A Meta-Summary of Challenges in Building Products with ML Components – Collecting Experiences from 4758+ Practitioners

Nadia Nahar*†, Haoran Zhang†, Grace Lewis‡, Shurui Zhou§, Christian Kästner†
*Carnegie Mellon University, †Carnegie Mellon Software Engineering Institute, ‡University of Toronto
*nadian@andrew.cmu.edu

***Abstract*—Incorporating machine learning (ML) components into software products raises new software-engineering challenges and exacerbates existing ones. Many researchers have invested significant effort in understanding the challenges of industry practitioners working on building products with ML components, through interviews and surveys with practitioners. With the intention to aggregate and present their collective findings, we conduct a meta-summary study: We collect 50 relevant papers that together interacted with over 4758 practitioners using guidelines for systematic literature reviews. We then collected, grouped, and organized the over 500 mentions of challenges within those papers. We highlight the most commonly reported challenges and hope this meta-summary will be a useful resource for the research community to prioritize research and education in this field.**

I. Introduction

After decades of effort in machine learning (ML) research to build better models, researchers from industry and academia have recently started to shift their attention to improving how to build software products with such models. Incorporating a ML component into a software product is often argued to be harder than incorporating traditional functional components, because of the specific characteristics of machine learning (e.g., based on data, no specifications in the traditional sense, fairness concerns) and how they impact the entire life cycle of the product [58], [83], [111], [41], [49]. While the traditional software development process has challenges of its own, bringing ML into the picture is argued to break a lot of existing software architecture and engineering assumptions [58], [49]. This leads initiatives to rethink existing processes and practices and shift priorities in software teams. As a result, we keep hearing from practitioners on how they perceive building, deploying, and incorporating machine learning in software products as a challenge, even when the initial ML research and model prototypes seemed promising.

While some practitioners give talks on challenges or write experience papers (e.g., examples in academic venues surveyed elsewhere [66], [84]), researchers have also been actively studying the challenges faced by practitioners when building software products with ML components across many projects. In recent years, many researchers have *interviewed* or *surveyed* practitioners to identify what has really changed for them with the introduction of machine learning, often with the goal of identifying challenges, research opportunities, and best practices in a rapidly changing field. While some studies focus on specific aspects, such as challenges regarding architecture [109], collaboration [77], or fairness [36], [94], 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.

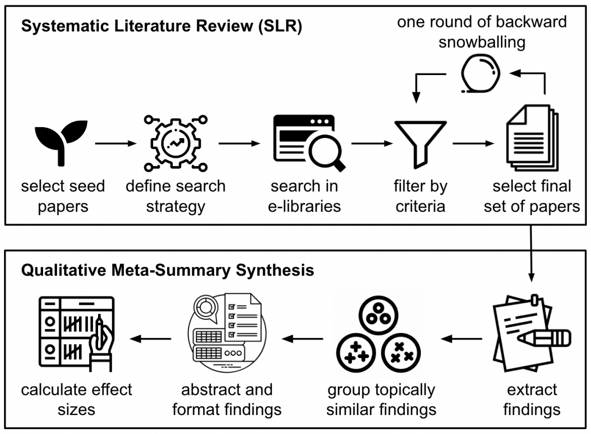
Systematic Literature Review (SLR)
one round of backward snowballing
select seed papers
define search strategy
search in e-libraries
filter by criteria
select final set of papers
Qualitative Meta-Summary Synthesis
calculate effect sizes
abstract and format findings
group topically similar findings
extract findings

Fig. 1. Research Method

In this paper, we aim to consolidate knowledge about challenges in the practice of building software products with ML components, 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* [103], [95], [24], we analyze, organize, and synthesize findings across all these studies (as shown in Fig. 1), answering the overall research question: **What are the challenges experienced by industry practitioners in building software products with ML components?**

We group the challenges from 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 engineering challenges in ML-specific stages: (4) model development, and (5) data engineering. Other issues relate to cross-cutting concerns related to (6) process, and (7) organization and teams. Following the meta-summary method, we present and organize the challenges mentioned by the practitioners in the original papers as they have been reported, without attempting to speculate or pass our own judgments on the findings. We conclude the paper with a brief discussion, reflecting our own views.

Our key contribution is the meta-summary, which we present in narrative form in this paper. We additionally provide details with clear traceability to findings and papers as supplementary documents [76].

II. Scoping and 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 [51], data exploration [74], [62], data-science processes [29], [69], development in notebooks [19], [32], [87], AutoML [121]), 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.

Understanding practitioner needs. Academic research is often criticized for being far removed from the needs faced by practitioners in industry [120], [94], [8]. 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 [86], [85], many direct collaborations with an industrial partner [52], [93], and many experience reports published by practitioners in papers [9], [48], [31], [59], talks [70], [30], [26], or blog posts [131], [34], [113]. To understand problems across teams, many researchers conduct interviews across multiple teams and organizations, e.g., [119], [58], [109], [77], [65], 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., [57], [44], [122], [130].

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

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 [129], [5], [97], [15], [38], safety and security [13], [63], [38], data management [90], and even trying to cover published research on SE4ML broadly [27], [68]. 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 [66], [84]. In this work, we specifically perform a meta-summary of academic papers reporting on interviews and surveys with practitioners.

III. Research Method

The goal of this paper is to summarize challenges in building products with ML components, 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 [50]. 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 [39]. As our aim is to discover patterns or themes of challenges in building products with ML components, 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 [39], [95], [103]. 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 [95]. Thus, we apply the meta-summary method [103], [24], [95] to perform the quantitative aggregation of the qualitative evidence that we present as findings. Fig. 1 shows the overview of the research process followed in this study.

A. Paper Selection

To increase reliability, reproducibility and objectivity of the process for paper selection, we follow the established procedure of conducting systematic literature reviews [50].

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. [104], 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, [45], [20], [64]. 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 [25], [66]). 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 I).

TABLE I
PAPER SELECTION

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

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.

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 II), 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., [35], [46], [23], [80]) or interviewing only non-technical people (e.g., [117], [12], [33], [100]).

Snowballing. To capture relevant papers that did not match our keywords in their abstract, we performed one iteration of backward snowballing [126], 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 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. This sudden explosion of interview and survey studies

TABLE II
INCLUSION AND EXCLUSION CRITERIA

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.

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 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.

B. Qualitative Meta-Summary Process

As stated earlier, we used the meta-summary research method [103], [95], [24] 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. To maintain consistency in extracting the findings, we considered only *challenges* that were derived from the interview and survey answers in the papers, not challenges derived from other literature or personal experience of the authors. 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. Different papers grouped findings in different ways and using different terminologies; we aimed to find a consistent organizational principle. For identifying similar topics and grouping those together, we needed to understand and compare those reported challenges in their original context. *Card sorting* is a common technique for grouping similar findings [115], [40], which we used for this paper. Following the standard *card sorting* method, we created one (virtual) card per reported challenge, and incrementally and iteratively organized those cards into groups of similar challenges. Multiple researchers went through all the cards in synchronous and asynchronous fashion to grasp the different concepts and identify relevant themes and clusters around the reported challenges. This being a collaborative effort, we did not aim for inter-rater agreement between independent grouping by individual researchers, but instead worked together as a team to build consensus. 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 when we found there was more than one theme, until all involved researchers were satisfied with the clusters and placement of the cards. 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. We performed this card sorting process in an online platform (miro.com), allowing us to manipulate colors, add different tags to the cards, add comments, emojis, and so on. We share the resulting card-sorting board as supplementary documents [76].

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, which provide the headings of our results presented in Section IV. We wanted to be concise, but also comprehensive to properly capture the themes in the card. At the same time, as Sandelowski and Barroso suggested [15], we were careful to preserve the context in which the findings appeared by going back and forth in the original papers when confusion arose, moving cards to other clusters or themes as needed.

Calculating effect sizes. Methods for meta-summaries recommend reporting the frequency of findings in the original sources [103]. 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 [36], [58]. 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").

C. Limitations and Threats to Validity

All research designs have limitations that threaten 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 do some judgment calls by consensus of all researchers for a number of papers, for example, whether to include [35], [2], [11], [6], [108].

As discussed earlier, the meta-summary synthesis method was chosen 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.

IV. Results

We report our findings of the meta-summary in this section 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.

A. 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 [6], [109], [55], [119], [51], [67], [118], [122], [44], [77], [127], [37], [61], [78], [99], [22], [85] (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 [44], [77] only consider paying for coding, as they do not understand the need for experimental analysis [61] and even difficulty convincing engineering teams to invest in collecting high-quality data [51]. 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 [122], [77], [37], [78], [22], [85]. 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 [55], [127], scoping and deciding the functional and non-functional requirements [119], [61], interpreting the model outcomes [109], [78], [99], and the infrastructure needs (e.g., appropriate data, monitoring infrastructure, retraining requirements) when building products [119], [127], [78]. 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 [119], [77], [94], [10].

Vagueness in ML problem specifications makes it difficult to map business goals to performance metrics [114], [60], [57], [109], [55], [119], [118], [122], [77], [94], [36], [61], [29], [78], [99], [65], [85] (17/50). Practitioners across many studies mention the challenge of formulating the specific software and ML problem in a way that satisfies business goals and objectives. ML practitioners find it difficult to map the high-level business goals to the low-level requirements for a model. While customers are broadly interested in improving the business, practitioners often find it difficult to quantify the contribution of the ML model and its return on investment. Also, *Responsible AI* initiatives find it difficult to quantify their contributions to the business, for example, measuring the value added by improving fairness and explainability, or to deliberate about tradeoffs between conflicting fairness and business objectives [10], [94], [36], [85]. Even with some notion of responsible AI requirements, practitioners find the requirements vague and not concrete enough to actually implement (e.g., unclear subpopulations and protected characteristics to balance discrimination) [119], [94]. 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 [109], [122], [61], [29].

Regulatory constraints specific to data and ML introduce additional requirements that restrict development [29], [108], [119], [10], [37], [81], [109] (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 [119], [37] 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 [108], [29], [10].

B. 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 [58], [109], [55], [67], [127], [61], [75], [65], [1], [73], [42] (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 [127], [61]. Building an end-to-end ML pipeline is considered to be a challenge due to the difficulties of integrating various ML and non-ML components in a system operating within an environment [58], [109], [55], the overwhelming complexity of integrating many tools and frameworks [67], [65], [73], the need for engineering skills beyond the comfort zone of some data scientists [73], 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 [109], [75], [65], [1], [42].

ML adds substantial design complexity with many, often implicit, data and tooling dependencies, and entanglements due to a lack of modularity [109], [110], [65], [114], [60], [58], [122], [127], [1], [4], [22] (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 [60], [58], [127], [22]. 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 [109], [65], [60], [122], [110], [1]. 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 [114], [122], [127]. Practitioners also point out that complexities arise due to a large amount of surrounding "glue code" to support the ML models [109], [4], and complicated dependency and configuration management [122], [4].

Difficulty in scaling model training and deployment on diverse hardware [109], [67], [61], [107], [29], [99], [110], [65], [42], [73] (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 [67], [61], [29], [99], [65], [73]. Among the common problems of such deployments, practitioners brought up the mismatch of development and production environments [61], [110], difficulties in building a scalable pipeline [107], [29], [65], [42], adhering to serving requirements such as latency and throughput [109], [65], as well as undocumented tribal knowledge within the team, hampering future deployments [110]. 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 [51], [2].

While monitorability and planning for change are often considered important, they are mostly considered only late after launching [109], [29], [110], [42], [58], [57], [127], [1], [55], [54], [77], [4], [10], [98], [73] (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 [77]. Even for companies that adopt a monitoring infrastructure, practitioners report struggling with ad-hoc monitoring practices of logging, creating alerts, or doing everything manually [110], [58]. 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 [109], [29], [42], [1], [4]. 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 [58], [127].

C. 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 [61], [29], [42], [57], [122], [55], [54], [4], [118], [51], [81], [7], [92], [130], [75], [28], [78], [2], [11] (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 [29], [7], (b) not automated enough [92], (c) requiring too much expert knowledge to be used [130], [57], [81], [92], (d) limited to specific tasks and types of data sets [7], [92], or (e) not suitable for their own problems [7], [11], [51]. 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 [110], [122], [77] (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.

D. Data Engineering

In developing ML 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 [109], [67], [61], [29], [99], [110], [65], [60], [1], [77], [4], [119], [51], [37], [92], [28], [73] (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 [28], [37], [77], [1], [92], [65], [61], [51]. Practitioners also mention their pain points in handling data errors and validating data quality, where better tool support is desired [107], [109], [110], [51], [99], [119], [60], [73]. 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 [99], [119], [92], [77], [37].

Internal data security and privacy policies restrict data access and use [67], [61], [29], [99], [65], [55], [77], [4], [51], [47] (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 [47], [61], [65], [29].

Although training-serving skew is common, many teams lack support for its required detection and monitoring [29], [110], [65], [57], [127], [77], [4] (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 [67], [107], [42], [1], [55], [118], [37] (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?" [42]*, but it's not possible for them to keep track of data and models across the life cycle without technological support [42], [1], [128]. 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 [37].

E. Quality Assurance

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

Testing and debugging ML models is difficult due to lack of specifications [109], [118], [44], [92], [130], [75], [28], [60], [57], [122], [77], [61], [107], [29], [99], [110], [65], [1], [4] (19/50). Practitioners find testing and debugging of ML models challenging. 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 [61], [29], [65], [60], [122], [77], [4], [44], [130], [28]. In particular, practitioners find it difficult to define accuracy thresholds for evaluations. Furthermore, practitioners find it difficult to select adequate test data, specifically curating test data of sufficient quality and quantity that is representative of the production environment [57], [122], [118], [92], [75]. 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 [60], [122], [44]. 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 [60], [122], [28]. 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 [110], [122], [28]. Additionally, practitioners raise challenges regarding evaluating model robustness, on one hand, suffering from the lack of a concrete methodology [92], [28], and on the other hand, having various metrics but no consensus on which metric to use [60].

Testing of model interactions, pipelines, and the entire system is considered challenging and often neglected [65], [60], [55], [77], [92], [130], [75], [28] (8/50). Testing literature often focuses on ML models and data quality, but less on how models are integrated into the system, 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 [60], [122], [130]. The lack of pipeline and system testing beyond the model is also considered a problematic area [55], [77], [92], [130], [75], [28]: 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 [130], [28], leading to system failures even where the model gives the correct results [75]. Practitioners also report having no systematic evaluation strategy nor automated tools and techniques for pipeline and system-level testing [60], [77].

Testing and monitoring models in production are considered important but difficult, and often not done [109], [110], [60], [77], [92] (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 to design online metrics that not only depend on the model but also on the external environment, user interactions after deployment, and the context of the product overall [110], [60]. They also find online testing time-consuming, as it requires longer observation periods to obtain meaningful results [109], [110]. Practitioners also state that there is no surefire strategy to precisely detect when the model is underperforming in online testing [110].

There are no standard processes or guidelines on how to assess system qualities such as fairness, security, and safety in practice [42], [54], [36], [118], [37], [98], [108], [10], [11] (9/50). Research often discusses how machine learning influences fairness, robustness, security, safety, and other qualities, but practitioners report that they find evaluating these as challenging. While practitioners consider these qualities important [42], [54], they often report having no effective methodology or concrete guidelines for evaluating them [54], [36], [118], [37], [98], [108], [11]. Even regarding fairness, which has received a lot of research attention lately, practitioners report finding it hard to apply auditing and debiasing methods due to not having a proper process in place [36], [37]. Some practitioners report waiting for complaints from customers rather than being proactive when it comes to fairness [36], or even blindly expecting the algorithms to inherently provide qualities like security against attacks [54].

F. Process

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

Development of products with ML component(s) is often ad-hoc, lacking well-defined processes [61], [29], [114], [122], [55], [4], [118], [44], [51], [75], [6] (11/50). Many practitioners report struggles finding a good process for developing ML components and products around them [61], [29], [114], [122], [44], often coming up with ad-hoc strategies and experiencing a lack of good engineering practices [44], [75]. 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 [61], [29], [6]. Also, they find it hard to set expectations for each sprint, as the project objectives may be unclear at the beginning and need to be revisited after the initial investigation [55], [6].

Uncertainty in ML development makes it hard to plan and estimate effort and time [61], [122], [4], [118], [44], [28], [6] (7/50). Modeling 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; instead, they commonly progress with many experiments with different algorithms and datasets [4], [44]. Practitioners thus report having difficulties setting expectations and (intermediate) deadlines for a project [61], [122], [4], [28], [6] and providing upfront effort and cost estimates [61], [4], [118].

Practitioners find documentation more important than ever in ML, but find it more challenging than traditional software documentation [61], [107], [29], [57], [77], [18], [53], [128], [88] (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 [29], [128]. 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 [29], [57], [128]. 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 [107], [77]. 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 [18], [53], [88]. 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 [18], [88].

G. *Organization and Teams*

Along with challenges faced in different development stages, practitioners also mention challenges 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 [109], [67], [122], [127], [4], [118], [123], [47], [75], [78], [6], [2] (12/50). Incorporation of ML in a product is not simply adding another component to the system; it requires people from multiple disciplines to get involved to support different aspects of this component. Teams require 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, operations, and domain expertise. The lack of this varied expertise in the team is commonly mentioned to be a challenge by practitioners [109], [122], [118], [47], [6], [2]. Also, as discussed in the next subsection that communication is often hindered by lack of AI literacy or common terminology [29], [1], [4], [75], [78], [127], cross-disciplinary knowledge seems to be important for team members to interact and understand each other's vocabulary; however, practitioner experiences indicate that such cross-disciplinary education is not broadly available yet [110], [28].

Many teams are not well prepared for the extensive interdisciplinary collaboration and communication needed in ML products [67], [107], [127], [77], [4], [7], [75], [78], [17], [22], [122] (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 [107], [122], [127], [78], [22]. 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 [29], [1], [4], [75]; software engineers lacking AI literacy [29], [75]; and data scientists and software engineers not understanding or interacting members with in with business roles [127], [78]. Practitioners report struggling with cultural differences, differences in expectations, and conflicting priorities [67], [4], [7], [78], and they often do not agree on assigned responsibilities [61], [77]. These multidisciplinary teams also suffer from miscommunications arising from inconsistency in their technical terminologies [77], [75], [17]. Siloing of teams by specialization and lack of communication across such silos are also observed in many production settings, fostering integration problems even further [77], [7].

ML development can be costly and resource limits can substantially curb/limit efforts [67], [110], [4], [47], [78], [2] (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 [54], [94], [81], [47], [98], [78], [108], [11] (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 [94], [36]. 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 [54], [81], [47], [98], [11].

V. Discussion and Conclusions

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.

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, the software-engineering literature is full of nuanced discussions of development life cycles and competing process models, but ML practitioners struggle adopting even the most flexible agile-inspired processes for their projects with the uncertainty that ML brings. Also, team collaboration and organizational challenges are well known in traditional software engineering, but those seem to become even more central with the additional complexity and inclusion of more people with different backgrounds, cultures, and priorities. Other challenges seem new, such as the data- and model-related challenges associated with ML components, and several of the reported challenges regarding architecture and quality assurance stemming from the different nature of reasoning in machine learning.

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. While we cannot provide a rigorous summary or analysis, we close by reflecting on possible directions.

- **Requirements Engineering.** For 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 ML projects [77], [95], [106], [119]. But again, while many practitioners mention suffering from unclear model requirements, we still do not seem to have a good solution, 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 [109], [124], [58], [56], 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 [31]) 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 [129], [97], [96], [38]. 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 development processes for ML models [116], [69], 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 a lot of progress can be made with better education and better adoption of good software engineering practices. There are plenty research opportunities to adapt existing practices, support them with tooling, and create new interventions altogether. We hope that the collection of challenges, which can be traced to the original studies where they were raised by practitioners, will be helpful in selecting and prioritizing research and education in our community.

Acknowledgments. Kästner's, Nahar's, and Zhang's work was supported in part by the National Science Foundation (#2131477), Zhou's work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC, RGPIN2021-03538), and Lewis' work was funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center (DM23-0228).

References

[1] Amershi, S. et al. 2019. Software Engineering for Machine Learning: A Case Study. *Proc. 41st Int'l Conf. on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, 291–300.

[2] Andrade, H., Lwakatare, L.E., Crnkovic, I. and Bosch, J. 2019. Software Challenges in Heterogeneous Computing: A Multiple Case Study in Industry. *Proc. 45th Euro. Int'l Conf. on Software Engineering and Advanced Applications (SEAA)*, 148–155.

[3] Arnold, M. et al. 2019. FactSheets: Increasing trust in AI services through supplier's declarations of conformity. *IBM Journal of R&D*. 63, 4/5, 6:1–6:13.

[4] Arpteg, A., Brinne, B., Crnkovic-Friis, L. and Bosch, J. 2018. Software Engineering Challenges of Deep Learning. *Proc. 44th Euro. Conf. on Software Engineering and Advanced Applications (SEAA)*, 50–59.

[5] Ashmore, R., Calinescu, R. and Paterson, C. 2022. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *ACM Computing Surveys*. 54, 5, 1–39.

[6] Baijens, J., Helms, R. and Iren, D. 2020. Applying Scrum in Data Science Projects. *Proc. 22nd Conf. on Business Informatics*, 30–38.

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

[8] Begel, A. and Zimmermann, T. 2014. Analyze this! 145 questions for data scientists in software engineering. *Proc. 36th Int'l Conf. on Software Engineering*, 12–23.

[9] Bernardi, L., Mavridis, T. and Estevez, P. 2019. 150 Successful Machine Learning Models: 6 Lessons Learned at Booking.com. *Proc. 25th ACM SIGKDD Int'l Conf. on Knowledge Discovery & Data Mining - KDD '19*, 1743–1751.

[10] Bhatt, U. et al. 2020. Explainable machine learning in deployment. *Proc. 2020 Conf. on Fairness, Accountability, and Transparency*, 648–657.

[11] Boenisch, F., Battis, V., Buchmann, N. and Poikela, M. 2021. "I Never Thought About Securing My Machine Learning Systems": A Study of Security and Privacy Awareness of Machine Learning Practitioners. *Proc. Mensch und Computer 2021*, 520–546.

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

[13] Borg, M. et al. 2020. 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, 1–19.

[14] Boyd, K.L. 2021. Datasheets for Datasets help ML Engineers Notice and Understand Ethical Issues in Training Data. *Proc. ACM on Human-Computer Interaction*. 5, CSCW2, 1–27.

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

[16] Breck, E., Cai, S., Nielsen, E., Salib, M. and Sculley, D. 2017. The ML test score: A rubric for ML production readiness and technical debt reduction. *Proc. 2017 IEEE Int'l Conf. on Big Data*, 1123–1132.

[17] Brennen, A. 2020. What Do People Really Want When They Say They Want "Explainable AI?" We Asked 60 Stakeholders. *Extd. Abs. of the 2020 CHI Conf. on Human Factors in Computing Systems*, 1–7.

[18] Chang, J. and Custis, C. 2022. Understanding Implementation Challenges in Machine Learning Documentation. *Equity and Access in Algorithms, Mechanisms, and Optimization*, 1–8.

[19] Chattopadhyay, S., Prasad, I., Henley, A.Z., Sarma, A. and Barik, T. 2020. What's wrong with computational notebooks? Pain points, needs, and design opportunities. *Proc. 2020 CHI Conf. on Human Factors in Computing Systems*, 1–12.

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

[21] Dilhara, M., Ketkar, A. and Dig, D. 2021. Understanding Software-2.0: A Study of Machine Learning Library Usage and Evolution. *ACM Trans. on Software Engineering and Methodology*. 30, 4, 1–42.

[22] Dove, G., Halskov, K., Forlizzi, J. and Zimmerman, J. 2017. UX Design Innovation: Challenges for Working with Machine Learning as a Design Material. *Proc. 2017 CHI Conf. on Human Factors in Computing Systems*, 278–288.

[23] Epperson, W., Wang, A.Y., DeLine, R. and Drucker, S.M. 2022. Strategies for Reuse and Sharing among Data Scientists in Software Teams. *Proc. 44th Int'l Conf. on Software Engineering: Software Engineering in Practice*, 243–252.

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

[25] Felizardo, K.R., Mendes, E., Kalinowski, M., Souza, É.F. and Vijaykumar, N.L. 2016. Using Forward Snowballing to update Systematic Reviews in Software Engineering. *Proc. 10th Int'l Symposium on Empirical Software Engineering and Measurement*, 1–6.

[26] Follow, R. Bridging the Gap Between Data Science & Engineer: Building High-Performance Teams.

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

[28] Golendukhina, V., Lenarduzzi, V. and Felderer, M. 2022. What is software quality for AI engineers? Towards a thinning of the fog. *Proc. 1st Int'l Conf. on AI Engineering: Software Engineering for AI*, 1–9.

[29] Haakman, M., Cruz, L., Huijgens, H. and van Deursen, A. 2021. AI Lifecycle Models Need To Be Revised. An Exploratory Study in Fintech. *Empirical Software Engineering*. 26, 5, 1–29.

[30] Harris, J. 2020. Beyond the jupyter notebook: how to build data science products. *Towards Data Science*.

[31] Hazelwood, K. et al. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. *Proc. 2018 Int'l Symposium on High Performance Computer Architecture (HPCA)*, 620–629.

[32] Head, A., Hohman, F., Barik, T., Drucker, S.M. and DeLine, R. 2019. Managing messes in computational notebooks. *Proc. 2019 CHI Conf. on Human Factors in Computing Systems - CHI '19*, 1–12.

[33] 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, 1–6.

[34] Hermann, J. and Del Balso, M. 2017. Meet Michelangelo: Uber's machine learning platform.

[35] Hill, C., Bellamy, R., Erickson, T. and Burnett, M. 2016. Trials and tribulations of developers of intelligent systems: A field study. *Proc. on Visual Languages and Human-Centric Computing (VL/HCC)*, 162–170.

[36] Holstein, K. et al. 2019. Improving Fairness in Machine Learning Systems: What Do Industry Practitioners Need? *Proc. 2019 CHI Conf. on Human Factors in Computing Systems*, 1–16.

[37] Hopkins, A. and Booth, S. 2021. Machine Learning Practices Outside Big Tech: How Resource Constraints Challenge Responsible Development. *Proc. Conf. on AI, Ethics, and Society*, 134–145.

[38] 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, 100270.

[39] Huang, X., Zhang, H., Zhou, X., Babar, M.A. and Yang, S. 2018. Synthesizing qualitative research in software engineering: a critical review. *Proc. 40th Int'l Conf. on Software Engineering*, 1207–1218.

[40] Hudson, W. 2013. Card Sorting. *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* The Interaction Design Foundation.

[41] Hulten, G. 2018. *Building Intelligent Systems: A Guide to Machine Learning Engineering*. Apress.

[42] Hummer, W. at al. 2019. ModelOps: Cloud-Based Lifecycle Management for Reliable and Trusted AI. *Proc. 2019 IEEE Int'l Conf. on Cloud Engineering (IC2E)*, 113–120.

[43] Hynes, N., Sculley, D. and Terry, M. 2017. The data linter: Lightweight, automated sanity checking for ML data sets. *NIPS MLSys Workshop*, 1-5.

[44] Ishikawa, F. and Yoshioka, N. 2019. How do engineers perceive difficulties in engineering of machine-learning systems? - questionnaire survey. *Proc. 2019 Joint 7th Int'l Workshop on Conducting Empirical Studies in Industry (CESI) and 6th Int'l Workshop on Software Engineering Research and Industrial Practice (SER&IP)*, 2–9.

[45] Jain, R. and Suman, U. 2015. A Systematic Literature Review on Global Software Development Life Cycle. *SIGSOFT Softw. Eng. Notes*. 40, 2, 1–14.

[46] Jentzsch, S. and Hochgeschwender, N. 2021. A qualitative study of Machine Learning practices and engineering challenges in Earth Observation. *it - Information Technology*. 63, 4, 235–247.

[47] John, M.M., Olsson, H.H. and Bosch, J. 2020. AI Deployment Architecture: Multi-Case Study for Key Factor Identification. *Proc. 27th Asia-Pacific Software Engineering Conf. (APSEC)*, 395–404.

[48] Kanagal, B. and Tata, S. 2018. Recommendations for All: Solving Thousands of Recommendation Problems Daily. *Proc. 34th Int'l Conf. on Data Engineering (ICDE)*, 1404–1413.

[49] Kästner, C. 2022. *Machine Learning in Production: From Models to Products*.

[50] Keele, S. 2007. *Guidelines for performing systematic literature reviews in software engineering*. Technical Rep., Ver. 2.3 EBSE Tech. Report.

[51] Kim, M., Zimmermann, T., DeLine, R. and Begel, A. 2018. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Transactions on Software Engineering*. 44, 11, 1024–1038.

[52] Kim, M., Zimmermann, T., DeLine, R. and Begel, A. 2016. The emerging role of data scientists on software development teams. *Proc. 38th Int'l Conf. on Software Engineering*, 96–107.

[53] Königstorfer, F. and Thalmann, S. 2022. AI Documentation: A path to accountability. *Journal of Responsible Technology*. 11, 100043.

[54] Kumar, R.S.S. et al. 2020. Adversarial Machine Learning - Industry Perspectives. *Proc. 2020 IEEE Security and Privacy Workshops (SPW).*, 69–75.

[55] Laato, S., Birkstedt, T., Mäantymäki, M., Minkkinen, M. and Mikkonen, T. 2022. AI governance in the system development life cycle: insights on responsible machine learning engineering. *Proc. 1st Int'l Conf. on AI Engineering: Software Engineering for AI*, 113–123.

[56] Lakshmanan, V., Robinson, S. and Munn, M. 2020. *Machine Learning Design Patterns*. O'Reilly Media, Inc.

[57] Lewis, G.A., Bellomo, S. and Ozkaya, I. 2021. Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. *Proc. IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI (WAIN)*, 133–140.

[58] Lewis, G.A., Ozkaya, I. and Xu, X. 2021. Software Architecture Challenges for ML Systems. *Proc. 2021 Int'l Conf. on Software Maintenance and Evolution (ICSME)*, 634–638.

[59] Lin, J. and Kolcz, A. 2012. Large-scale machine learning at Twitter. *Proc. Int'l Conf. on Management of Data*, 793–804.

[60] Li, S., Guo, J., Lou, J.-G., Fan, M., Liu, T. and Zhang, D. 2022. Testing machine learning systems in industry: an empirical study. *Proc. 44th Int'l Conf. on Software Engineering: Software Engineering in Practice*, 263–272.

[61] Liu, H., Eksmo, S., Risberg, J. and Hebig, R. 2020. Emerging and Changing Tasks in the Development Process for Machine Learning Systems. *Proc. Int'l Conf. on Software and System Processes*, 125–134.

[62] Liu, J., Boukhelifa, N. and Eagan, J.R. 2020. Understanding the Role of Alternatives in Data Analysis Practices. *IEEE transactions on visualization and computer graphics*. 26, 1, 66–76.

[63] Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S. and Leung, V.C.M. 2018. A Survey on Security Threats and Defensive Techniques of Machine Learning: A Data Driven View. *IEEE Access*. 6, 12103–12117.

[64] Lopez, G. and Guerrero, L.A. 2017. Awareness Supporting Technologies used in Collaborative Systems: A Systematic Literature Review. *Proc. 2017 ACM Int'l Conf. on Computer Supported Cooperative Work and Social Computing*, 808–820.

[65] Lwakatare, L.E., Raj, A., Bosch, J., Olsson, H.H. and Crnkovic, I. 2019. A taxonomy of software engineering challenges for machine learning systems: An empirical investigation. *Proc. 2019 Int'l Conf. on Agile Software Development*, 227–243.

[66] Lwakatare, L.E., Raj, A., Crnkovic, I., Bosch, J. and Olsson, H.H. 2020. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*. 127, 106368, 106368.

[67] Mäkinen, S., Skogström, H., Laaksonen, E. and Mikkonen, T. 2021. Who Needs MLOps: What Data Scientists Seek to Accomplish and How Can MLOps Help? *Proc. IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*, 109–112.

[68] Martínez-Fernández, S. et al. 2022. Software Engineering for AI-Based Systems: A Survey. *ACM Transactions on Software Engineering and Methodology*. 31, 2, 1–59.

[69] 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, 3048–3061.

[70] McGlohon, M. 2021. Demystifying Machine Learning in Production: Reasoning about a Large-Scale ML Platform.

[71] McGraw, G., Figueroa, H., Shepardson, V. and Bonett, R. 2020. An architectural risk analysis of machine learning systems: Toward more secure machine learning. *Berryville Institute of Machine Learning, Clarke County, VA. Accessed on: Mar*. 23.

[72] Mitchell, M. et al. 2019. Model Cards for Model Reporting. *Proc. Int'l Conf. on Fairness, Accountability, and Transparency*, 220–229.

[73] Muiruri, D., Lwakatare, L.E., K Nurminen, J. and Mikkonen, T. 2022. Practices and Infrastructures for ML Systems–An Interview Study in Finnish Organizations. *TechRxiv*.

[74] Muller, M. et al. 2019. How Data Science Workers Work with Data: Discovery, Capture, Curation, Design, Creation. *Proc. 2019 CHI Int'l Conf. on Human Factors in Computing Systems*, 1–15.

[75] Myllyaho, L. et al. 2022. On misbehaviour and fault tolerance in machine learning systems. *Journal of Systems and Software*. 183, 111096.

[76] Nahar, N. 2022. Supplementary documents: A meta-summary of challenges in building products with ML components – collecting experiences from 4758+ practitioners. OSF.

[77] Nahar, N., Zhou, S., Lewis, G. and Kästner, C. 2022. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process. *Proc. 44th Int'l Conf. on Software Engineering*, 413–425.

[78] Namvar, M., Intezari, A., Akhlaghpour, S. and Brienza, J.P. 2022. Beyond effective use: Integrating wise reasoning in machine learning development. *Int'l Journal of Information Management*.

[79] Nascimento, E., Nguyen-Duc, A., Sundbø, I. and Conte, T. 2020. Software engineering for artificial intelligence and machine learning software: A systematic literature review. *arXiv*.

[80] Nikanjam, A., Morovati, M.M., Khomh, F. and Ben Braiek, H. 2022. Faults in deep reinforcement learning programs: a taxonomy and a detection approach. *Automated software engineering*. 29, 1.

[81] Nikhil, K. et al. 2022. "If security is required": Engineering and Security Practices for Machine Learning-based IoT Devices. *Proc. 4th Int'l Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*, 1–8.

[82] Nushi, B., Kamar, E., Horvitz, E. and Kossmann, D. 2017. On human intellect and machine failures: troubleshooting integrative machine learning systems. *Proc. Thirty-First AAAI Conf. on Artificial Intelligence*, 1017–1025.

[83] Ozkaya, I. 2020. What is really different in engineering AI-enabled systems? *IEEE software*. 37, 4, 3–6.

[84] Paleyes, A., Urma, R.G. and Lawrence, N.D. 2022. Challenges in deploying machine learning: A survey of case studies. *ACM computing surveys*.

[85] Passi, S. and Jackson, S.J. 2018. Trust in Data Science: Collaboration, Translation, and Accountability in Corporate Data Science Projects. *Proc. ACM on Human-Computer Interaction*, 1–28.

[86] Passi, S. and Sengers, P. 2020. Making data science systems work. *Big data & society*. 7, 2, 205395172093960.

[87] Pimentel, J.F., Murta, L., Braganholo, V. and Freire, J. 2019. A large-scale study about quality and reproducibility of jupyter notebooks. *Proc. 16th Int'l Conf. on Mining Software Repositories*, 507–517.

[88] Piorkowski, D., González, D., Richards, J. and Houde, S. 2020. Towards evaluating and eliciting high-quality documentation for intelligent systems. *arXiv*.

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

[90] Polyzotis, N., Roy, S., Whang, S.E. and Zinkevich, M. 2018. Data Lifecycle Challenges in Production Machine Learning: A Survey. *ACM SIGMOD Record*. 47, 2, 17–28.

[91] Rahimi, M., Guo, J.L.C., Kokaly, S. and Chechik, M. 2019. Toward Requirements Specification for Machine-Learned Components. *Proc. 27th Int'l Requirements Engineering Workshops (REW)*, 241–244.

[92] Rahman, M.S. et al. 2021. Machine Learning Application Development: Practitioners' Insights. *arXiv*.

[93] Rahman, M.S., Rivera, E., Khomh, F., Guéhéneuc, Y.-G. and Lehnert, B. 2019. Machine Learning Software Engineering in Practice: An Industrial Case Study. *arXiv*.

[94] Rakova, B., Yang, J., Cramer, H. and Chowdhury, R. 2020. Where Responsible AI meets Reality: Practitioner Perspectives on Enablers for shifting Organizational Practices. *Proc. ACM on Human-Computer Interaction*, 1–23.

[95] Ribeiro, D.M., Cardoso, M., da Silva, F.Q.B. and França, C. 2014. Using qualitative metasummary to synthesize empirical findings in literature reviews. *Proc. 8th ACM/IEEE Int'l Symposium on Empirical Software Engineering and Measurement*, 1–4.

[96] Ribeiro, M.T., Wu, T., Guestrin, C. and Singh, S. 2020. Beyond Accuracy: Behavioral Testing of NLP models with CheckList. *arXiv*.

[97] Riccio, V. et al. 2020. Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering*. 25, 6, 5193–5254.

[98] Rismani, S. et al. and Rostamzadeh, N. 2022. From plane crashes to algorithmic harm: applicability of safety engineering frameworks for responsible ML. *arXiv*.

[99] Riungu-Kalliosaari, L., Kauppinen, M. and Männistö, T. 2017. What Can Be Learnt from Experienced Data Scientists? A Case Study. *Product-Focused Software Process Improvement*, 55–70.

[100] Saha, D. et al. 2020. Human Comprehension of Fairness in Machine Learning. *Proc. AAAI/ACM Int'l Conf. on AI, Ethics, and Society*, 152.

[101] Salay, R., Queiroz, R. and Czarnecki, K. 2017. An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software. *arXiv*.

[102] Sambasivan, N., Kapania, S., Highfill, H., Akrong, D., Paritosh, P. and Aroyo, L.M. 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. *Proc. 2021 CHI Int'l Conf. on Human Factors in Computing Systems*, 1–15.

[103] Sandelowski, M., Barroso, J. and Voils, C.I. 2007. Using qualitative metasummary to synthesize qualitative and quantitative descriptive findings. *Research in nursing & health*. 30, 1, 99–111.

[104] Sculley, D. et al. 2015. Hidden Technical Debt in Machine Learning Systems. *Advances in Neural Information Processing Systems 28*. C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, eds. Curran Associates, Inc. 2503–2511.

[105] Sculley, D., Otey, M.E., Pohl, M., Spitznagel, B., Hainsworth, J. and Zhou, Y. 2011. Detecting adversarial advertisements in the wild. *Proc. 17th ACM SIGKDD Int'l Conf. on Knowledge discovery and data mining*, 274–282.

[106] 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, e15182.

[107] Serban, A., van der Blom, K., Hoos, H. and Visser, J. 2020. Adoption and Effects of Software Engineering Best Practices in Machine Learning. *Proc. 14th ACM/IEEE Int'l Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–12.

[108] Serban, A., van der Blom, K., Hoos, H. and Visser, J. 2021. Practices for Engineering Trustworthy Machine Learning Applications. *Proc. 1st Workshop on AI Engineering-Software Engineering for AI*, 97–100.

[109] Serban, A. and Visser, J. 2022. Adapting Software Architectures to Machine Learning Challenges. *Proc. 2022 IEEE Int'l Conf. on Software Analysis, Evolution and Reengineering (SANER)*, 152–163.

[110] Shankar, S., Garcia, R., Hellerstein, J.M. and Parameswaran, A.G. 2022. Operationalizing Machine Learning: An Interview Study. *arXiv*.

[111] Shaw, M. and Zhu, L. 2022. Can Software Engineering Harness the Benefits of Advanced AI? *IEEE Software*. 39, 6, 99–104.

[112] Siebert, J., Joeckel, L., Heidrich, J., Nakamichi, K., Ohashi, K., Namba, I., Yamamoto, R. and Aoyama, M. 2020. Towards Guidelines for Assessing Qualities of Machine Learning Systems. *Proc. Int'l Conf. on the Quality of Information and Communications Technology*, 17–31.

[113] Smith, D. 2017. Exploring development patterns in data science.

[114] d. S. Nascimento, E., Ahmed, I., Oliveira, E., Palheta, M.P., Steinmacher, I. and Conte, T. 2019. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. *Proc. 2019 ACM/IEEE Int'l Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–6.

[115] Spencer, D. 2009. *Card Sorting: Designing Usable Categories*. Rosenfeld Media.

[116] Studer, S., Bui, T.B., Drescher, C., Hanuschkin, A., Winkler, L., Peters, S. and Mueller, K.-R. 2021. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Machine Learning and Knowledge Extraction*. 3, 2, 392–413.

[117] Tonekaboni, S., Joshi, S., McCradden, M.D. and Goldenberg, A. 2019. What Clinicians Want: Contextualizing Explainable Machine Learning for Clinical End Use. *Proc. 4th Machine Learning for Healthcare Conference*, 359–380.

[118] Uchihira, N. 2022. Project FMEA for Recognizing Difficulties in Machine Learning Application System Development. *Proc. Int'l Conf. on Management of Engineering and Technology (PICMET)*, 1–8.

[119] Vogelsang, A. and Borg, M. 2019. Requirements Engineering for Machine Learning: Perspectives from Data Scientists. *Proc. 27th Int'l Requirements Engineering Workshops (REW)*, 245–251.

[120] Wagstaff, K. 2012. Machine Learning that Matters. *arXiv*.

[121] Wang, D. et al. 2019. Human-AI Collaboration in Data Science: Exploring Data Scientists' Perceptions of Automated AI. *Proc. ACM on Human-Computer Interaction*. 3, CSCW, 1–24.

[122] Wan, Z., Xia, X., Lo, D. and Murphy, G.C. 2019. How does Machine Learning Change Software Development Practices? *IEEE Transactions on Software Engineering*. 47, 9, 1857–1871.

[123] Washizaki, H., Takeuchi, H., Khomh, F., Natori, N., Doi, T. and Okuda, S. 2020. Practitioners' insights on machine-learning software engineering design patterns: a preliminary study. *Proc. 2020 IEEE Int'l Conf. on Software Maintenance and Evolution (ICSME)*, 797–799.

[124] Washizaki, H., Uchida, H., Khomh, F. and Guéhéneuc, Y.-G. 2020. Machine learning architecture and design patterns. *IEEE Software*. 8.

[125] Washizaki, H., Uchida, H., Khomh, F. and Guéhéneuc, Y.-G. 2019. Studying Software Engineering Patterns for Designing Machine Learning Systems. *Proc. 10th Int'l Workshop on Empirical Software Engineering in Practice (IWESEP)*, 49–495.

[126] Wohlin, C. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. *Proc. 18th Int'l Conf. on Evaluation and Assessment in Software Engineering*, 1–10.

[127] Zdanowska, S. and Taylor, A.S. 2022. A study of UX practitioners roles in designing real-world, enterprise ML systems. *Proc. 2022 CHI Int'l Conf. on Human Factors in Computing Systems*, 1–15.

[128] Zhang, A.X., Muller, M. and Wang, D. 2020. How do data science workers collaborate? Roles, workflows, and tools. *Proc. ACM on human-computer interaction*. 4, CSCW1, 1–23.

[129] Zhang, J.M., Harman, M., Ma, L. and Liu, Y. 2022. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*. 48, 1, 1–36.

[130] Zhang, X., Yang, Y., Feng, Y. and Chen, Z. 2019. Software Engineering Practice in the Development of Deep Learning Applications. *arXiv*.

[131] Zinkevich, M. 2017. Rules of machine learning: Best practices for ML engineering. minegrado.ovh.