticeable shift in all participants in the sticky story group (***P1***, ***P19***, ***P23***, ***P24***): The stories encouraged them to move beyond "checking the boxes" of oversight toward deeper reflection on risks, systemic consequences, and their own role in addressing them (e.g., ***P23***: *"I wasn't expecting it to go this deep... I really liked it."*). For some, this shift translated into a new sense of ownership and proactive responsibility, as they began connecting stories to gaps in their work and articulating concrete actions they intended to take. This reflects a trajectory similar to that of the Indifferents:

Compliance mindset $\longrightarrow$ Assumptions challenged / Recognition of stake $\longrightarrow$ Critical reflection $\longrightarrow$ Proactive responsibility

We conjecture that *severity* (☠), *relevance* (◎), and *concreteness* (🪪) of harm stories are particularly important to move participants beyond their routine view of RAI.

Learners. Participants in this profile entered with only a vague awareness of Responsible AI. Many had heard of fairness concerns in the abstract, but lacked concrete exposure to what such issues actually look like in practice. Within this profile, we saw two orientations: (a) some participants did not initially recognize the gaps in their knowledge or the need to deepen it (*P7*, *P8*, *P15*, *P31*), while (b) others openly admitted limited familiarity but expressed curiosity and willingness to explore (*P9*, *P10*, *P20*, *P22*, *P28*). Although the sample size is too small for statistical claims, anecdotally, participants in the sticky story group showed a larger learning shift compared to those in the baseline group. The engagement of learners suggested the following trajectory:

> Unawareness $\longrightarrow$ Recognition of gaps (group a) / Broadening of perspective (group b) $\longrightarrow$ Proactive involvement

This mirrors the trajectory from unconscious incompetence to conscious incompetence in pegagogy literature [103]. We suspect *surprisingness* () and *diversity* () of stories is particularly important here.

Champions. While our study primarily aimed to recruit non-champions, we classified some of our participants as champions when we talked to them (*P4*, *P11*, *P12*, *P13*, *P29*). These individuals did not inherit RAI responsibilities through a formal role but still were intrinsically motivated to pursue RAI, often driving initiatives without organizational incentives. Among this group, we did *not* observe substantial differences in harm identification across conditions. In both the baseline and sticky story conditions, most of them engaged with the stories and critically reflected on them. As we expected, champions are already motivated to engage deeply with RAI assessments, whereas non-champions (, , ,) seem to need a little push to spark meaningful reflection and action.

4.3.6 Discussion

Prior research in RAI has overwhelmingly focused on supporting "champions"—practitioners already motivated to advocate for and advance RAI principles within their organizations [282, 201, 133, 257, 281]. In line with this work, we show once more that AI-generated stories can support champions in fairness assessments.

However, we also show, as our formative study and emerging HCI literature suggest [18, 245], that our baseline stories that mirror existing interventions are most effective for individuals who are already receptive or formally responsible for RAI work, but often fall short in engaging others.

Our sticky story intervention is designed to address this gap by operating at the psychological level: Aiming to capture attention, spark curiosity, and encourage critical self-examination. By shifting the focus from procedure to motivation and mindset, we envision sticky stories as a means to reframe RAI work from a bureaucratic obligation to a more vivid and memorable experience. In a way, our approach complements existing RAI tools and techniques, helping them reach and resonate with a wider range of practitioners, engaging them to a level where subsequently more traditional RAI tools or frameworks become effective.

4.3.6.1 Designing for Attitude Shifts: RAI Meets Psychology

Affecting long-term change is difficult. Past efforts to engage reluctant stakeholders have turned to nudges, gamification, or simplified documentation workflows [19, 169, 353, 354, 32]. While these strategies can improve initial uptake, some research suggests they do not reliably foster deeper shifts in mindset or sustained behavioral change [143, 414, 269]. In our work, we build on *Transformative Learning Theory* [213] and *Cognitive Dissonance Theory* [101] that suggest fundamental underlying strategies to achieve long-term change through *disorienting dilemmas* and *cognitive dissonance*—conditions necessary for critical reflection and potential transformation. While these theories explain *why* disorienting dilemmas enable transformation, *Made to Stick* complements them by explaining *how* to design such dilemmas so they reliably capture attention and remain memorable. *Sticky stories* intervene precisely at this level: they present practitioners with novel, surprising, and severe consequences grounded in their own systems, eliciting the discomfort and curiosity that precede deeper reflection.

While we cannot actually measure long-term effects with our experiments, we can observe signs of exactly the kind of mechanisms and trajectories when participants engage with sticky stories that the theories predict: Transformation, according to Transformative Learning Theory [213], is a multi-stage long journey beginning with a (a) *disorienting dilemma* that (b) *challenges prior assumptions*,

followed by (c) *self-examination* and (d) *critical assessment* of those assumptions, (e) *recognition of shared experiences* with others, (f) *exploration of new roles*, (g) *planning a course of action*, (h) *trying out new roles*, (i) *building competence* in those roles, and (j) finally *reintegration* of these changes into one's perspective and behavior. We observed participant actions aligning with (a) to (f) repeatedly and saw signs of (g), which is an encouraging sign that the underlying mechanisms for long-term change might also work here. Research also suggests that sustaining mindset change typically requires repeated reinforcement, ongoing organizational support, and integration into routine practices [380, 261], which goes beyond the scope of our research.

4.3.6.2 Who Benefits Most from the Sticky Stories, and How?

The few champions (👤+) in our study were already highly engaged with RAI tasks – they benefited from sticky stories, but were also just as motivated to work through the baseline stories and engaged deeply in the task even without stories. In contrast, non-champions (👤−, 👤, 👤, 👤) showed noticeable changes in attitudes and behaviors when interacting with sticky stories, with increases in time spent, harms identified, and critical reflection. This suggests that while existing interventions may be effective among already motivated individuals, non-champions benefit from additional prompts to trigger *disorienting dilemmas*—and sticky stories appear to provide that push.

Moreover, the observed trajectories suggest that different story qualities, such as severity and surprisingness, may resonate differently with different kinds of non-champions. *Learners* (👤) may benefit most from diverse (🎨) stories that expose them to new possibilities; *followers* (👤) may be more influenced by severe (☠) stories that highlight potential consequences; *indifferents* (👤) may respond primarily to surprising (😮) stories that challenge expectations; and *resistors* (👤−) may require a combination of surprising (😮) and relevant (🎯) stories that create both dissonance and relevance to shift their engagement. We do not imply that any single quality is sufficient—transformation may depend on multiple qualities acting together—but certain qualities may have stronger effects depending on practitioner type. This points to an important direction for future work: Systematically evaluating how individual story qualities affect different practitioner profiles.

4.3.6.3 Practical Risks and Limitations in Story Design

While sticky stories show clear benefits for attention and engagement in our experiment, they may introduce new ethical, practical, and methodological risks, and so their use must be critically examined on several fronts.

Do Ethics Need to be Extreme to Elicit Reflection? A recurring tension in our findings concerns the role of drama or emotional salience in prompting ethical reflection. Within the RAI and applied ethics literature, several scholars emphasize that meaningful ethical practice often unfolds through routine, quiet, and procedural deliberation rather than dramatic events or extreme scenarios [34, 317, 259, 201]. From this perspective, extreme interventions may not be necessary, and an overemphasis on them could risk obscuring the subtle, cumulative, and structural harms that characterize many real-world AI failures.

At the same time, research in psychology and cognitive science consistently shows that attention, memory, and sensemaking are disproportionately shaped by vividness, novelty, and emotional resonance [407, 335, 26]. Work in communication, journalism, and marketing similarly suggests that people are more likely to notice, recall, and act upon messages that contain surprising or consequential elements [126, 29, 114]. These insights raise important questions for RAI practice: In fast-paced professional environments where attention is scarce, do vivid or "dramatic" scenarios trigger effective deeper engagement or are they ultimately distracting? Reliance on extreme stories is also likely not an effective long-term strategy for regular interventions, as practitioners likely grow numb and fatigued to such stories. Ideally, extreme introductions would capture attention initially and change minds for the long term, thus enabling a nuanced and detail-oriented, and often drama-free, subsequent analysis beyond the initial stories.

Representational and Emotional Risks of Narrative. Narrative interventions raise critical questions about representation, power, and emotional impact. HCI and AI ethics scholars have long warned that narratives can unintentionally sensationalize harms, reproduce stereotypes, or reinforce dominant perspectives while obscuring more structural or less visible forms of injustice [74, 327]. Similarly, design and computing research has shown that storytelling practices can marginalize

certain voices or frame communities through deficit-oriented lenses [74]. When narrative generation is delegated to LLMs, these risks are amplified: Generative systems are known to encode and reproduce structural biases in their training data, privileging some identities, harms, and cultural frames while rendering others less visible [328, 177, 105]. As a consequence, harm stories created with LLMs may skew toward familiar tropes, overemphasize dramatic but already highly visible cases, or underrepresent the slow, infrastructural, and context-specific damages documented in sociotechnical and critical data studies [215, 107]. This pattern raises important questions about whose experience is centered or erased in AI harm assessment processes, and how automated storytelling might perpetuate these imbalances.

In our sticky story pipeline, we sought to reduce these risks by using structured prompting and deliberately steering generation away from default narrative paths—such as by using initial LLM outputs as counter-examples for "surprisingness," and systematically varying stakeholders, demographic attributes, and harm categories. This strategy broadened the narrative space and reduced surface-level repetition. Nonetheless, such approaches can only partially mitigate systemic tendencies baked into LLMs: Even with targeted prompting, generative models may still gravitate toward dominant cultural logics or overlook subtle and infrastructural harms [215, 333, 398, 130]. Therefore, we argue that embedding narrative interventions in RAI practice requires ongoing, deliberate efforts—including participatory curation, engagement with affected communities, and the development of robust ethical criteria for story inclusion [256, 136, 70, 243]—to ensure these tools do not simply reproduce, but instead challenge, dominant patterns of harm and representation. As narrative-based tools become more widespread, addressing these risks is essential to avoid reinforcing the very injustices RAI seeks to address.

The Double-Edged Sword of Severity and Concreteness. While leveraging severity and concreteness in narrative interventions can heighten attention and promote reflection, stories that stray too far from practitioners' real-world experience or drift into science fiction territory risk undermining engagement. In our user study, some participants found the consequences depicted in some sticky stories too unrealistic to take seriously. For example, P26 remarked that one story "seems a little far-fetched," and P17 described another as "very implausible...wouldn't

happen in practice." Unfortunately, the technical nature of LLMs makes it very challenging to ensure realism and avoid "hallucinations.". That is, independent of whether a story is severe or dramatic, an unrealistic story poses a risk not only to credibility, but to the perceived relevance of RAI generally.

The underlying problem is technical and hard to overcome, but an active area of LLM research. Many strategies have been explored to reduce "hallucinations" and keep generated text more realistic, such as retrieval-augmented generation (RAG) to induce real-world knowledge and context into the generation process (as done in Farsight [375]). In addition, better story generation pipelines could incorporate plausibility-checking agents, human-in-the-loop evaluation, or automated fact-checking workflows to help filter out implausible scenarios before presentation. Combining generated sticky stories with links to actual news articles, if found, or case studies might further help practitioners anchor reflection in real-world stakes, reinforcing connections to prior work on scenario-based harm assessment.

Importantly, not every story needs to perfectly map to real failures. While improving realism can help ensure narratives are accepted as relevant prompts for reflection, rather than dismissed as speculative fiction, at the same time, even less realistic stories may retain value as creative catalysts, highlighting overlooked risk categories and opening space for broader ethical consideration. In fact, we observed frequently how participants dismissed a specific story, but used it as a jumping-off point to explore a related but more realistic concern in their specific project.

Looking ahead, we see value not only in technical work, such as enhancing the realism, contextual relevance, and fact-grounding of LLM-generated narratives, but also in conceptual research to better understand how the presentation of stories to practitioners influences how they react, as well as exploring how factors like severity, concreteness, or surprisingness influence practitioner engagement and decision-making. Identifying the sweet spot between realism, attention-grabbing, and not-quite-real but useful inspiration for creative analysis will be a useful direction, requiring both HCI and ML research.

Workflow and Productivity Concerns. Our study found that integrating sticky stories into harm reflection substantially increased the time practitioners spent on these tasks. This finding echoes prior HCI and CSCW research suggesting that interventions fostering deeper engagement or reflection often come with a trade-

off in efficiency or throughput [319, 20, 311]. While deeper reflection is widely viewed as desirable in ethics and RAI [35, 146, 98, 252], in industry, incentives often align more with high-velocity production and quick deployment, which slows adoption of RAI practices to the deep frustration of RAI champions [148, 282, 4, 363]. It is not obvious which role tools like ours can play in an environment with misaligned incentives. We optimistically hope that sticky stories change minds and get developers to personally buy into RAI practices; these developers might further use the sticky stories to convince others in their team or management chain to invest into RAI practices. At that point, existing complementary approaches to supporting responsible AI practices in organizations [223, 395, 117, 174] could take over.

Overall, in recognizing these various risks, we advocate for the careful and ethically-informed integration of sticky stories into RAI practice. This calls for ongoing curation of story pipelines, ethical oversight, and empirical study of practitioner well-being and organizational impact. Stories should be paired with concrete "what now?" options, ensuring that practitioners can move from recognition to action. Importantly, sticky stories are most effective when they complement existing tools and processes—helping practitioners appreciate the significance of ethical issues across the full spectrum of ML systems.

4.4 Summary and Reflection

In this chapter, we present three interventions designed to enhance collaboration and bridge knowledge boundaries in the development of ML products. These interventions contribute both empirically grounded insights and practical mechanisms for improving collaboration across stakeholders:

- **A study of naturally occurring evaluation practices for LLM-based systems**, which surfaces 19 empirically grounded practices used by industry practitioners to evaluate LLM outputs in real-world software products.
- **A series of interventions to support the generation of meaningful explanations in ML products**, including an explainable AI policy, role-playing chatbots, and persona development education. Through iterative experiments, this work demonstrates how different forms of guidance and perspective-taking interventions can help practitioners produce explanations that better align with end-user needs.

- **A narrative-based intervention for responsible AI engagement**, which introduces *sticky stories*—structured narratives describing potential harms of ML systems—to encourage practitioners to engage more deeply with responsible AI considerations. A user study shows that these narratives increase practitioner engagement and broaden the identification of potential harms.

Taken together, these interventions demonstrate different ways of mitigating collaboration challenges in ML product development by helping align the differing mental models held by stakeholders such as data scientists, software engineers, designers, and governance teams. As summarized in Figure 4.2, the studies illustrate how collaboration challenges arise at different types of knowledge boundaries and therefore require different forms of support. The interventions explored in this chapter target the points where these perspectives must be articulated, translated, and aligned.

At the *syntactic level* (🔤), teams often lack shared terminology, evaluation criteria, or clearly defined concepts. Interventions such as defining evaluation metrics and introducing explainability policies primarily address this level by establishing common reference points for collaboration. However, the studies also show that shared terminology alone is insufficient for effective collaboration. Many challenges occur at the *semantic level* (💬), where stakeholders interpret concepts such as "performance," "explanation," or "responsible AI" differently depending on their roles and expertise. Several of the interventions therefore focus on mechanisms that support translation across perspectives—for example, chatbot role-play, persona-informed reasoning, educational guidance, and structured responsible AI assessment guides. These mechanisms help practitioners better understand how different stakeholders interpret system behavior and what information they require. The studies also surface the presence of *pragmatic boundaries* (🧑‍🤝‍🧑), where collaboration difficulties stem not only from differences in interpretation but also from differences in incentives, responsibilities, and priorities. The *sticky-story intervention* prompts practitioners to engage more actively with responsible AI considerations and supports not only shared understanding but also early steps toward negotiating how ML systems should be designed and governed in practice.

Across the three studies, two recurring intervention themes emerge. First, explicitly defining key terms, metrics, and requirements helps create a shared foundation

for collaboration by reducing ambiguity at *syntactic* and *semantic boundaries* (❶). Second, LLM-based tools—such as chatbots and harm story generation—can act as mechanisms for contextual translation, enabling practitioners to explore stakeholder perspectives and align decisions about how system implications should be managed in practice (❷). Together, these mechanisms illustrate how collaboration can be supported through processes that enable the *transfer* (🔠), *translation* (💬), and—in the case of responsible AI—initial *transformation* (🧑‍🔬) of knowledge across disciplinary boundaries.

Additionally, this chapter illustrates the value of combining multiple research approaches when studying interventions in complex socio-technical environments such as ML product development. Because collaboration practices unfold within real development contexts, understanding how interventions function requires examining them across different settings. The studies in this chapter therefore span three complementary contexts: empirical investigation of emerging practices in industry, interdisciplinary collaboration in the design of policy artifacts, and controlled experiments and user studies conducted in educational settings. Each context provides a different type of insight—industry studies reveal how practitioners adapt to challenges in real development environments, interdisciplinary collaboration supports the design of artifacts that integrate multiple perspectives, and controlled studies allow systematic examination of how practitioners respond to different forms of guidance.

Together, these approaches enable the chapter to move beyond proposing interventions toward examining how they function across different socio-technical contexts, providing a richer understanding of how collaborative mechanisms can be introduced, evaluated, and refined in the development of ML-enabled systems.

Chapter 5

Conclusion

As outlined in Chapter 1 and illustrated in Figure 1.1, my research contributions include: identifying collaboration challenges in the development of machine learning (ML) products, and designing interventions to improve collaboration across stakeholders. Together, these contributions advance our understanding of how practitioners collaborate when developing ML systems and propose practical mechanisms to bridge the knowledge boundaries that arise in this process.

5.1 Summary of Contributions

5.1.1 Identifying Collaboration Challenges in ML Product Development

The first thrust of this dissertation investigates the collaboration challenges faced by practitioners when building ML products. Through qualitative interviews with practitioners and a meta-summary of existing research, this work identifies key friction points that arise when software engineers, data scientists, governance stakeholders, and end users interact during ML product development. These challenges often emerge at knowledge boundaries, where stakeholders differ in terminology, interpretation, or priorities.

The findings reveal that ML product development introduces new forms of complexity compared to traditional software development. In particular, uncertainty in model behavior, the subjectivity of evaluation criteria, and the need to communicate

model decisions to diverse stakeholders complicate collaboration. Interpreting these challenges through the lens of *syntactic* (🔤), *semantic* (💬), and *pragmatic (🤝) knowledge boundaries* offers a structured perspective for understanding where and why collaboration challenges arise in ML systems development.

5.1.2 Designing Interventions to Improve Collaboration

The second thrust of this dissertation focuses on designing and evaluating interventions to address specific collaboration challenges identified in the first thrust.

The first intervention surfaces naturally occurring industry practices for evaluating large language model (LLM) outputs. Through a mixed-method study combining interviews and a large-scale survey, this work identifies 19 empirically grounded practices that practitioners use when evaluating LLM-based products. These practices help practitioners navigate syntactic, semantic, and pragmatic knowledge boundaries that arise when determining what constitutes acceptable model behavior and how evaluation should be conducted in real-world systems.

The second intervention focuses on explainable AI (XAI), where practitioners must generate explanations that are meaningful to different stakeholders. Through a series of iterative experiments, this work examines how different forms of guidance influence explanation quality. An explainable AI policy initially helped clarify expectations but failed to ensure explanations were meaningful to end users. Subsequent interventions introduced role-playing chatbots and persona development exercises, which improved practitioners' ability to consider stakeholder perspectives and produce explanations better aligned with end-user needs.

The third intervention addresses engagement with responsible AI (RAI) practices. Building on a formative study that revealed limited motivation among practitioners to consider ethical risks, this work introduces *sticky stories*—structured narratives describing potential harms of ML systems. A user study demonstrates that these narratives increase practitioner engagement, broaden the range of harms identified, and encourage deeper reflection on ethical implications during ML system development.

Collectively, these interventions highlight the importance of providing the right scaffolding for practitioners to navigate collaboration challenges in ML product development. Effective solutions require carefully designed structural interventions

that help stakeholders align their interpretations and decisions across knowledge boundaries, along with evaluation methods that meaningfully assess their impact in realistic development contexts.

5.2 Limitations and Risks of the Interventions

While this dissertation proposes interventions to support collaboration and responsible AI practices, it is important to acknowledge several limitations and potential risks associated with such approaches.

First, these interventions primarily operate at the level of individual practitioners and teams, and therefore may not fully account for broader organizational constraints. In practice, decisions are often shaped by incentives, deadlines, power structures, and competing priorities. As a result, even when practitioners are able to recognize issues—such as risks surfaced through interventions like sticky stories—they may not always be in a position to act on them. Prior work in organizational theory (e.g., the notion of "tempered radicals" [497]) highlights that individuals often balance advocating for change with maintaining alignment within existing systems. Consequently, the effectiveness of these interventions may depend on organizational conditions that support such actions.

Second, there is a risk that efforts to support responsible AI through tooling and automation may oversimplify complex, context-dependent decisions. While such tools can make important considerations more visible and accessible, they may also create a false sense of completeness, where practitioners rely on automated outputs without critically engaging with the underlying issues. This raises concerns about the potential misuse of such tools, where they are treated as substitutes for careful reasoning rather than as aids to support it.

Finally, by lowering the barrier to accessing and interpreting information across roles, these interventions may unintentionally reduce direct communication between stakeholders. In particular, LLM-supported mechanisms that translate and contextualize information could make it easier for practitioners to proceed without engaging in deeper discussions with others. While this may improve efficiency in some cases, it risks bypassing the rich, collaborative processes through which shared understanding is often developed.

Taken together, these limitations suggest that while the proposed interventions can meaningfully support collaboration, they are not a substitute for organizational alignment, critical reflection, and direct engagement among stakeholders.

5.3 Reflection

In this section, I reflect on the findings of this dissertation and offer several conjectures about how ML system development may evolve. These reflections extend beyond the empirical results, representing my interpretation on the broader directions suggested by this work.

Reflection 1: I conjecture that, rather than relying on full automation or fixed, prescriptive processes, the engineering of ML systems will benefit more from enabling practitioners to diagnose and correct their own assumptions, limitations, and blind spots during development. This dissertation shows that many challenges arise not from the absence of techniques, but from how they are applied in practice. Practitioners often rely on implicit assumptions about system behavior, evaluation criteria, or stakeholder needs, which remain unexamined until failures occur. As ML systems become more complex and context-dependent, these gaps become harder to detect and more consequential. Addressing them requires shifting from purely tool-centric solutions toward approaches that support reflection, perspective-taking, and self-correction during development.

§4.1: evaluation → implicit → make explicit
§4.3: RAI → ignored → make explicit

Looking ahead, I believe this raises opportunities for tools and environments that support **"helping practitioners help themselves."** Rather than automating decisions end-to-end, such systems could guide practitioners in reflecting on their choices, surfacing alternative perspectives, and identifying potential blind spots, questioning their decisions, and anticipating potential failures. This is particularly important because many of these decisions are context-dependent and, I believe, cannot be fully resolved through automation alone, at least with current approaches. In this view, the role of AI-assisted tools is not only to increase efficiency, but to augment human judgment—helping practitioners become more aware, more deliberate, and ultimately more capable in navigating complex ML development processes.

More broadly, these observations suggest the need to cultivate **T-shaped practitioners**: individuals with deep technical expertise and the ability to engage across perspectives. Education and training should therefore focus not only on building technical skills, but on helping practitioners recognize the limits of their knowledge, reason under uncertainty, and incorporate diverse viewpoints into system design. In this view, improving ML systems is inseparable from improving how practitioners think, learn, and make decisions within complex, evolving development contexts.

Reflection 2: Collaboration in ML development is often treated as a single problem, but designing effective interventions benefits from dissecting it into distinct gaps that require different responses. In this work, what initially appeared as general "collaboration challenges" became more nuanced as we examined them more closely. Similar breakdowns often had very different underlying causes (e.g., teams agreeing on a metric but still making conflicting decisions based on how they interpret its implications). In some cases, the issue was a simple mismatch in terminology; in others, it was a difference in how system behavior was understood; and in more difficult situations, it reflected deeper tensions in priorities or incentives. Treating all of these as the same kind of problem led to interventions that appeared reasonable, but consistently failed to address the underlying issue.

Our proposal of generic interventions (§4.2) or existing prior generic interventions (§4.3) did not work

This distinction is reflected in the interventions explored in this dissertation. In the explainability setting (§4.2), providing clear guidance did not consistently improve outcomes because the challenge was a 💬 *semantic boundary*—practitioners struggled to interpret what would be meaningful to different stakeholders. In contrast, in the responsible AI setting (§4.3), existing frameworks and guidelines were already available, yet remained underutilized due to a 🏋↔🏋 *pragmatic boundary*, where the issue was not understanding but motivation and action. The *"sticky story"* intervention was effective not by adding information, but by persuading practitioners to see the relevance of these concerns in their own context—and again, such an approach would likely not be effective for addressing our explanability concern (§4.2).

Taken together, these examples suggest that improving collaboration is not simply about adding more guidance, training, or tools, but about recognizing what kind of gaps or *knowledge boundaries* are present in a given situation. Some gaps require clarification, others require perspective-taking, and others still require per-

suasion. A key challenge moving forward is not only identifying that collaboration is breaking down, but understanding why—and choosing interventions that match that underlying need.

Reflection 3: Much of current ML development practices rely heavily on ad-hoc decision-making; I envision that advancing the field requires introducing the right forms of scaffolding to make these decisions more consistent and explicit. Across this dissertation, I repeatedly observed that practitioners are often required to make important decisions—such as how to define requirements, evaluate system behavior, or anticipate potential failures—without clear structure. These decisions are frequently made based on individual experience or local context, leading to variation in how similar problems are approached. Even when tools or guidelines are available, they are not consistently used, or are applied differently across teams.

Different teams kept reinventing the wheels in §4.1

The findings suggest that what is missing is not capability, but structure. In several cases, introducing relatively lightweight scaffolding—such as guiding questions, perspective-taking prompts, or concrete scenarios—helped practitioners reason more systematically about their decisions. Rather than prescribing fixed processes, these forms of scaffolding made implicit assumptions more visible and helped practitioners identify what needed to be considered in a given situation.

This points toward a shift in how ML systems are developed: I argue against fully replacing practitioner judgment—even with increasingly capable AI agents—and instead advocate for supporting practitioners with structures that make complex decisions easier to reason about, particularly as many of these decisions are context-dependent, evolving, and difficult to fully specify in advance. In this view, scaffolding helps make key decisions—around requirements, evaluation, and failure—more explicit and consistent across contexts, reducing variability in practice while still allowing flexibility where needed. More broadly, this reflects a move away from ad-hoc, intuition-driven development toward more deliberate and structured forms of ML engineering, without enforcing rigid processes.

Taken together, these reflections point toward a broader shift in how ML systems are engineered. For developers and data scientists, this means moving beyond implementing models to deciding how to define acceptable performance, interpret uncertain or ambiguous outputs, and handle edge cases where the system behaves unexpectedly. Rather than relying on more automation or additional tools alone, I conjecture that progress will depend on supporting how practitioners make decisions in systems where behavior is uncertain, evolving, and often under-specified. This includes enabling practitioners to surface and correct their own assumptions, recognize the limits of their knowledge, and adapt their approach as systems evolve. It also requires moving beyond treating collaboration as a single problem, and instead identifying the specific types of gaps that shape how stakeholders understand, interpret, and act on system behavior.

Reflection 1: self-help

Reflection 2: diagnosing boundaries

At the core of this shift is the need to make implicit decisions explicit—whether in defining requirements, evaluating outputs, or anticipating failures—and to support these decisions with the right forms of scaffolding. Because ML systems cannot be fully specified in advance, their behavior depends not only on models, but on how these decisions are made and revisited over time. Taken together, these observations point toward a form of disciplined ML engineering—not through rigid processes, but through structured support that enables practitioners to navigate uncertainty more deliberately, consistently, and effectively.

Reflection 3: right scaffolding

5.4 Knowledge Boundaries in the Era of AI-Assisted Development

The rise of generative AI tools and autonomous agents is beginning to reshape how software systems are designed and developed. Developers increasingly interact with AI systems that assist with coding, testing, documentation, and design decisions. While these tools promise substantial productivity gains, they also introduce new collaboration challenges. Many of these challenges resemble the knowledge boundaries examined throughout this dissertation, but now arise between human practitioners and AI agents that participate directly in development activities.

From this perspective, interactions between humans and AI agents can also be understood as forms of boundary crossing. Developers must interpret suggestions

generated by AI systems, assess their reliability, and integrate them into broader engineering workflows. At the same time, AI agents increasingly mediate collaboration among humans by generating code, explanations, and design artifacts that circulate across teams. As a result, collaboration in modern software development may involve not only collaboration across human roles but also collaboration with and through AI systems.

Viewing these emerging dynamics through the lens of *knowledge boundaries* suggests that many of the challenges observed in ML product development may extend naturally to human–agent collaboration. For example, developers and AI systems may lack shared terminology or representations for describing system behavior, creating challenges similar to *syntactic boundaries* (🔤). Differences in how humans interpret AI-generated outputs or recommendations may create *semantic boundaries* (💬). *Pragmatic* (🤝) tensions may arise when human judgment, organizational priorities, and AI-generated recommendations must be reconciled during development decisions.

These emerging dynamics highlight an important opportunity to extend the *knowledge boundary* perspective developed in this dissertation. As AI systems become more deeply embedded in software development processes, understanding how practitioners interpret, negotiate, and integrate AI-generated contributions will become increasingly important for ensuring reliable and accountable system design. Examining **human–agent** and **multi-agent collaboration** through this lens may therefore provide a useful foundation for understanding how future development environments can effectively augment human expertise while maintaining clarity, responsibility, and trust in the engineering process.

5.5 Future Research Directions

This dissertation shows that many failures in ML product development do not arise from model performance alone, but from the absence of disciplined engineering practices around models. Teams frequently struggle to define what a model should do, how it should be evaluated, how its behavior should be monitored, and how its decisions should be communicated to stakeholders.

At the same time, the findings of this dissertation suggest that collaboration challenges are only one manifestation of a broader gap in current ML development

practices. Many of the difficulties practitioners face stem from the lack of systematic engineering approaches for specifying, evaluating, and governing ML systems. Unlike traditional software components, whose behavior is defined by explicit rules and can be validated against expected outputs, ML models derive their behavior from data, making their behavior harder to fully specify or verify in advance.

Addressing these challenges requires advancing engineering methods that extend across the entire ML system lifecycle. Future research can therefore explore how to help **move ML development from ad-hoc experimentation toward disciplined engineering practice—guided by principled methods, structured collaboration, and a deeper understanding of human needs**.

Promising directions include improving mechanisms for defining and negotiating requirements for ML-enabled systems, developing robust evaluation and monitoring practices for modern AI systems—particularly those based on foundation models—and integrating human and organizational considerations more deeply into engineering workflows. Advancing these areas will be essential for ensuring that ML-enabled systems can be designed, evaluated, and governed reliably in real-world contexts.

5.6 Closing Remarks

Machine learning is rapidly becoming a core component of modern software systems, yet the engineering practices needed to build and govern these systems remain underdeveloped. This dissertation argues that advancing ML-enabled systems requires not only better models, but better ways for practitioners to reason, collaborate, and make decisions under uncertainty. The path forward involves advancing software engineering practices to better support ML-enabled systems, combining principled methods, collaborative processes, and human understanding—so that intelligent systems can be developed with the rigor, responsibility, and trustworthiness that society increasingly demands.

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] Akkerman, S.F. and Bakker, A. 2011. Boundary Crossing and Boundary Objects. *Review of educational research*. 81, 2 (2011), 132–169.

[3] Akkiraju, R. et al. 2020. Characterizing Machine Learning Processes: A Maturity Framework. *Business Process Management* (2020), 17–31.

[4] Ali, S.J. et al. 2023. Walking the walk of AI ethics: Organizational challenges and the individualization of risk among ethics entrepreneurs. *In Proc. 2023 ACM Conference on Fairness, Accountability, and Transparency* (2023), 217-226

[5] Ameisen, E. 2020. *Building Machine Learning Powered Applications: Going from Idea to Product*. O'Reilly Media, Inc.

[6] Amershi, S. et al. 2019. Guidelines for Human-AI Interaction. *In Proc. CHI Conf. on Human Factors in Computing Systems* (2019), 1–13.

[7] Amershi, S. et al. 2015. ModelTracker: Redesigning Performance Analysis Tools for Machine Learning. *In Proc. of 33rd Conf. on Human Factors in Computing Systems* (2015), 337–346.

[8] Amershi, S. et al. 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.

[9] Andrade, H. et al. 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.

[10] APTOS 2019 Blindness Detection: 2019. *https://www.kaggle.com/competitions/aptos2019-blindness-detection*. Accessed: 2025-01-19.

[11] Arbelaez Ossa, L. et al. 2022. Re-focusing explainability in medicine. *Digital Health*. 8, (2022).

[12] Arnold, M. et al. 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.

[13] Aroyo, L. and Welty, C. 2015. Truth is a lie: Crowd truth and the seven myths of human annotation. *AI Magazine*. 36, 1 (2015), 15–24.

[14] Arpteg, A. et al. 2018. Software Engineering Challenges of Deep Learning. *In Proc. 44th Euromicro Conf. on SEAA* (2018), 50–59.

[15] Ashmore, R. et al. 2019. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *ACM computing surveys (CSUR) 54.5 (2021), 1-39*.

[16] Baijens, J. et al. 2020. Applying Scrum in Data Science Projects. *In Proc. 22nd Conference on Business Informatics (CBI)* (2020), 30–38.

[17] Baker-Brunnbauer, J. 2021. Management perspective of ethics in artificial intelligence. *AI and ethics*. 1, 2 (2021), 173–181.

[18] Balayn, A. et al. 2023. "Fairness toolkits, A checkbox culture?" on the factors that fragment developer practices in handling algorithmic harms. *In Proc. 2023 AAAI/ACM Conference on AI, Ethics, and Society* (2023), 482–495.

[19] Ballard, S. et al. 2019. Judgment call the game: Using value sensitive design and design fiction to surface ethical concerns related to technology. *In Proc. 2019 on Designing Interactive Systems Conference* (2019), 421–433.

[20] Bardzell, J. and Bardzell, S. 2013. What is "critical" about critical design? *In Proc. SIGCHI Conference on Human Factors in Computing Systems* (2013).

[21] Barredo Arrieta, A. et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*. 58, (2020), 82–115.

[22] Basili, V. et al. 2014. GQM+strategies: A comprehensive methodology for aligning business strategies with software measurement. *arXiv [cs.SE]*.

[23] Bass, L. et al. 1998. *Software Architecture in Practice*. Addison-Wesley Longman Publishing Co., Inc.

[24] Bass, M. et al. 2009. A Coordination Risk Analysis Method for Multi-site Projects: Experience Report. *In Proc. of Int'l Conf. on Global Software Engineering* (2009), 31–40.

[25] Bäuerle, A. et al. 2022. Symphony: Composing Interactive Interfaces for Machine Learning. *In Proc. 2022 CHI Conference on Human Factors in Computing Systems* (2022), 1–14.

[26] Baumeister, R.F. et al. 2001. Bad is stronger than good. *Review of general psychology: journal of Division 1, of the American Psychological Association*. 5, 4 (2001), 323–370.

[27] Baylor, D. et al. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. *In Proc. Int'l Conf. on Knowledge Discovery and Data Mining* (2017).

[28] Begel, A. and Zimmermann, T. 2014. Analyze this! 145 questions for data scientists in software engineering. *In Proc. 36th International Conference on Software Engineering* (2014), 12–23.

[29] Berger, J. and Milkman, K.L. 2012. What makes online content viral? *JMR, Journal of marketing research*. 49, 2 (2012), 192–205.

[30] Bernardi, L. et al. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. *In Proc. 25th Int'l Conf. on ACM SIGKDD* (2019), 1743–1751.

[31] Bessen, J. et al. 2022. The cost of ethical AI development for AI startups. *In Proc. 2022 AAAI/ACM Conference on AI, Ethics, and Society* (2022), 92–106.

[32] Bhat, A. et al. 2023. Aspirations and Practice of ML Model Documentation: Moving the Needle with Nudging and Traceability. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–17.

[33] Bhatt, U. et al. 2020. Explainable machine learning in deployment. *In Proc. Conference on Fairness, Accountability, and Transparency* (2020), 648–657.

[34] Bietti, E. 2021. From ethics washing to ethics bashing: A moral philosophy view on tech ethics. *Journal of social computing*. 2, 3 (2021), 266–283.

[35] Binns, R. 2018. Fairness in Machine Learning: Lessons from Political Philosophy. *Conference on Fairness, Accountability and Transparency* (2018), 149–159.

[36] Blueprint for an AI Bill of Rights: Making Automated Systems Work for the American People: 2022. *https://www.whitehouse.gov/ostp/ai-bill-of-rights/*.

[37] Boenisch, F. et al. 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.

[38] Bogart, C. et al. 2021. When and how to make breaking changes: Policies and practices in 18 open source software ecosystems. *ACM Transactions on Software Engineering and Methodology*. 30, 4 (2021), 1–56.

[39] Bogucka, E. et al. 2024. Co-designing an AI impact assessment report template with AI practitioners and AI compliance experts. *arXiv [cs.HC]*.

[40] Borch, C. 2022. Machine learning, knowledge risk, and principal-agent problems in automated trading. *Technology in society*. 68, (2022), 101852.

[41] Boren, T. and Ramey, J. 2000. Thinking aloud: reconciling theory and practice. *IEEE transactions on professional communication*. 43, 3 (2000), 261–278.

[42] Borg, M. et al. 2019. Safely Entering the Deep: A Review of Verification and Validation for Machine Learning and a Challenge Elicitation in the Automotive Industry. *Journal of Automotive Software Engineering*. 1, 1 (2019), 1–9.

[43] Bosch, J. et al. 2021. Engineering AI Systems: A Research Agenda. *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*. IGI Global. 1–19.

[44] Boujut, J.-F. and Blanco, E. 2003. Intermediary Objects as a Means to Foster Cooperation in Engineering Design. *Computer supported cooperative work: CSCW: an international journal*. 12, 2 (2003), 205–219.

[45] Boyd, K. 2022. Designing up with value-sensitive design: Building a field guide for ethical ML development. *In Proc. 2022 ACM Conference on Fairness, Accountability, and Transparency* (2022), 2069–2082.

[46] 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.

[47] Braiek, H.B. and Khomh, F. 2020. On testing machine learning programs. *The Journal of systems and software*. 164, (2020), 110542.

[48] Brandstädter, S. and Sonntag, K. 2016. Interdisciplinary Collaboration. *Advances in Ergonomic Design of Systems, Products and Processes* (2016), 395–409.

[49] Braude, E.J. and Bernstein, M.E. 2016. *Software Engineering: Modern Approaches, Second Edition*. Waveland Press.

[50] Breck, E. et al. 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.

[51] Brennen, A. 2020. What Do People Really Want When They Say They Want "Explainable AI?" We Asked 60 Stakeholders. *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (2020), 1–7.

[52] Brookfield, S.D. 2017. *Becoming a critically reflective teacher*. Jossey-Bass.

[53] Brown, G.F.C. 1995. *Factors that facilitate or inhibit interdisciplinary collaboration within a professional bureaucracy*. University of Arkansas.

[54] Buçinca, Z. et al. 2023. AHA!: Facilitating AI Impact Assessment by Generating Examples of Harms. *arXiv [cs.HC]*.

[55] Cai, C.J. et al. 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.

[56] Campbell, S. et al. 2020. Purposive sampling: complex or simple? Research case examples. *Journal of Research in Nursing*. 25, 8 (2020), 652–661.

[57] Carlile, P.R. 2002. A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development. *Organization Science*. 13, 4 (2002), 442–455.

[58] Carlile, P.R. 2004. Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries. *Organization Science*. 15, 5 (2004), 555–568.

[59] Cataldo, M. et al. 2006. Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools. *In Proc. of Conf. Computer Supported Cooperative Work (CSCW)* (2006), 353–362.

[60] Challener, D.W. et al. 2025. Flesch-Kincaid Grade Level readability scores to evaluate readability of clinical documentation during an electronic health record transition. *Advances in Health Information Science and Practice*. 1, 1 (2025).

[61] Chang, J. and Custis, C. 2022. Understanding Implementation Challenges in Machine Learning Documentation. In Proc. 2nd ACM Conf. *Equity and Access in Algorithms, Mechanisms, and Optimization* (2022), 1–8.

[62] Chattopadhyay, S. et al. 2020. What's Wrong with Computational Notebooks? Pain Points, Needs, and Design Opportunities. *In Proc. of CHI Conf. on Human Factors in Computing Systems* (2020), 1–12.

[63] Chen, B. et al. 2025. Unleashing the potential of prompt engineering for large language models. *Patterns (New York, N.Y.)*. 6, 6 (2025).

[64] Cheng, D. et al. 2018. Manifesting Bugs in Machine Learning Code: An Explorative Study with Mutation Testing. *In Proc. of Int'l Conf. on Software Quality, Reliability and Security (QRS)* (2018), 313–324.

[65] Chen, Z. et al. 2020. Understanding Challenges in Deploying Deep Learning Based Software: An Empirical Study. *Journal of Environmental Sciences (China) English Ed.*.

[66] Chotisarn, N. et al. 2020. A systematic literature review of modern software visualization. *Journal of visualization / the Visualization Society of Japan*. 23, 4 (2020), 539–558.

[67] Claessen, K. and Hughes, J. 2011. QuickCheck. *SIGPLAN Notices*. 46, 4 (2011), 53–64.

[68] Colaner, N. 2022. Is explainable artificial intelligence intrinsically valuable? *AI & society*. 37, 1 (2022), 231–238.

[69] Collins, K.M.T. 2010. Advanced sampling designs in mixed research: Current practices and emerging trends in the social and behavioral sciences. *SAGE Handbook of Mixed Methods in Social & Behavioral Research*. SAGE Publications, Inc. 353–378.

[70] Commons-based Data Set Governance for AI: *https://openfuture.eu/publication/commons-based-data-set-governance-for-ai*. Accessed: 2025-12-05.

[71] Conway, M.E. 1968. How Do Committees Invent? *Datamation*. 14, 4 (1968), 28–31.

[72] Cooper, A. 2004. *The inmates are running the asylum*. Sams Publishing.

[73] Cossette, B.E. and Walker, R.J. 2012. Seeking the Ground Truth: A Retroactive Study on the Evolution and Migration of Software Libraries. *In Proc. of Int'l Symposium Foundations of Software Engineering (FSE)* (2012), 1–11.

[74] Costanza-Chock, S. 2020. *Design Justice: Community-Led Practices to Build the Worlds We Need*. MIT Press.

[75] Curtis, B. et al. 1988. A field study of the software design process for large systems. *Communications of the ACM*. 31, 11 (1988), 1268–1287.

[76] Dabbish, L. et al. 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.

[77] Davis, J. and Daniels, R. 2016. *Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale*. O'Reilly Media, Inc.

[78] Deng, W.H. et al. 2022. Exploring how machine learning practitioners (try to) use fairness toolkits. *In Proc. 2022 ACM Conference on Fairness, Accountability, and Transparency* (2022), 473–484.

[79] Deng, W.H. et al. 2023. Investigating practices and opportunities for cross-functional collaboration around AI fairness in industry practice. *In Proc. 2023 ACM Conference on Fairness, Accountability, and Transparency.* (2023), 705–716.

[80] Deng, W.H. et al. 2025. Supporting industry computing researchers in assessing, articulating, and addressing the potential negative societal impact of their work. *In Proc. ACM on human-computer interaction*. 9, 2 (2025), 1–37.

[81] Deng, W.H. et al. 2023. Understanding Practices, Challenges, and Opportunities for User-Engaged Algorithm Auditing in Industry Practice. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–18.

[82] Dignum, V. 2019. *Responsible artificial intelligence: how to develop and use AI in a responsible way*. Springer International Publishing.

[83] Dilhara, M. et al. 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.

[84] Dixit, T. et al. 2024. RETAIN: Interactive tool for REgression Testing guided LLM migrAtIoN. *arXiv [cs.IR]*.

[85] Dolata, M. et al. 2024. Development in times of hype: How freelancers explore Generative AI? *In Proc. IEEE/ACM 46th International Conference on Software Engineering* (2024), 1–13.

[86] Dominique, B. et al. 2023. FactSheets for hardware-aware AI models: A case study of analog in memory computing AI models. *In Proc. 2023 IEEE International Conference on Software Services Engineering (SSE)* (2023), 148–158.

[87] Dove, G. et al. 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.

[88] Dual-coding theory: 2025. *https://en.wikipedia.org/wiki/Dual-coding_theory*.Accessed: 2025-01-19.

[89] Ehsan, U. et al. 2024. Seamful XAI: Operationalizing seamful design in Explainable AI. *In Proc. ACM on human-computer interaction*. 8, CSCW1 (2024), 1–29.

[90] Elsayed-Ali, S. et al. 2023. Responsible & inclusive cards: An online card tool to promote critical reflection in technology industry work practices. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–14.

[91] Epperson, W. et al. 2022. Strategies for Reuse and Sharing among Data Scientists in Software Teams. *In Proc. 44th International Conference on Software Engineering: Software Engineering in Practice* (2022), 243–252.

[92] Ericsson, K.A. and Simon, H.A. 1993. *Protocol Analysis*. MIT Press.

[93] Espinosa, J.A. et al. 2007. Team Knowledge and Coordination in Geographically Distributed Software Development. *Journal of Management Information Systems*. 24, 1 (2007), 135–169.

[94] Evolving Microsoft Security Development Lifecycle (SDL): How continuous SDL can help you build more secure software: *microsoft.com/evolving-microsoft-security-development-lifecycle.* Accessed: 2025-01-19.

[95] Faan, M.S.P. and Aprn, J.B.P. 2006. *Handbook for Synthesizing Qualitative Research*. Springer Publishing Company.

[96] 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/*. Accessed: 2025-01-19.

[97] Falessi, D. et al. 2018. Empirical software engineering experts on the use of students and professionals in experiments. *Empirical Software Engineer*. 23, 1 (2018), 452–489.

[98] F. Clancy, R. et al. 2025. Exploring AI ethics in global contexts: a culturally responsive, psychologically realist approach. *AI and ethics*. 5, 6 (2025), 6329–6338.

[99] Feldt, R. et al. 2018. Four commentaries on the use of students and professionals in empirical software engineering experiments. *Empirical Software Engineer*. 23, 6 (2018), 3801–3820.

[100] Felizardo, K.R. et al. 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.

[101] Festinger, L. 1957. *A Theory of Cognitive Dissonance*. Stanford University Press.

[102] Orban, R. 2016. Bridging the Gap Between Data Science & Engineer: Building High-Performance Teams. *https://www.slideshare.net/slideshow/bridging-the-gap-between-data-science-engineer-building-highperformance-teams/57526374*. Accessed: 2025-01-19.

[103] Four stages of competence: 2025. *https://en.wikipedia.org/wiki/Four_stages_of_competence*. Accessed: 2025-01-19.

[104] Fredricks, J.A. et al. 2004. School engagement: Potential of the concept, state of the evidence. *Review of educational research*. 74, 1 (2004), 59–109.

[105] Gallegos, I.O. et al. 2024. Bias and fairness in large language models: A survey. *Computational linguistics (Association for Computational Linguistics)*. (2024), 1–83.

[106] Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence: *gartner.com/gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence*. Accessed: 2025-01-19.

[107] Ghosh, S. et al. 2024. Do generative AI models output harm while representing non-Western cultures: Evidence from A community-centered approach. *arXiv [cs.CY]*.

[108] Gierczyk, M. et al. 2024. The snowball sampling strategy in the field of social sciences. Contexts and considerations. *Przegląd Badań Edukacyjnych*. 2, 43 (2024), 87–104.

[109] Giray, G. 2021. A Software Engineering Perspective on Engineering Machine Learning Systems: State of the Art and Challenges. *Journal of Systems and Software*. 180, (2021), 111031.

[110] Golendukhina, V. et al. 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.

[111] Gonzalez, D. et al. 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.

[112] Gravett, S. 2002. Transformative Learning through Action Research: A Case Study from South Africa. *Adult Education Research Conference* (2002).

[113] Green, M.C. and Appel, M. 2024. Narrative transportation: How stories shape how we see ourselves and the world. *Advances in Experimental Social Psychology*. Elsevier. 1–82.

[114] Green, M.C. and Brock, T.C. 2000. The role of transportation in the persuasiveness of public narratives. *Journal of personality and social psychology*. 79, 5 (2000), 701–721.

[115] Haakman, M. et al. 2021. AI Lifecycle Models Need To Be Revised. An Exploratory Study in Fintech. *Empirical Software Engineering*. 26, 5 (2021), 1–29.

[116] Hadi Mogavi, R. et al. 2022. When gamification spoils your learning: A qualitative case study of gamification misuse in a language-learning app. *In Proc. Ninth ACM Conference on Learning @ Scale* (2022).

[117] Hadley, E. et al. 2025. Investigating algorithm review boards for organizational responsible artificial intelligence governance. *AI and ethics*. 5, 3 (2025), 2485–2495.

[118] Hampton, D.L. 2018. Letter to Jyri Leskela, Optomed Oy.

[119] Hanington, B. and Martin, B. 2019. *Universal Methods of Design Expanded and Revised: 125 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions*. Rockport Publishers.

[120] Han, S. *awesome-llmops: Awesome series for LLMOps*. Github. *https://github.com/tensorchord/Awesome-LLMOps*. Accessed: 2025-01-19.

[121] Harris, J. 2020. Beyond the jupyter notebook: how to build data science products. *Towards Data Science*. Accessed: 2025-01-19.

[122] Harsh, S. 2011. Purposeful Sampling in Qualitative Research Synthesis. *Qualitative Research Journal*. 11, 2 (2011), 63–75.

[123] Hassan, A.E. et al. 2024. Rethinking software engineering in the foundation model era: From task-driven AI copilots to goal-driven AI pair programmers. *arXiv [cs.SE]*.

[124] Hazelwood, K. et al. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. *In Proc. 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)* (2018), 620–629.

[125] Head, A. et al. 2019. Managing messes in computational notebooks. *In Proc. of CHI Conf. on Human Factors in Computing Systems* (2019).

[126] Heath, D. and Heath, C. 2009. *Made to Stick*. Random House Trade.

[127] Henry, K.E. et al. 2022. Human-machine teaming is key to AI adoption: clinicians' experiences with a deployed machine learning system. *NPJ digital medicine*. 5, 1 (2022), 1–6.

[128] Herbsleb, J.D. and Grinter, R.E. 1999. Splitting the Organization and Integrating the Code: Conway's Law Revisited. *In Proc. of Int'l Conf. Software Engineering (ICSE)* (1999), 85–95.

[129] Hermann, J. and Del Balso, M. 2017. Meet Michelangelo: Uber's machine learning platform. *https://www.uber.com/blog/michelangelo-machine-learning-platform/*. Accessed: 2025-01-19.

[130] Hida, R. et al. 2025. Social bias evaluation for large language models requires prompt variations. *Findings of the Association for Computational Linguistics: EMNLP 2025* (2025), 14507–14530.

[131] Hill, C. et al. 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.

[132] Hohman, F. et al. 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.

[133] Holstein, K. et al. 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.

[134] 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.

[135] Hsieh, H.-F. and Shannon, S.E. 2005. Three approaches to qualitative content analysis. *Qualitative health research*. 15, 9 (2005), 1277–1288.

[136] Hsu, Y.-C. et al. 2022. Empowering local communities using artificial intelligence. *Patterns*. 3, 3 (2022), 100449.

[137] Huang, X. et al. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*. 37, (2020), 100270.

[138] Huang, X. et al. 2018. Synthesizing qualitative research in software engineering: a critical review. *In Proc. 40th International Conference on Software Engineering* (2018), 1207–1218.

[139] Hudson, W. 2013. Card Sorting. *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* The Interaction Design Foundation.

[140] Hukkelberg, I. and Rolland, K. 2020. Exploring Machine Learning in a Large Governmental Organization: An Information Infrastructure Perspective. *European Conference on Information Systems*. (2020).

[141] Hulten, G. 2019. *Building Intelligent Systems: A Guide to Machine Learning Engineering*. Apress.

[142] Humbatova, N. et al. 2020. Taxonomy of real faults in deep learning systems. *In Proc. 42nd Int'l Conf. on Software Engineering (ICSE)* (2020), 1110-1121

[143] Hummel, D. and Maedche, A. 2019. How effective is nudging? A quantitative review on the effect sizes and limits of empirical nudging studies. *Journal of behavioral and experimental economics*. 80, (2019), 47–58.

[144] Hummer, W. et al. 2019. ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI. *In Proc. 2019 IEEE International Conference on Cloud Engineering (IC2E)* (2019), 113–120.

[145] Hynes, N. et al. 2017. The data linter: Lightweight, automated sanity checking for ml data sets. *NIPS MLSys Workshop*. 1, (2017), 5.

[146] Ibitoye, A.O. et al. 2025. Rethinking responsible AI from ethical pillars to sociotechnical practice. *AI and ethics*. 5, 6 (2025), 6207–6223.

[147] 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.

[148] Işık, Ö. and Goswami, A. 2025. The Three Obstacles Slowing Responsible AI. *MIT Sloan Management Review*. (2025).

[149] Itti, L. and Baldi, P. 2009. Bayesian surprise attracts human attention. *Vision research*. 49, 10 (2009), 1295–1306.

[150] 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.

[151] Jentzsch, S. and Hochgeschwender, N. 2021. A qualitative study of Machine Learning practices and engineering challenges in Earth Observation. *it - Information Technology*. 63, 4 (2021), 235–247.

[152] Jeong, C. 2023. A study on the implementation of generative AI services using an enterprise data-based LLM application architecture. *arXiv [cs.AI]*.

[153] Jiang, E. et al. 2022. PromptMaker: Prompt-based prototyping with large language models. *CHI Conference on Human Factors in Computing Systems Extended Abstracts* (2022), 1-8.

[154] John, M.M. et al. 2020. AI Deployment Architecture: Multi-Case Study for Key Factor Identification. *In Proc. 27th Asia-Pacific Software Engineering Conference (APSEC)* (2020), 395–404.

[155] John, M.M. et al. 2021. Towards MLOps: A Framework and Maturity Model. *In Proc. 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2021), 1–8.

[156] Kallina, E. et al. 2025. Stakeholder participation for responsible AI development: Disconnects between guidance and current practice. *In Proc. 2025 ACM Conference on Fairness, Accountability, and Transparency* (2025), 1060–1079.

[157] Kamoi, R. et al. 2024. Evaluating LLMs at detecting errors in LLM responses. *arXiv [cs.CL]*.

[158] Kanagal, B. and Tata, S. 2018. Recommendations for All: Solving Thousands of Recommendation Problems Daily. *In Proc. 34th International Conference on Data Engineering (ICDE)* (2018), 1404–1413.

[159] Kandel, S. et al. 2012. Enterprise data analysis and visualization: An interview study. *IEEE transactions on visualization and computer graphics*. 18, 12 (2012), 2917–2926.

[160] Kang, D. et al. 2020. Model Assertions for Monitoring and Improving ML Models. *In Proc. Machine Learning and Systems 2 (2020): 481-496.*.

[161] Kästner, C. and Kang, E. 2020. Teaching Software Engineering for Al-Enabled Systems. *In Proc. 42nd Int'l Conf. on Software Engineering: Software Engineering Education and Training (ICSE-SEET)* (2020), 45–48.

[162] Katal, A. et al. 2019. DevOps: Bridging the gap between Development and Operations. *In Proc. 3rd International Conference on Computing Methodologies and Communication (ICCMC)* (2019), 1–7.

[163] Kaur, H. et al. 2024. Interpretability gone bad: The role of bounded rationality in how practitioners understand machine learning. *In Proc. ACM on human-computer interaction*. 8, CSCW1 (2024), 1–34.

[164] Keele, S. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Rep., Ver. 2.3 EBSE Tech. Report. EBSE.

[165] Khattab, O. et al. 2023. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv [cs.CL]*.

[166] Kihlstrom, J.F. 2021. Ecological validity and "ecological validity." *Perspectives on psychological science: a journal of the Association for Psychological Science*. 16, 2 (2021), 466–471.

[167] Kim, M. et al. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering*. 44, 11 (2018), 1024–1038.

[168] Kim, M. et al. 2016. The emerging role of data scientists on software development teams. *In Proc. 38th International Conference on Software Engineering* (2016), 96–107.

[169] Kim, S.-E. et al. 2025. Dilemmas in AI ethics: A digital game for moral reasoning and collective decision-making. *In Proc. International Conference on Artificial Intelligence in Education* (Cham, 2025), 434–447.

[170] Kim, T.S. et al. 2024. EvalLM: Interactive evaluation of large language model prompts on user-defined criteria. *In Proc. CHI Conference on Human Factors in Computing Systems* (2024), 1–21.

[171] Königstorfer, F. and Thalmann, S. 2022. AI Documentation: A path to accountability. *Journal of Responsible Technology*. 11, (2022), 100043.

[172] Kumar, R.S.S. et al. 2020. Adversarial Machine Learning - Industry Perspectives. *In Proc. IEEE Security and Privacy Workshops (SPW).* (2020), 69–75.

[173] Laato, S. et al. 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.

[174] Laine, J. et al. 2024. Ethics-based AI auditing: A systematic literature review on conceptualizations of ethical principles and knowledge contributions to stakeholders. *Information & management*. 61, 5 (2024), 103969.

[175] Lakshmanan, V. et al. 2020. *Machine Learning Design Patterns*. O'Reilly Media, Inc.

[176] Lanne, M. et al. 2025. Organisational tensions in introducing socially sustainable AI. *AI & society*. (2025). DOI:https://doi.org/10.1007/s00146-025-02293-y.

[177] Large Language Models generate biased content, warn researchers: 2024. *https://www.ucl.ac.uk/news/2024/apr/large-language-models-generate-biased-content-warn-researchers*. Accessed: 2025-12-05.

[178] Lazar, J. et al. 2017. *Research Methods in Human-Computer Interaction*. Morgan Kaufmann.

[179] Lebovitz, S. et al. 2022. To engage or not to engage with AI for critical judgments: How professionals deal with opacity when using AI for medical diagnosis. *Organization science*. 33, 1 (2022), 126–148.

[180] Leveson, N.G. 2016. *Engineering a safer world: Systems thinking applied to safety*. The MIT Press.

[181] Lewis, G.A. et al. 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.

[182] Lewis, G.A. et al. 2021. Software Architecture Challenges for ML Systems. *In Proc. International Conference on Software Maintenance and Evolution (ICSME)* (2021), 634–638.

[183] Lewis, P. et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv [cs.CL]*.

[184] Liang, J.T. et al. 2024. Prompts are programs too! Understanding how developers build software containing prompts. *arXiv [cs.SE]*.

[185] Liao, Q.V. et al. 2023. Designerly understanding: Information needs for model transparency to support design ideation for AI-powered user experience. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–21.

[186] 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.

[187] Li, P.L. et al. 2017. Cross-Disciplinary Perspectives on Collaborations with Software Engineers. *In Proc. of 10th Int'l Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)* (2017), 2–8.

[188] Li, S. et al. 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.

[189] Liu, H. et al. 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.

[190] Liu, J. et al. 2020. Understanding the Role of Alternatives in Data Analysis Practices. *IEEE transactions on visualization and computer graphics*. 26, 1 (2020), 66–76.

[191] Liu, Q. et al. 2018. A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View. *IEEE Access*. 6, (2018), 12103–12117.

[192] Liu, V. and Chilton, L.B. 2022. Design guidelines for prompt engineering text-to-image generative models. *CHI Conference on Human Factors in Computing Systems* (2022), 1-23.

[193] 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.

[194] Louviere, J.J. et al. 2014. *Stated choice methods*. Cambridge University Press.

[195] Lundberg, S.M. and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. *Advances in Neural Information Processing Systems (NIPS)*. 30, (2017).

[196] 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.

[197] Lwakatare, L.E. et al. 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.

[198] Lwakatare, L.E. et al. 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*. 127, (2020).

[199] Madaio, M. et al. 2022. Assessing the fairness of AI systems: AI practitioners' processes, challenges, and needs for support. *In Proc. ACM on human-computer interaction*. 6, CSCW1 (2022), 1–26.

[200] Madaio, M. et al. 2024. Learning about responsible AI on-the-job: Learning pathways, orientations, and aspirations. *In Proc. 2024 ACM Conference on Fairness, Accountability, and Transparency* (2024), 1544–1558.

[201] Madaio, M.A. et al. 2020. Co-Designing Checklists to Understand Organizational Challenges and Opportunities around Fairness in AI. *In Proc. 2020 CHI Conference on Human Factors in Computing Systems* (2020), 1–14.

[202] Mahanti, R. 2019. *Data Quality: Dimensions, Measurement, Strategy, Management, and Governance*. Quality Press.

[203] Mäkinen, S. et al. 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.

[204] Mann, K. et al. 2009. Reflection and reflective practice in health professions education: a systematic review. *Advances in health sciences education: theory and practice*. 14, 4 (2009), 595–621.

[205] Martínez-Fernández, S. et al. 2022. Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology*. 31, 2 (2022), 1–59.

[206] Martinez-Plumed, F. et al. 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.

[207] Massaro, D.W. et al. 1988. Communication and Persuasion: Central and Peripheral Routes to Attitude Change. *The American journal of psychology*. 101, 1 (1988), 155.

[208] McGlohon, M. 2021. Demystifying Machine Learning in Production: Reasoning about a Large-Scale ML Platform.

[209] McGraw, G. et al. 2020. An architectural risk analysis of machine learning systems: Toward more secure machine learning. *Berryville Institute of Machine Learning*. 23, (2020).

[210] Menon, A.V. et al. 2024. Lessons from clinical communications for explainable AI. *In Proc. AAAI/ACM Conference on AI, Ethics, and Society*. 7, (2024), 958–970.

[211] Meyer, B. 1997. *Object-Oriented Software Construction*. Prentice-Hall.

[212] Mezirow, J. 2000. *Learning as transformation: Critical perspectives on a theory in progress*. Jossey-Bass.

[213] Mezirow, J. 1991. *Transformative dimensions of adult learning*. Jossey-Bass.

[214] Mezirow, J. 2018. Transformative learning theory. *Contemporary Theories of Learning*. Routledge. 114–128.

[215] Mickel, J. et al. 2025. More of the same: Persistent representational harms under increased representation. *arXiv [cs.CL]*.

[216] Microsoft 2022. *Microsoft responsible ai standard v2*. Microsoft.

[217] Microsoft RAI Impact Assessment Template: *https://blogs.microsoft.com/wp-content/uploads/prod/sites/5/2022/06/Microsoft-RAI-Impact-Assessment-Template.pdf*. Accessed: 2025-01-19.

[218] Mishra, A. et al. 2023. PromptAid: Prompt exploration, perturbation, testing and iteration using visual analytics for large Language Models. *arXiv [cs.HC]*.

[219] Mistrík, I. et al. 2010. *Collaborative Software Engineering*. Springer Science & Business Media.

[220] Mitchell, M. et al. 2019. Model Cards for Model Reporting. *In Proc. Conf. on Fairness, Accountability, and Transparency* (2019), 220–229.

[221] MIT Finds 95% Of GenAI Pilots Fail Because Companies Avoid Friction: 2025. *https://www.forbes.com/sites/jasonsnyder/2025/08/26/mit-finds-95-of-genai-pilots-fail-because-companies-avoid-friction/*. Accessed: 2026-02-20.

[222] ml-practical-usecases: A database of 450 Machine Learning (ML) system design case studies from 100+ companies: *https://github.com/mallahyari/ml-practical-usecases*. Accessed: 2025-09-09.

[223] Mokander, J. and Floridi, L. 2024. Operationalising AI governance through ethics-based auditing: An industry case study. *AI and Ethics 3.2 (2023): 451-468*.

[224] Molnar, C. 2020. *Interpretable Machine Learning*. Lulu.com.

[225] Muiruri, D. et al. 2022. Practices and Infrastructures for ML Systems–An Interview Study in Finnish Organizations. *TechRxiv*.

[226] Muller, M. et al. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. *In Proc. 2019 CHI Conference on Human Factors in Computing Systems* (2019), 1–15.

[227] Murphy, N.R. et al. 2016. *Site reliability engineering*. O'Reilly Media.

[228] Murugadoss, B. et al. 2024. Evaluating the evaluator: Measuring LLMs' adherence to task evaluation instructions. *arXiv [cs.AI]*.

[229] Myllyaho, L. et al. 2022. On misbehaviour and fault tolerance in machine learning systems. *Journal of Systems and Software*. 183, (2022), 111096.

[230] Nahar, N. et al. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. *Proc. 44th Int'l Conf. on Software Engineering* (2022), 413–425.

[231] Nahar, N. et al. 2023. A Dataset and Analysis of Open-Source Machine Learning Products. *arXiv [cs.SE]*.

[232] Nahar, N. et al. 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.

[233] Nahar, N. et al. 2025. Beyond the comfort zone: Emerging solutions to overcome challenges in integrating LLMs into software products. *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (2025), 516–527.

[234] Nahar, N. et al. 2024. Regulating Explainability in Machine Learning Applications – Observations from a Policy Design Experiment. *In Proc. 2024 ACM Conference on Fairness, Accountability, and Transparency* (2024), 2101–2112.

[235] Namvar, M. et al. 2022. Beyond effective use: Integrating wise reasoning in machine learning development. *International journal of information management*. (2022), 102566.

[236] Nathan, L.P. et al. 2008. Envisioning systemic effects on persons and society throughout interactive system design. *In Proc. 7th ACM conference on Designing interactive systems* (2008), 1–10.

[237] Nehra, M. Top 10 Programming Languages for Desktop Apps in 2022. Decipher Zone.

[238] Ng, E. 2022. Letter to John Smith, AEYE Health, Inc.

[239] Ng, E. 2020. Letter to Kaushal Solanki, Eyenuk, Inc.

[240] Nikanjam, A. et al. 2022. Faults in deep reinforcement learning programs: a taxonomy and a detection approach. *Automated software engineering*. 29, 1 (2022).

[241] Nikhil, K. et al. 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.

[242] Nushi, B. et al. 2017. On human intellect and machine failures: troubleshooting integrative machine learning systems. *In Proc. Thirty-First AAAI Conference on Artificial Intelligence* (2017), 1017–1025.

[243] Odhiambo, J.M. and Ondimu, K. 2025. A framework for ethical AI-generated content governance. *Preprints*.

[244] 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).

[245] Omar, Z.A. et al. 2025. Beyond Accuracy, SHAP, and Anchors – On the difficulty of designing effective end-user explanations. *arXiv [cs.HC]*.

[246] Onwuegbuzie, A.J. and Burke Johnson, R. eds. 2021. *The Routledge reviewer's guide to mixed methods analysis*. Routledge.

[247] OpenAI et al. 2023. GPT-4 Technical Report. *arXiv [cs.CL]*.

[248] Ovaska, P. et al. 2003. Architecture as a coordination tool in multi-site software development. *Software Process Improvement and Practice*. 8, 4 (2003), 233–247.

[249] Ozkaya, I. 2020. What Is Really Different in Engineering AI-Enabled Systems? *IEEE Software*. 37, 4 (2020), 3–6.

[250] Paleyes, A. et al. 2022. Challenges in deploying machine learning: A survey of case studies. *ACM computing surveys*. (2022).

[251] Pang, R.Y. et al. 2024. BLIP: Facilitating the exploration of undesirable consequences of digital technologies. *In Proc. CHI Conference on Human Factors in Computing Systems* (2024), 1–18.

[252] Pant, A. et al. 2023. Ethics in the Age of AI: An Analysis of AI Practitioners' Awareness and Challenges. *ACM Transactions on Software Engineering and Methodology*. (2023).

[253] Park, S. et al. 2021. Facilitating Knowledge Sharing from Domain Experts to Data Scientists for Building NLP Models. *In Proc. 26th International Conference on Intelligent User Interfaces*, 2021

[254] Parnas, D.L. 1972. On the Criteria to be used in Decomposing Systems into Modules. *Communications of the ACM*. 15, 12 (1972), 1053–1058.

[255] Parnin, C. et al. 2025. Building your own product copilot: Challenges, opportunities, and needs. *2025 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)* (2025), 338–348.

[256] Parthasarathy, A. et al. 2024. Participatory approaches in AI development and governance: A principled approach. *arXiv [cs.CY]*.

[257] Passi, S. and Barocas, S. 2019. Problem Formulation and Fairness. *In Proc. Conference on Fairness, Accountability, and Transparency* (2019), 39–48.

[258] 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.

[259] Passi, S. and Sengers, P. 2020. Making data science systems work. *Big Data & Society*. 7, 2 (2020).

[260] Patel, K. et al. 2008. Investigating statistical machine learning as a tool for software development. *In Proc. of SIGCHI Conf. on Human Factors in Computing Systems* (2008), 667–676.

[261] Paunesku, D. et al. 2015. Mind-set interventions are a scalable treatment for academic underachievement. *Psychological science*. 26, 6 (2015), 784–793.

[262] People + AI Guidebook: 2019. *https://pair.withgoogle.com/guidebook/*. Accessed: 2024-07-09.

[263] Perspective API: Using Machine Learning to Reduce Toxicity Online: 2017. *https://www.perspectiveapi.com/*. Accessed: 2025-01-19.

[264] Petridis, S. et al. 2024. ConstitutionMaker: Interactively critiquing large language models by converting feedback into principles. *In Proc. 29th International Conference on Intelligent User Interfaces* (2024), 853–868.

[265] Pimentel, J.F. et al. 2019. A large-scale study about quality and reproducibility of jupyter notebooks. *In Proc. of 16th Int'l Conf. on Mining Software Repositories (MSR)* (2019).

[266] Piorkowski, D. et al. 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.

[267] Piorkowski, D. et al. 2020. Towards evaluating and eliciting high-quality documentation for intelligent systems. *arXiv [cs.SE]*.

[268] Kästner, Christian. 2025. *Machine Learning in Production: From Models to Products*. MIT Press.

[269] Polman, E. and Maglio, S.J. 2024. The Problem With Behavioral Nudges. *The Wall Street Journal*. The Wall Street Journal.

[270] Polyzotis, N. et al. 2018. Data Lifecycle Challenges in Production Machine Learning: A Survey. *ACM SIGMOD Record*. 47, 2 (2018), 17–28.

[271] Polyzotis, N. et al. 2017. Data Management Challenges in Production Machine Learning. *In Proc. ACM Int'l Conf. on Management of Data* (2017), 1723–1726.

[272] Polyzotis, N. et al. 2019. Data validation for machine learning. *In Proc. of Machine Learning and Systems* (2019), 334–347.

[273] Popular Machine Learning Applications and Use Cases in our Daily Life: 2019. *https://www.analyticsvidhya.com/blog/2019/07/ultimate-list-popular-machine-learning-use-cases/*. Accessed: 2025-09-09.

[274] Prediction Guard: *https://predictionguard.com/*. Accessed: 2026-03-10.

[275] ProjectPro, B.Y. 2021. 15 Machine Learning Use Cases and Applications in 2025. *ProjectPro*. Accessed: 2026-03-10.

[276] Pryzant, R. et al. 2023. Automatic Prompt Optimization with "gradient descent" and beam search. *arXiv [cs.CL]*.

[277] Punter, T. et al. 2003. Conducting on-line surveys in software engineering. *In Proc. 2003 International Symposium on Empirical Software Engineering* (2003).

[278] Rahimi, M. et al. 2019. Toward Requirements Specification for Machine-Learned Components. *In Proc. 27th International Requirements Engineering Conference Workshops (REW)* (2019), 241–244.

[279] Rahman et al, M.S. 2019. Machine Learning Software Engineering in Practice: An Industrial Case Study. *arXiv [cs.SE]*.

[280] Rahman, M.S. et al. 2021. Machine Learning Application Development: Practitioners' Insights. *arXiv [cs.SE]*.

[281] Raji, I.D. et al. 2020. Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. *In Proc. 2020 conference on fairness, accountability, and transparency* (2020), 33–44.

[282] Rakova, B. et al. 2020. Where Responsible AI meets Reality: Practitioner Perspectives on Enablers for shifting Organizational Practices. *In Proc. ACM on Human-Computer Interaction* (2020), 1–23.

[283] Rebedea, T. et al. 2023. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. *arXiv [cs.CL]*.

[284] Ré, C. et al. 2019. Overton: A data system for monitoring and improving machine-learned products. *arXiv [cs.LG]*.

[285] Reimers, N. and Gurevych, I. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *In Proc. 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019).

[286] Responsible AI: *https://www.ibm.com/trust/responsible-ai*. Accessed: 2025-09-11.

[287] Responsible AI: Ethical policies and practices: *https://www.microsoft.com/en-us/ai/responsible-ai*. Accessed: 2025-09-11.

[288] Responsible AI Maturity Model: 2023. *https://www.microsoft.com/en-us/research/publication/responsible-ai-maturity-model/*. Accessed: 2025-09-11.

[289] Responsible AI Transparency Report: *https://cdn-dynmedia-1.microsoft.com/is/content/microsoftcorp/microsoft/msc/documents/presentations/CSR/Responsible-AI-Transparency-Report-2024.pdf*. Accessed: 2025-09-11.

[290] responsible-development-of-ai.pdf: *https://ai.google/static/documents/responsible-development-of-ai.pdf*. Accessed: 2026-03-10.

[291] Ribeiro, D.M. et al. 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.

[292] Ribeiro, M.T. et al. 2020. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. *In Proc. 58th annual meeting of the association for computational linguistics*, 2020.

[293] Riccio, V. et al. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering*. 25, 6 (2020), 5193–5254.

[294] Richardson, B. et al. 2021. Towards fairness in practice: A practitioner-oriented rubric for evaluating fair ML toolkits. *In Proc. 2021 CHI Conference on Human Factors in Computing Systems* (2021), 1–13.

[295] Rismani, S. et al. 2023. From Plane Crashes to Algorithmic Harm: Applicability of Safety Engineering Frameworks for Responsible ML. *In Proc. of 2023 CHI Conf. on Human Factors in Computing Systems* (2023), 1–18.

[296] Rismani, S. and Moon, A. 2023. What does it mean to be a responsible AI practitioner: An ontology of roles and skills. *In Proc. 2023 AAAI/ACM Conference on AI, Ethics, and Society* (2023), 584–595.

[297] Riungu-Kalliosaari, L. et al. 2017. What Can Be Learnt from Experienced Data Scientists? A Case Study. *Product-Focused Software Process Improvement* (2017), 55–70.

[298] Rodrigues, L. et al. 2022. Gamification suffers from the novelty effect but benefits from the familiarization effect: Findings from a longitudinal study. *International journal of educational technology in higher education*. 19, 1 (2022), 13.

[299] Rong, Y. et al. 2023. Towards Human-Centered Explainable AI: A Survey of User Studies for Model Explanations. *IEEE transactions on pattern analysis and machine intelligence* (2023).

[300] 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.

[301] Ryan, C.L. et al. 2022. Transformative learning theory applications in health professional and nursing education: An umbrella review. *Nurse education today*. 119, 105604 (2022), 105604.

[302] Sagor, R. 2011. *The Action Research Guidebook: A Four-Stage Process for Educators and School Teams*. Corwin Press.

[303] Saha, D. et al. 2020. Human Comprehension of Fairness in Machine Learning. *In Proc. AAAI/ACM Conference on AI, Ethics, and Society* (2020), 152.

[304] Sahoo, P. et al. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv [cs.AI]*.

[305] Salay, R. et al. 2017. An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software. *arXiv [cs.AI]*.

[306] Salman, I. et al. 2015. Are Students Representatives of Professionals in Software Engineering Experiments? *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (2015), 666–676.

[307] Sambasivan, N. et al. 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.

[308] Sandelowski, M. et al. 2007. Using qualitative metasummary to synthesize qualitative and quantitative descriptive findings. *Research in nursing & health*. 30, 1 (2007), 99–111.

[309] Sarma, A. et al. 2012. Palantir: Early Detection of Development Conflicts Arising from Parallel Code Changes. *IEEE Transactions on Software Engineering*. 38, 4 (2012), 889–908.

[310] Schelter, S. et al. 2018. Automating Large-scale Data Quality Verification. *In Proc. VLDB Endowment International Conference on Very Large Data Bases*. 11, 12 (2018), 1781–1794.

[311] Schiavo, G. et al. 2022. Trade-offs in the design of multimodal interaction for older adults. *Behaviour & information technology*. 41, 5 (2022), 1035–1051.

[312] Schoen, D.A. 2017. *The reflective practitioner: How professionals think in action*. Routledge.

[313] Schreier, M. 2012. *Qualitative content analysis in practice*. Sage Publications.

[314] Sculley, D. et al. 2011. Detecting adversarial advertisements in the wild. *In Proc. 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (2011), 274–282.

[315] Sculley, D. et al. 2015. Hidden Technical Debt in Machine Learning Systems. *Adv. in Neu. Info. Proc. Sys.* 28, (2015), 2503–2511.

[316] Seidman, I. 2019. *Interviewing as qualitative research*. Teachers' College Press.

[317] Selbst, A.D. et al. 2019. Fairness and Abstraction in Sociotechnical Systems. *In Proc. Conference on Fairness, Accountability, and Transparency* (2019), 59–68.

[318] 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.

[319] Sengers, P. et al. 2005. Reflective design. *In Proc. 4th decennial conference on Critical computing: between sense and sensibility* (2005), 49–58.

[320] Serban, A. et al. 2020. Adoption and Effects of Software Engineering Best Practices in Machine Learning. *In Proc. of 14th Int'l Symposium on Empirical Software Engineering and Measurement (ESEM)* (2020), 1–12.

[321] Serban, A. et al. 2021. Practices for Engineering Trustworthy Machine Learning Applications. *In Proc. 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)* (2021), 97–100.

[322] 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.

[323] Seymoens, T. et al. 2018. A methodology to involve domain experts and machine learning techniques in the design of human-centered algorithms. *In Proc. IFIP Working Conf. on Human Work Interaction Design* (2018).

[324] Shankar, S. et al. 2022. Operationalizing Machine Learning: An Interview Study. *arXiv [cs.SE]*.

[325] Shankar, S. et al. 2024. SPADE: Synthesizing assertions for large language model pipelines. *arXiv [cs.DB]*.

[326] Shankar, S. et al. 2024. Who validates the validators? Aligning LLM-assisted evaluation of LLM outputs with human preferences. *In Proc. 37th Annual ACM Symposium on User Interface Software and Technology* (2024), 1–14.

[327] Shelby, R. et al. 2023. Sociotechnical harms of algorithmic systems: Scoping a taxonomy for harm reduction. *In Proc. 2023 AAAI/ACM Conference on AI, Ethics, and Society* (2023), 723–741.

[328] Shieh, E. et al. 2025. Laissez-faire harms: Algorithmic biases in generative language models (extended abstract). *In Proc. AAAI/ACM Conference on AI, Ethics, and Society*. 8, 3 (2025), 2373–2374.

[329] Shin, T. et al. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv [cs.CL]*.

[330] Shneiderman, B. 2020. Bridging the gap between ethics and practice. *ACM transactions on interactive intelligent systems*. 10, 4 (2020), 1–31.

[331] Siebert, J. et al. 2020. Towards Guidelines for Assessing Qualities of Machine Learning Systems. *In Proc. of Int'l Conf. on the Quality of Information and Communications Technology* (2020), 17–31.

[332] Singh, G. et al. 2019. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.* 3, POPL (2019), 1–30.

[333] Sivakumar, N. et al. 2025. Bias after prompting: Persistent discrimination in large language models. *Findings of the Association for Computational Linguistics: EMNLP 2025* (2025), 18568–18593.

[334] Ha, Sierra K., et al. 2025. Impact of teleretinal screening program on diabetic retinopathy screening compliance rates in community health centers: a quasiexperimental study. *BMC Health Services Research*. 25, 318 (2025).

[335] Slovic, P. et al. 2007. The affect heuristic. *European journal of operational research*. 177, 3 (2007), 1333–1352.

[336] Smith, D. 2017. Exploring development patterns in data science. *https://www.theorylane.com/2017/10/20/some-development-patterns-in-data-science/. Accessed: 2026-02-19.*

[337] Smith, D. et al. 2014. Overcoming barriers to collaboration in an open source ecosystem. *Technology Innovation Management Review*. 4, 1 (2014).

[338] Smith, J.J. et al. 2025. Pragmatic fairness: Evaluating ML fairness within the constraints of industry. *In Proc. 2025 ACM Conference on Fairness, Accountability, and Transparency* (2025), 628–638.

[339] de Souza Nascimento, E. et al. 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.

[340] Sommerville, I. Software Engineering 9th Edition, vol. 137035152 (2011). *ISBN-10*.

[341] de Souza, C.R.B. and Redmiles, D.F. 2008. An Empirical Study of Software Developers' Management of Dependencies and Changes. *In Proc. Int'l Conf. Software Engineering (ICSE)* (2008), 241–250.

[342] Spencer, D. 2009. *Card Sorting: Designing Usable Categories*. Rosenfeld Media.

[343] Springer, A. et al. 2018. Dice in the black box: User experiences with an inscrutable algorithm. *arXiv [cs.HC]*.

[344] Strauss, A. and Corbin, J. 1994. Grounded theory methodology: An overview. *Handbook of qualitative research*. N.K. Denzin, ed. 273–285.

[345] Strauss, A. and Corbin, J.M. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*. SAGE Publications.

[346] Stringer, E.T. and Aragón, A.O. 2020. *Action Research*. SAGE Publications.

[347] Strobelt, H. et al. 2023. Interactive and visual prompt engineering for ad-hoc task adaptation with large language models. *IEEE Transactions on Visualization and Computer Graphics*. 29, 1 (2023), 1146–1156.

[348] Studer, S. et al. 2021. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*. 3, 2 (2021), 392–413.

[349] Supplementary Documents: A Dataset and Analysis of Open-Source Machine Learning Products: *https://osf.io/gqyex/*. Accessed: 2025-01-19.

[350] Tahir, M. et al. 2020. Evaluation of quality and readability of Online Health Information on high Blood Pressure using DISCERN and Flesch-Kincaid tools. *Applied Sciences (Basel, Switzerland)*. 10, 9 (2020), 3214.

[351] Tang, Y. et al. 2021. An Empirical Study of Refactorings and Technical Debt in Machine Learning Systems. *In Proc. EEE/ACM 43rd international conference on software engineering (ICSE)* (2021), 238–250.

[352] Testing Language Models (and Prompts) Like We Test Software: 2023. *https://towardsdatascience.com/testing-large-language-models-like-we-test-software-92745d28a359/*. Accessed: 2026-03-10.

[353] The board game: 2025. *https://tethics.eu/the-board-game/*. Accessed: 2025-09-09.

[354] The Ethical Dilemmas Board Game: *https://cfrr.worldbank.org/publications/ethical-dilemmas-board-game*. Accessed: 2025-09-09.

[355] 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*. Accessed: 2025-01-19.

[356] The Shift from Models to Compound AI Systems: *http://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/*. Accessed: 2025-08-27.

[357] Tonekaboni, S. et al. 2019. What Clinicians Want: Contextualizing Explainable Machine Learning for Clinical End Use. *In Proc. 4th Machine Learning for Healthcare Conference* (2019), 359–380.

[358] Tramèr, F. et al. 2017. FairTest: Discovering Unwarranted Associations in Data-Driven Applications. *In Proc. European Symposium on Security and Privacy (EuroS P)* (2017), 401–416.

[359] Tranquillo, J. 2017. The T-Shaped Engineer. *Journal of Engineering Education Transformations*. 30, 4 (2017), 12–24.

[360] 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.

[361] Vaast, E. 2025. Experiencing and addressing the moral ambivalence of developing digital technology: Insights from artificial intelligence developers. *In Proc. Annual Hawaii International Conference on System Sciences* (2025).

[362] Vakkuri, V. et al. 2022. How do software companies deal with artificial intelligence ethics? A gap analysis. *In Proc. 26th International Conference on Evaluation and Assessment in Software Engineering* (2022), 100–109.

[363] Varanasi, R.A. and Goyal, N. 2023. "It is currently hodgepodge": Examining AI/ML Practitioners' Challenges during Co-production of Responsible AI Values. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–17.

[364] Vatsal, S. and Dubey, H. 2024. A survey of prompt engineering methods in large language models for different NLP tasks. *arXiv [cs.CL]*.

[365] Vera Liao, Q. and Varshney, K.R. 2021. Human-Centered Explainable AI (XAI): From Algorithms to User Experiences. *arXiv [cs.AI]*.

[366] Villamizar, H. et al. 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.

[367] 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.

[368] Wagstaff, K. 2012. Machine Learning that Matters. *arXiv 1206.4656*.

[369] Wang, A. et al. 2024. Strategies for increasing corporate responsible AI prioritization. *In Proc. AAAI/ACM Conference on AI, Ethics, and Society* (2024), 1514–1526.

[370] Wang, A.Y. et al. 2019. How Data Scientists Use Computational Notebooks for Real-Time Collaboration. *In Proc. ACM on Human-Computer Interaction*. 3, CSCW (2019), 39.

[371] Wang, D. et al. 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.

[372] Wang, Q. et al. 2023. Designing responsible AI: Adaptations of UX practice to meet responsible AI challenges. *In Proc. 2023 CHI Conference on Human Factors in Computing Systems* (2023), 1–16.

[373] Wang, S.M. et al. 2023. Development and integration of machine learning algorithm to identify peripheral arterial disease: Multistakeholder qualitative study. *JMIR formative research*. 7, (2023).

[374] Wang, Y. et al. 2020. Toward an understanding of responsible artificial intelligence practices. *In Proc. 53rd Hawaii International Conference on System Sciences* (2020), 4962–4971.

[375] Wang, Z.J. et al. 2024. Farsight: Fostering responsible AI awareness during AI application prototyping. *In Proc. CHI Conference on Human Factors in Computing Systems* (2024), 1–40.

[376] Wan, Z. et al. 2019. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering*. 47, 9 (2019), 1857–1871.

[377] Washizaki, H. et al. 2020. Machine learning architecture and design patterns. *IEEE Software*. 8, (2020).

[378] Washizaki, H. et al. 2020. Practitioners' insights on machine-learning software engineering design patterns: a preliminary study. *In Proc. 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (2020), 797–799.

[379] Waterman, M. et al. 2015. How Much Up-Front? A Grounded theory of Agile Architecture. *In Proc. of 37th Int'l Conf.on Software Engineering* (2015), 347–357.

[380] Watkins-Hayes, C. 2019. *Remaking a life*. University of California Press.

[381] What language are most commonly used for web development: *https://www.dotnetlanguages.net/web-languages-what-language-are-most-commonly-used-for-web-development/*. Accessed: 2025-01-19.

[382] 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*. Accessed: 2025-01-19.

[383] 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/*. Accessed: 2025-01-19.

[384] Widder, D.G. et al. 2024. Power and play: Investigating "license to critique" in teams' AI ethics discussions. *In Proc. ACM on human-computer interaction*. 8, CSCW2 (2024), 1–23.

[385] Widder, D.G. and Nafus, D. 2023. Dislocated accountabilities in the "AI supply chain": Modularity and developers' notions of responsibility. *Big data & society*. 10, 1 (2023).

[386] Widyasari et al, R. 2023. NICHE: A Curated Dataset of Engineered Machine Learning Projects in Python. *arXiv [cs.SE]*.

[387] Wiens, J. et al. 2019. Do no harm: a roadmap for responsible machine learning for health care. *Nature medicine*. 25, 9 (2019), 1337–1340.

[388] Williamson, J.M.L. and Martin, A.G. 2010. Analysis of patient information leaflets provided by a district general hospital by the Flesch and Flesch-Kincaid method. *International Journal of Clinical Practice*. 64, 13 (2010), 1824–1831.

[389] Winecoff, A.A. and Watkins, E.A. 2022. Artificial concepts of artificial intelligence. *In Proc. 2022 AAAI/ACM Conference on AI, Ethics, and Society* (2022), 788–799.

[390] Winecoff, A. and Bogen, M. 2025. Improving governance outcomes through AI documentation: Bridging theory and practice. *In Proc. 2025 CHI Conference on Human Factors in Computing Systems* (2025), 1–18.

[391] 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.

[392] Wohlrab, R. et al. 2019. Boundary objects and their use in agile systems engineering. *Journal of software (Malden, MA)*. 31, 5 (2019), e2166.

[393] Wu, S. et al. 2023. ScatterShot: Interactive in-context example curation for text transformation. *In Proc. 28th International Conference on Intelligent User Interfaces* (2023), 353–367.

[394] Wu, T. et al. 2022. PromptChainer: Chaining large language model prompts through visual programming. *CHI Conference on Human Factors in Computing Systems Extended Abstracts* (New York, NY, USA, Apr. 2022).

[395] Xia, B. et al. 2023. Towards concrete and connected AI risk assessment (C2AIRA): A systematic mapping study. *arXiv [cs.SE]*.

[396] Xie, X. et al. 2011. Testing and Validating Machine Learning Classifiers by Metamorphic Testing. *The Journal of systems and software*. 84, 4 (2011), 544–558.

[397] Yang, Q. et al. 2018. Grounding Interactive Machine Learning Tool Design in How Non-Experts Actually Build Models. *In Proc. Conf. on Designing Interactive Systems* (2018), 573–584.

[398] Yang, X. et al. 2025. Rethinking prompt-based debiasing in large language models. *arXiv [cs.CL]*.

[399] Yokoyama, H. 2019. Machine Learning System Architectural Pattern for Improving Operational Stability. *In Proc. of Int'l Conf. on Software Architecture Companion (ICSA-C)* (2019), 267–274.

[400] 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.

[401] Zeller, A. 2005. *Why programs fail: A guide to systematic debugging*. Morgan Kaufmann.

[402] Zhang, A.X. et al. 2020. How do data science workers collaborate? Roles, workflows, and tools. *In Proc. ACM Hum. Comput. Interact.* 4, CSCW1 (2020), 1–23.

[403] Zhang, J.M. et al. 2022. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*. 48, 1 (2022), 1–36.

[404] Zhang, X. et al. 2019. Software Engineering Practice in the Development of Deep Learning Applications. *arXiv [cs.SE]*.

[405] Zheng, L. et al. 2023. Judging LLM-as-a-judge with MT-bench and Chatbot Arena. *Advances in neural information processing systems*. 36, (2023), 46595–46623.

[406] Zhou, S. et al. 2020. How Has Forking Changed in the Last 20 Years? A Study of Hard Forks on GitHub. *In Proc. of 42nd Int'l Conf. on Software Engineering (ICSE)* (2020), 445–456.

[407] Zillmann, D. 2006. Exemplification effects in the promotion of safety and health. *The Journal of communication*. 56 (2006), S221–S237.

[408] Zinkevich, M. 2017. Rules of machine learning: Best practices for ML engineering. *https://developers.google.com/machine-learning/guides/rules-of-ml*. (2017).

[409] 2025. Appendix: Policy alone is probably not the solution: A large-scale experiment on how developers struggle to design meaningful end-user explanations. *https://osf.io/hbzyd/?view_only=a8d7c9c2c046407d9ce30c2b2f87eff4*. Accessed: 2025-01-19.

[410] *Awesome-LLMOps: An awesome & curated list of best LLMOps tools for developers*. Github. *https://github.com/InftyAI/Awesome-LLMOps*. Accessed: 2025-01-19.

[411] Topsakal, O., & Akinci, T. C. 2023. Creating Large Language Model Applications Utilizing LangChain: A Primer on Developing LLM Apps Fast. In Proc. applied engineering and natural sciences (2023), Vol. 1, No. 1, 1050-1056.

[412] Nahar, N. et al. 2026. I Don't Think RAI Applies to My Model. *Think RAI Applies to My Model" - Engaging Non-champions with Sticky Stories for Responsible AI Work. In Proc. 2026 CHI Conference on Human Factors in Computing Systems (2026).*

[413] Nahar, N. et al. 2024. Regulating Explainability in Machine Learning Applications – Observations from a Policy Design Experiment. *In Proc. ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT)* (2024).

[414] 2024. The problem with the nudge effect: it can make you buy more carrots – but it can't make you eat them. *The Guardian*. The Guardian.

[415] Humble, Jez, and David Farley. 2010. Continuous delivery: reliable software releases through build, test, and deployment automation. *Pearson Education*.

[416] Lwakatare, Lucy Ellen, et al. 2019. DevOps in practice: A multiple case study of five companies. Information and software technology 114 (2019): 217-230.

[417] Assres, G. et al. 2025. State-of-the-Art and Challenges of Engineering ML-Enabled Software Systems in the Deep Learning Era. *ACM Computing Surveys*.

[418] Mailach, A. et al. 2025. Themes of Building LLM-Based Applications for Production: A Practitioner's View. *In Proc. 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)* (2025), 18-30.

[419] Rahimi, M. et al. 2023. How Mature is Requirements Engineering for AI-Based Systems? A Systematic Mapping Study on Practices, Challenges, and Future Research Directions. *Information and Software Technology*.

[420] Winkler, S. and Vogelsang, A. 2022. Requirements Engineering for Machine Learning-Based AI Systems: A Tertiary Study. *Journal of Systems and Software*.

[421] Vogelsang, A. et al. 2020. Requirements Engineering for Machine Learning: A Review and Reflection. *In Proc. 30th international requirements engineering conference workshops (REW)* (2020), 166-175.

[422] Ghofrani, J. et al. 2024. Data Challenges in AI Systems and Their Solutions: A Requirements and AI Engineering Systematic Literature Review and Comparison. *IEEE Access*.

[423] Schelter, S. et al. 2021. What About the Data? A Mapping Study on Data Engineering for AI Systems. *In Proc. 4th International Conference on AI Engineering–Software Engineering for AI (CAIN)* (2021), 43-52.

[424] Girardi, D. et al. 2023. Agile Management for Machine Learning: A Systematic Mapping Study. *Empirical Software Engineering*.

[425] Zöller, M. and Huber, M. 2021. A Multivocal Literature Review on the Benefits and Limitations of Industry-Leading AutoML Tools. *Information and Software Technology*.

[426] Kumar, A. et al. 2023. A Review on Interplay of AutoML and MLOps ("AutoMLOps"): Current State, Challenges, and Future Scope. *Machine Learning and Knowledge Extraction*.

[427] Hummer, W. et al. 2022. A Systematic Literature Review of MLOps Adoption in Software Development Companies: Attributes, Maturity, Benefits and Challenges. *Empirical Software Engineering*.

[428] Zhang, Q. et al. 2024. Maintainability and Scalability in Machine Learning: Challenges and Solutions. *ACM Computing Surveys*.

[429] Protschky, D. et al. 2025. What Gets Measured Gets Improved: Monitoring Machine Learning Applications in Their Production Environments. *IEEE Access*.

[430] Hupont, I. et al. 2023. A.I. Robustness: A Human-Centered Perspective on Technological Challenges and Opportunities. *ACM Computing Surveys*.

[431] Tahaei, M. et al. 2023. Designing Secure AI-Based Systems: A Multi-Vocal Literature Review. *In Proc. IEEE Int'l Conf. on Software Engineering Workshops (ICSEW)* (2023), 13-19.

[432] Galhotra, S. et al. 2022. Fairness Testing: A Comprehensive Survey and Analysis of Trends. *ACM Transactions on Software Engineering and Methodology*.

[433] Chadli, Kouider, Goetz Botterweck, and Takfarinas Saber. 2024. The Environmental Cost of Engineering Machine Learning-Enabled Systems: A Mapping Study. *In Proc. 4th Workshop on Machine Learning and Systems) (2020), 200-207*.

[434] Alegre, L. et al. 2022. The Real Deal: A Review of Challenges and Opportunities in Moving Reinforcement Learning-Based Traffic Signal Control Systems Towards Reality. *IEEE Transactions on Intelligent Transportation Systems*.

[435] Martínez-Fernández, S. et al. 2022. Continuous Software Engineering Practices in AI/ML Development Past the Narrow Lens of MLOps: Adoption Challenges. *IEEE Software*.

[436] Zimelewicz, Eduardo, et al. 2024. ML-Enabled Systems Model Deployment and Monitoring: Status Quo and Problems. *In Proc. International Conference on Software Quality* (2024), 112-131.

[437] Alves, Antonio Pedro Santos, Marcos Kalinowski, and Daniel Méndez. 2024. Requirements Engineering for ML-Enabled Systems: Status Quo and Problems. *In Proc. of the Brazilian Symposium on Software Quality (2024)*, 697-699.

[438] Villamizar, Hugo, and Marcos Kalinowski. 2024. Identifying Concerns When Specifying Machine Learning-Enabled Systems: A Perspective-Based Approach. *In Proc. of the Brazilian Symposium on Software Quality (2024)*, 673-675.

[439] Wan, Zhiyuan, et al. 2023. Software Architecture in Practice: Challenges and Opportunities. *In Proc. 31st ACM joint European software engineering conference and symposium on the foundations of software engineering (2023), 1457-1469.*

[440] Sens, Yorick, et al. 2025. A Large-Scale Study of Model Integration in ML-Enabled Software Systems. *In Proc. 47th International Conference on Software Engineering (ICSE) (2025), 1165-1177.*

[441] Bucaioni, Alessio, Rick Kazman, and Patrizio Pelliccione. 2025. A Checklist of Quality Concerns for Architecting ML-Intensive Systems. *Journal of Systems and Software*.

[442] Rajbahadur, Gopi Krishnan, et al. 2024. From Cool Demos to Production-Ready FMware: Core Challenges and a Technology Roadmap. *arXiv preprint*.

[443] Pahune, Saurabh, and Zahid Akhtar. 2025. Transitioning from MLOps to LLMOps: Navigating the Unique Challenges of Large Language Models. *Information 16.2*: 87.

[444] Hassan, Ahmed E., et al. 2024. Rethinking Software Engineering in the Era of Foundation Models: A Curated Catalogue of Challenges in the Development of Trustworthy FMware. *In Proc. 32nd ACM international conference on the foundations of software engineering* (2024), 294-305.

[445] Chen, Xiang, et al. 2024. An Empirical Study on Challenges for LLM Application Developers. *ACM Transactions on Software Engineering and Methodology* 34.7 (2025): 1-37.

[446] Chen, Xin, Yan Li, and Xiaoming Wang. 2025. Design Principles and Guidelines for LLM Observability: Insights from Developers. *In Proc. of the CHI Conf. on Human Factors in Computing Systems Extended Abstracts* (2025), 1-9.

[447] Kumar, Aayush, et al. 2025. Why AI Agents Still Need You: Findings from Developer-Agent Collaborations in the Wild. *In Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE)* (2025).

[448] Sun, Simin, and Miroslaw Staron. 2026. Agentic Pipelines in Embedded Software Engineering: Emerging Practices and Challenges. *arXiv preprint*.

[449] Amalfitano, Domenico, et al. 2024. A Research Roadmap for Augmenting Software Engineering Processes and Software Products with Generative AI. *ACM Transactions on Software Engineering and Methodology*.

[450] Du, Zhuoyun, et al. 2025. Multi-Agent Collaboration via Cross-Team Orchestration. *In Proc. Findings of the Association for Computational Linguistics (2025)*.

[451] Nahar, Nadia, et al. 2025. Policy alone is probably not the solution: A large-scale experiment on how developers struggle to design meaningful end-user explanations. *arXiv preprint*.

[452] Chin-Yee, Benjamin, and Ross Upshur. 2019. "Three Problems with Big Data and Artificial Intelligence in Medicine." *Perspectives in Biology and Medicine* 62 (2): 237–256.

[453] Dong, Yi, Ronghui Mu, Gaojie Jin, et al. 2024. "Building Guardrails for Large Language Models." In *arXiv [cs.CL]*.

[454] Costa, Manuel, Boris Köpf, Aashish Kolluri, et al. 2025. "Securing AI Agents with Information-Flow Control." In *arXiv [cs.CR]*.

[455] Yang, Qian, Aaron Steinfeld, Carolyn Rosé, and John Zimmerman. 2020. "Re-Examining Whether, Why, and How Human-AI Interaction Is Uniquely Difficult to Design." *In Proc. 2020 CHI Conference on Human Factors in Computing Systems* (2020) , April 21, 1–13.

[456] Selbst, Andrew D., and Solon Barocas. 2018. "The Intuitive Appeal of Explainable Machines." *Fordham Law Review* 87: 1085–1139.

[457] Alpsancar, Suzana, Tobias Matzner, and Martina Philippi. 2024. "Unpacking the Purposes of Explainable AI." *Smart Ethics in the Digital World: In Proc. 21th International Conference on the Ethical and Social Impacts of ICT (2024)*, 31–35.

[458] Zhang, Chanyuan (abigail), Soohyun Cho, and Miklos Vasarhelyi. 2022. "Explainable Artificial Intelligence (XAI) in Auditing." *International Journal of Accounting Information Systems* 46: 100572.

[459] Wachter, Sandra, Brent Mittelstadt, and Luciano Floridi. 2017. "Why a Right to Explanation of Automated Decision-Making Does Not Exist in the General Data Protection Regulation." *International Data Privacy Law* 7 (2): 76–99.

[460] Luo, Haoyan, and Lucia Specia. 2024. "From Understanding to Utilization: A Survey on Explainability for Large Language Models." In *arXiv [cs.CL]*.

[461] Miller, Tim, Piers Howe, and Liz Sonenberg. 2017. "Explainable AI: Beware of Inmates Running the Asylum or: How I Learnt to Stop Worrying and Love the Social and Behavioural Sciences." In *arXiv [cs.AI]*.

[462] Panigutti, Cecilia, Andrea Beretta, Daniele Fadda, et al. 2023. "Co-Design of Human-Centered, Explainable AI for Clinical Decision Support." *ACM Trans. Interact. Intell. Syst.* 13 (4): 1–35.

[463] Rong, Yao, Tobias Leemann, Thai-Trang Nguyen, et al. 2022. "Towards Human-Centered Explainable AI: User Studies for Model Explanations." In *arXiv [cs.AI]*.

[464] Eiband, Malin, Daniel Buschek, Alexander Kremer, and Heinrich Hussmann. 2019. "The Impact of Placebic Explanations on Trust in Intelligent Systems." *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (2019), 1-6.

[465] Stumpf, Simone, Adrian Bussone, and Dympna O'sullivan. 2016. "Explanations Considered Harmful? User Interactions with Machine Learning Systems." *In Proc. ACM SIGCHI Conference on Human Factors in Computing Systems (2016)*.

[466] Ehsan, Upol, Samir Passi, Q. Vera Liao, et al. 2021. "The Who in Explainable AI: How AI Background Shapes Perceptions of AI Explanations." *In Proc. CHI Conference on Human Factors in Computing Systems* (2021), 28, 1–32.

[467] Laato, Samuli, Miika Tiainen, A. K. M. Najmul Islam, and Matti Mäntymäki. 2022. "How to Explain AI Systems to End Users: A Systematic Literature Review and Research Agenda." *Internet Research* 32 (7): 1–31.

[468] Marsden, Eric. 2014. *Risk Regulation, Liability and Insurance: Literature Review of Their Influence on Safety Management*. Foundation for an Industrial Safety Culture. https://www.foncsi.org/en/publications/risk-regulation-liability-insurance. Accessed: 2025-01-19.

[469] Heimer, C. A. 2025. *Governing the Global Clinic: HIV and the Legal Transformation of Medicine*. https://press.uchicago.edu/dam/ucp/books/pdf/Heimer_Appendices.pdf. Accessed: 2025-01-19.

[470] "Press Releases: Artificial Intelligence Act: MEPs Adopt Landmark Law." 2024. https://www.europarl.europa.eu/news/en/press-room/20240308IPR19015/artificial-intelligence-act-meps-adopt-landmark-law. Accessed: 2025-01-19.

[471] U.S. White House. 2022. "Blueprint for an AI Bill of Rights: Making Automated Systems Work for the American People." https://www.whitehouse.gov/ostp/ai-bill-of-rights/. Accessed: 2025-01-19.

[472] The White House. 2023. "Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence." www.whitehouse.gov/briefing-room/presidential-actions. Accessed: 2025-01-19.

[473] Jobin, Anna, Marcello Ienca, and Effy Vayena. 2019. "The Global Landscape of AI Ethics Guidelines." *Nature Machine Intelligence* 1 (9): 389–399.

[474] Gray, Garry C., and Susan S. Silbey. 2014. "Governing inside the Organization: Interpreting Regulation and Compliance." *American Journal of Sociology* 120 (1): 96–145.

[475] Silbey, Susan S. 2013. "Organizational Challenges to Regulatory Enforcement and Compliance." In *The ANNALS of the American Academy of Political and Social Science*, vol. 649, 649. no. 1.

[476] Cech, Erin A., and H. M. Sherick. 2015. "Chapter 10: Depoliticization and the Structure of Engineering Education." In *International Perspectives on Engineering Education*.

[477] Suran M, Hswen Y. 2024. "How Do Policymakers Regulate AI *and* Accommodate Innovation in Research and Medicine?" *The Journal of the American Medical Association* 331 (3): 185–187.

[478] Nelson, Alondra. 2024. "The Right Way to Regulate AI." *Foreign Affairs*, 12: 1–11.

[479] Ferreira, Gabriel, Christian Kästner, Joshua Sunshine, Sven Apel, and William Scherlis. 2019. "Design Dimensions for Software Certification: A Grounded Analysis." In *arXiv [cs.SE]*.

[480] Papademetris, Xenophon, Ayesha N. Quraishi, and Gregory P. Licholai. 2022. *Introduction to Medical Software: Foundations for Digital Health, Devices, and Diagnostics*. Cambridge University Press.

[481] Bietti, Elettra. 2020. "From Ethics Washing to Ethics Bashing: A View on Tech Ethics from within Moral Philosophy." *In Proc. 2020 Conference on Fairness, Accountability, and Transparency* (2020) 210–219.

[482] Metcalf, Jacob, Emanuel Moss, and Danah Boyd. 2019. "Owning Ethics: Corporate Logics, Silicon Valley, and the Institutionalization of Ethics." *Social Research: An International Quarterly* 86 (2): 449–476.

[483] Greene, Daniel, Anna Lauren Hoffmann, and Luke Stark. 2019. "Better, Nicer, Clearer, Fairer: A Critical Assessment of the Movement for Ethical Artificial Intelligence and Machine Learning." *Hawaii International Conference on System Sciences (2019)*, 2122–2131.

[484] Ochigame, Rodrigo. 2019. "The Invention of 'ethical AI': How Big Tech Manipulates Academia to Avoid Regulation." The Intercept.

[485] Ferretti, Thomas. 2022. "An Institutionalist Approach to AI Ethics: Justifying the Priority of Government Regulation over Self-Regulation." *Moral Philosophy and Politics* 9 (2): 239–265.

[486] Thorne, S., Kirkham, S.R. and MacDonald-Emes, J. 1997. Interpretive description: a noncategorical qualitative alternative for developing nursing knowledge. *Research in nursing & health*. 20, 2 (1997), 169–177.

[487] Thorne, S. 2016. *Interpretive description: Qualitative research for applied practice*. Routledge.

[488] Flower, L. and Hayes, J. R. 1981. A Cognitive Process Theory of Writing. *Cognitive Science*. 4, 4 (1981), 365–387.

[489] Hayes, J. R. 1996. A New Framework for Understanding Cognition and Affect in Writing. In Levy, C. M. and Ransdell, S., eds., *The Science of Writing: Theories,*

Methods, Individual Differences, and Applications. Lawrence Erlbaum Associates, 1–27.

[490] Kolb, D. A. 1984. *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall.

[491] Pruitt, J. and Grudin, J. 2003. Personas: Practice and Theory. In *In Proc. Conference on Designing for User Experiences (2023)*. ACM, 1–15.

[492] Carroll, J. M. 2000. *Making Use: Scenario-Based Design of Human–Computer Interactions*. MIT Press.

[493] Leveson, N. G. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press.

[494] Kumar, A. et al. 2025. Why AI Agents Still Need You: Findings from Developer-Agent Collaborations in the Wild. *arXiv [cs.SE]*.

[495] Cynthia, Shamse Tasnim, Joy Krishan Das, and Banani Roy. 2026. Are We All Using Agents the Same Way? An Empirical Study of Core and Peripheral Developers Use of Coding Agents. *arXiv [cs.SE]*.

[496] Dakhel, Arghavan Moradi, et al. 2023. Github copilot ai pair programmer: Asset or liability? *Journal of Systems and Software 203 (2023): 111734.*

[497] Meyerson, D.E. 2003. *Tempered Radicals: How People Use Difference to Inspire Change at Work*. Harvard Business School Press.