Socio-Technical Anti-Patterns in Building ML-Enabled Software

Insights from Leaders on the Forefront

Alina Mailach
Leipzig University
ScaDS.AI Dresden/Leipzig

Norbert Siegmund
Leipzig University
ScaDS.AI Dresden/Leipzig

Abstract—Although machine learning (ML)-enabled software systems seem to be a success story considering their rise in economic power, there are consistent reports from companies and practitioners struggling to bring ML models into production. Many papers have focused on specific, and purely technical aspects, such as testing and pipelines, but only few on socio-technical aspects.

Driven by numerous anecdotes and reports from practitioners, our goal is to collect and analyze socio-technical challenges of productionizing ML models centered around and within teams. To this end, we conducted the largest qualitative empirical study in this area, involving the manual analysis of 66 hours of talks that have been recorded by the MLOps community.

By analyzing talks from practitioners for practitioners of a community with over 11,000 members in their Slack workspace, we found 17 anti-patterns, often rooted in organizational or management problems. We further list recommendations to overcome these problems, ranging from technical solutions over guidelines to organizational restructuring. Finally, we contextualize our findings with previous research, confirming existing results, validating our own, and highlighting new insights.

I. INTRODUCTION

“...So many guests have come on [at MLOps Community Channel] and said that MLOps is an organizational problem. It’s not a technology problem.” M46

In just a few years, machine learning (ML)-enabled software systems became ubiquitous in our daily lives. The huge number of existing applications empowered with cognitive and visual AI capabilities and the emergence of entirely new business domains based on ML draw an indisputable success story. Yet, we see only the fraction of products that made it from development to production [1], [2].

There is a large body of work providing explanations to this productionization challenge of ML models, such as technical debt during development [3], automation and pipeline challenges in the area of MLOps [4], [5], and testing and debugging [6], [7], [8]. However, often these works are focusing on a pure technical level. This is surprising as building ML systems is usually a multi-person, or even a multi-team project [9]. Gail Murphy framed the lack of such studies in her recent keynote “Is software engineering research addressing software engineering problems?” at ASE’20 [10].

Only few papers address the socio-technical challenges. For example, Granlund and others have found profound organizational challenges when integrating and scaling ML applications across organizations [4]. Most notably, Nahar and others have investigated the intersection of software systems and ML models focusing on collaboration between teams [11]. They found collaboration challenges among and within teams through miscommunications, a lack of documentation, non-valued engineering, and unclear processes. Such studies provide a rich, yet incomplete picture on the social and organizational aspects of building ML-enabled software. Anecdotal evidence from blog posts, tweets, and (video) podcasts, such as in Figure 1 hint that not solely tools and technologies are main factors for failed projects, but management, organization, and social aspects. Further, so called gray literature has been shown to contain valuable insights to enrich literature reviews [12]. It is, thus, surprising that there is no systematic study that lifts this huge body of knowledge provided by practitioners who encounter these problems in the first place and find their insights valuable to share and discuss with each other.

We set out to explore what experts and leaders report on socio-technical challenges when building ML-enabled software. Specifically, we state our research question around the organization and management of teams and persons: What socio-technical challenges and what kind of solutions do practitioners discuss in regard to productionizing machine learning?
To get unbiased opinions and discussions around the development of ML-enabled systems for answering our research question, we searched for an interest group whose members are focusing on ML-system development in a holistic way, that is, they are not focused on specific stages of the development process, represent a wide diversity of members in terms of roles, company size, and country of residence, and have a culture of sharing their activities in a suitable format. The biggest interest group that fulfills all of these criteria is the MLOps.community[1]. A community with more than 11,000 professional members dedicated to share experiences when putting ML into production.

We employ a research methodology known as reflexive thematic analysis [13], [14] to identify thematic patterns and interpret them. We manually analyzed in a qualitative empirical study 73 videos published by the MLOps community, thereby analyzing more than 66 hours of video material[2], making it the largest qualitative study we are aware of (cf. [15], [16], [17]). This enormous corpus of data gives us insights from a large variety of companies with different sizes and diverse business goals. Since the practitioners originate from five different continents, we obtain a broad picture of socio-technical challenges, involving different cultures.

Our main contributions are the following: We found 17 anti-patterns whose causes are mainly of non-technical nature. That is, the reported anti-patterns of practitioners often originate from a lack of organizational decisions and guidelines, and undefined organizational processes as well as miscommunication between management and the development team. Specifically, we could identify three main areas in the organization of ML projects causing problems: leadership vacuum at management level, organizational silos, and communication within an organization. Practitioners mention that tools and technologies may mitigate the symptoms of anti-patterns, but the causes lie too often within the organization itself. This draws parallels to similar observations in traditional software development, such as Conway’s Law [18], [19].

As second contribution, we contextualize previous research by discussing how challenges found before [11] can be traced to our identified anti-patterns and their causes. Thereby, we not only produce novel anti-patterns, but also confirm from another data source existing ones, strengthening the generalizability and trustworthiness of our and prior work in this area. We could enrich former findings due to the unique opportunity of evaluating interviews of mostly leaders (CEOs and managers) instead of developers. This novel perspective from a management point of view is a missing piece of a puzzle to have a holistic view on the development activities and challenges in building ML-enabled software. Furthermore, confirming and extending on former findings shows how practitioner communities can be leveraged to gain insights from samples with special characteristics and size that are otherwise barely obtainable.

Finally, we report back possible best practices to address the mentioned causes. Overall, we found substantial evidence that socio-technical problems are often the root cause of many issues that practitioners and research alike aim to address with technological solutions. By framing these problems as anti-patterns and rooting them at the social and organizational level, we raise awareness of this underrepresented line of research.

II. METHODOLOGY

To answer our research question, we conducted a qualitative empirical study. We investigated 73 out of a total of 210 videos published by the MLOps community using reflexive thematic analysis (RTA) [13], [14], [20] to identify and analyse thematic patterns. RTA is a fully qualitative method from the family of thematic analysis and is widely used in psychological research, but becomes increasingly adopted in other fields, such as human computer interaction [21]. In contrast to other methods for thematic analysis, such as coding reliability approaches, RTA allows researchers to engage with the data in an analytical manner rather than just providing summaries of what was said [22]. This was necessary in our case, since we could not rely on a structured questionnaire that would prompt practitioners to speak about certain content (as in an interview study).

A key feature of RTA is ‘late’ theme development, meaning that themes are generated from codes analytically. Themes represent shared structures, underpinned by a key concept [13], [22]. Braun & Clarke outline 6 steps in the analytical process: familiarization with the data (1), coding (2), initial theme development (3), developing, reviewing and refining themes (4), defining and naming themes (5), and writing (6), which are run through recursively. This way, RTA enables to systematically analyze diverse and inconsistent data, such as conversations and, thus, gives larger flexibility to this kind of data compared to, for example, grounded theory and related methodologies [22]. Since we use an existing corpus of unstructured interviews and talks, this research methodology is ideal for such a use case.

We apply RTA by having an inductive (bottom-up) phase and a deductive (top-down) phase as displayed in Figure 2. We provide the interim selection of codes that has been used (and refined) during the deductive phase and the later structure of themes as supplementary material. Note that codes and themes evolve and change during the interpretation and iteration (one clearly does not start with the final themes). The interim codes represent the stage of our research when we

[1] https://mlops.community/
[2] This roughly maps to 1,200 pages of automatic generated transcripts.
identified that each theme consists of multiple symptoms (later evolved to anti-patterns), causes, and solutions (later evolved to recommendations). Next, we explain our source material, demographics, and the phases of our methodology in detail.

A. The MLOps community

The MLOps community is an interest group founded in 2020 that is centered around knowledge sharing. Its major output formats are Meetups and Coffee Sessions, both held online and streamed via YouTube. Although the community’s notion centers around pipelines and MLOps, the channels and interviews cover the whole development process, which makes them an ideal base for our analysis.

B. Induction phase

From the full set of 210 videos, we took 82 from the ‘Full Talks’ playlist on the respective YouTube channel. Due to the long playtime of the videos and the relative little information contained in the video description, we conducted an initial filtering. For each video, we searched for the keyword ‘team’ in the automatically generated transcripts and manually evaluated whether the speaker is speaking about socio-technical challenges. Although this singular keyword may seem too limited at first, we argue that socio-technical challenges arise around and within teams. Other non-technical aspects, such as ethics or legal concerns, that also occur in the development process are out of scope of our analysis. After the filtering, we yielded 37 videos from which we expected to contain at least one part relevant to our research question.

Following the guide of Braun & Clarke [13], we familiarized ourselves with the video material while simultaneously correcting the automatically generated transcript (1). We used the identified relevant sections of the videos as starting points and transcribed all relevant content before and after to fully understand the context. We then started to develop initial codes (2) and already started to generate initial themes (3). In regular confirmation and validation meetings, we entered phase 4, in which we reviewed and refined our initial themes. Here, we reflected on differences and similarities in our perspectives, resulting in an iterative refinement of the themes. We already started with defining and naming themes (5) during this induction phase, which helped identify that structuring the initial themes into the presented form of anti-patterns/causes and solutions is meaningful. For further iterations, we used the form of symptoms, causes, and solutions in our code collection. The rationale was that the themes we identified so far, had more than one challenge included but not necessarily all speakers mentioned each challenge. Some of them just described issues that we identify as causes. The initial themes developed to be the seven contexts and development activities presented in this paper.

C. Deduction and refinement phase

To further engage with the remaining videos, we diverged from the original proposal of Braun & Clark and used our extracted codes to identify relevant videos in the remaining 128 videos. We therefore read title and description of the videos as well as their topic tags, which are provided by the uploader for some recordings to label different sections of the videos. Another 38 videos were marked as relevant and we started again familiarizing with the data by correcting the automatic transcriptions (1). This way, we reduced the threat to bias our selection against our initial keyword. We then used our initial codes and themes to find similar passages within the videos. The goal of this phase was enriching and refining the themes with the additional data. Although we run through earlier steps (2-3), we focused mainly on refining (4), defining and naming (5), and writing (6). We finally arrived at our structure of anti-patterns and causes. We reviewed all relevant passages again and extracted practitioners’ recommendations.

We maintained the tracing from a context or development activity to the according videos (cf. Table I). This allows us to mark a theme as ‘thick’ (i.e., often discussed) or ‘thin’ (i.e., rarely discussed). However, we refrain from a more granular mapping to individual causes or anti-patterns because interviewees reported patterns and causes in different levels of detail. Some only enumerate problems whereas others provide detailed discussions. So, while a quantitative weighing of a theme’s importance may provide additional insights, we are careful to not lead the reader into misinterpreting frequencies with severity of challenges. Hence, we conservatively follow the fully qualitative approach of RTA and refrain from providing more detailed numbers.

D. Demographics

The final sample consists of 73 videos with a total of 66 hours of conversations and talks with a mean number of views being 628. The oldest video was recorded in April 2020 and the most recent video has been recorded in May 2022 and represents the end of our data collection. For each video of the original 210 videos, we used the YouTube-Transcription-API to derive automatic transcripts for an initial filtering step.

The speakers in the videos come from different, mainly US-American and European countries and mostly hold leading positions. Often, guests do not solely talk about their current position, but also report stories from former employees, colleagues, and discussions in the Slack community channels.

<table>
<thead>
<tr>
<th>Context</th>
<th>Meetups [M]</th>
<th>Coffee Sessions [CS]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model to product integration</td>
<td>3, 10, 17, 21, 35, 36, 43, 46, 47, 54, 56, 57, 64, 75, 73, 90, 84, 92, 94, 95, 1</td>
<td>20, 34, 35, 44, 44, 62, 69, 17, 18, 21, 23, 35, 38, 39, 43, 49, 54, 55, 56, 57, 64, 75, 73, 90, 84, 92, 94, 95, 1</td>
</tr>
<tr>
<td>Data producer to consumer inte-</td>
<td>10, 44, 46, 59, 64, 94</td>
<td>19, 26, 38, 39, 65, 89</td>
</tr>
<tr>
<td>gration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Redundant development</td>
<td>22, 29, 49, 68</td>
<td>13, 23, 26, 44, 49, 67</td>
</tr>
<tr>
<td>Headless-chicken-hiring</td>
<td>10, 37, 46, 49</td>
<td>19, 23, 25, 27, 38, 40, 44, 81, 89, 95</td>
</tr>
<tr>
<td>Résumé-driven-development</td>
<td>5, 30, 38, 85, 86</td>
<td></td>
</tr>
<tr>
<td>Hype-driven product creation</td>
<td>5, 10, 11, 30, 37, 45, 46, 49, 54, 55, 59, 60, 61, 62, 44, 46, 56, 67, 75, 85, 87, 68, 69</td>
<td>88, 89, 93, 94</td>
</tr>
<tr>
<td>Management vs. Data Science</td>
<td>29, 30, 58, 64, 65</td>
<td>6, 21</td>
</tr>
</tbody>
</table>
That is, we do not necessarily see them as sole representatives for their respective companies, but rather as individuals experiencing the move from traditional software to ML-enabled software.

E. Threats to validity

Qualitative studies are naturally threatened by a subjective analysis process. That is, we could have focused too heavily on specific topics while missing others. This includes the manual encoding and generation of themes. RTA emphasizes that themes are an interpretation of codes and should not be seen as independent from the researchers [13], [20], [22]. We held regular meetings in which we focused on building a rich and common understanding of the data and reflected on our process and assumptions. Further, we put our results into perspective with prior work, thereby validating the anti-patterns and causes found in our study.

The initial selection of the keyword and relying only on a single community may pose an additional threat to external validity. With its over 11,000 members, spanning several continents, we are convinced that this represents one of the most generalizable sources available, especially when compared to singular company papers. Moreover, by having a two-step process, we extend our initial filtering process with topics mentioned around teams, such that further socio-technical challenges appear. Our surprisingly high alignment with findings of other papers that use structured interviews provide further evidence of the validity of our findings.

F. Results Overview

We group our findings into three main areas: organizational silos, communication within an organization, and organizational leadership vacuum. In these areas, we found 7 contexts or development activities in which, in total, 17 anti-patterns can emerge. The next sections are structured as follows: We first describe the context of reported anti-patterns. Then, we list all corresponding anti-patterns (denoted with AP) together with citations that are exemplary for our findings. Afterwards, we describe their causes (denoted with C) and suggested recommendations (denoted with R). Finally, we put our results into perspective with prior work in Section VI, thereby, confirming earlier findings and validating ours.

III. ORGANIZATIONAL SILOS

Organization silos often manifest when multiple teams work on disjoint tasks, challenging data and model transfer.

A. Model to production integration

“How do I hand off this model to this other team? And there’s a lot that goes into it, it’s the actual version of the code to instantiate the model, it’s the right set of weights. You probably want some way to track metadata associated with it, in case there are scores to be able to interact programmatically. And so at its core, this suddenly goes from what sounds like it should be a pretty easy problem to ‘I’m spending weeks and meetings and emails trying to physically figure out and help this person instantiate my model and load my weights.’ [...] And then, it gets worse because you try to build a new version of the model and instead of that person being able to grab the new version, they have to go through this whole stupid cycle again.” [M36]

Many times practitioners describe the process of getting an already developed model into a production stage as especially tedious and error-prone. This problem was reported to occur across all stages of company maturity. A model is developed by either a single data scientist (integrated team, as defined by Nahar and others [11]) or a dedicated data science team (center-of-competence). However, the developing party is always dependant on another organizational entity (e.g. another team member or another team) to integrate and deploy said model (model handover). Furthermore, the model handover is not only a handover of the model as such, but oftentimes includes feature pipelines that have to be implemented in ways that differ substantially between development and production environment. The interdependence between the two organizational entities increases when the code bases (i.e. model development vs. production) differ in terms of languages and tools. This can lead to the necessity to reimplement whole pipelines, which then have to be validated again by another person or team. The following anti-patterns have been observed independent of whether the model team or an engineering team was responsible for productionizing the model.

(API) Long release cycles: Symptomatic outcomes are long release cycles, back and forth between model developing and product team or even the complete inability to productionize existing models. The two teams are tightly coupled and spend a lot of time discussing all necessary details. While direct communication is often times preferable, it can become an obstacle if there are no code abstractions, documentations, and clear processes defined so that the teams get together for every detail in every new version. In some cases, the whole model has to be rewritten by the product team, leading to long intermittent phases between two versions as well as technical challenges such as assuring that the rewritten model holds the same properties as the original one.

(AP2) Tension between teams: The collaboration between the model developing entity (often data scientists) and the entity that is responsible for putting it into production (often software engineers) is characterized by reciprocal complaints that are usually driven by a lack of understanding. The speaker

<table>
<thead>
<tr>
<th>Roles</th>
<th>Management (director, head, chief, CFO, CEO, founder) (31.3%), leader (+ principal &amp; resident) (27.6%), data scientists &amp; engineers (+ data-, ML-, MLOps-engineers) (22.7%), product management (7.6%), other (10.8%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>North America (50.8%), Europe (39.2%), South America (6.8%), Asia (1.4%), Australia (2.7%)</td>
</tr>
<tr>
<td>Size</td>
<td>Large (40.3%), mid-size (16.9%), small or startup (35.1%), other (7.8%)</td>
</tr>
<tr>
<td>Sector</td>
<td>Technology (31.1%), consultancy (14.9%), ML-platform (14.9%), e-commerce (9.5%), banking &amp; fintech (8.1%), venture capital (6.8%), other (13.5%)</td>
</tr>
</tbody>
</table>

TABLE II

DEMOGRAPHICS

Company characteristics

| Location | North America (50.8%), Europe (39.2%), South America (6.8%), Asia (1.4%), Australia (2.7%) |
| Size | Large (40.3%), mid-size (16.9%), small or startup (35.1%), other (7.8%) |
| Sector | Technology (31.1%), consultancy (14.9%), ML-platform (14.9%), e-commerce (9.5%), banking & fintech (8.1%), venture capital (6.8%), other (13.5%) |

Speaker characteristics

<table>
<thead>
<tr>
<th>Roles</th>
<th>Management (director, head, chief, CFO, CEO, founder) (31.3%), leader (+ principal &amp; resident) (27.6%), data scientists &amp; engineers (+ data-, ML-, MLOps-engineers) (22.7%), product management (7.6%), other (10.8%)</th>
</tr>
</thead>
</table>
in M3 gives a brief example of what communication between the two entities looks like in practice: “So, you had these multiple jumps and it was a daily exercise of complaining about the other person. So the software engineer would complain to the data scientist saying ‘I don’t understand what’s going on here. It’s crazy, it’s too complicated’ and the data scientist is going to the software engineer ‘It’s not complicated, it’s dead easy, how can you not understand it?’” _M3_.

(AP3) Blocked data scientists: Long release cycles and the tight coupling of the two teams implies that data scientists are not able to focus on developing or improving new models, but invest their time in working on the production side. The same is the case when the data science team is supposed to productionize the model themselves: While model development is often seen as the core task of a data scientist, they are now engaged with the production part and are therefore no longer developing or optimizing new models.

(C1) Clash of cultures and tools: Speakers see the causes for painful handovers in the inherently different cultures of software engineering and data science, where both have their own values and expectations. For example, model development is usually research-oriented, leading to a reported lack of product-centered thinking in data science. Artifacts, produced in the model development phase, are often not tested and do not follow code conventions. Moreover, both fields use different tools, languages, and frameworks, leading to different terms and concepts, which further enforces communication issues. “One of the challenges I have with data scientists is to understand the kind of language they use […]” _M3_.

(C2) Missing engineering competence: Data scientists and software engineers alike lack skills to sufficiently understand the other side. Software engineering practices, such as code quality, design patterns, automation pipelines, virtualization, and scalable infrastructure are often unknown to data scientists. On the other hand, the software team operationalizing the model is missing key data science skills and is thus treating the model as a black box, which, in turn, necessitates check backs from the engineering team; all of this leading to production delays.

(R1) Model registry and feature store: A common approach is easing the symptoms of anti-patterns instead of targeting the causes. That is, technological solutions can circumvent some of the issues existing when integrating a model to production. Especially, model registry and feature stores enable a cleaner interface between organizational entities and a clear contract in terms of documentation and standards between them. Model registries are repositories in which the model artifacts are stored together with all relevant information (e.g. code, data, parameters, metrics). Feature stores, on the other hand, enable versioning of feature calculation for training and production in a central database, eliminating mistakes and long validation cycles. By using these tools, the organizational entities are forced to work in systematic and structured ways. This is also possible without these tools, but requires non-technical solutions, such as guidelines and contracts.

(R2) Restructure teams to be cross-functional: While tech-

ological solutions focus on bridging existing gaps, many speakers mention that an organizational restructuring from functional teams towards cross-functional or integrated teams have helped them to bridge the gap between data scientists and software engineers. When working closely together, people are likely to share skills and develop a common understanding of the product they are developing.

(R3) Pair data science and software engineers: Especially for helping data scientists to write code of high quality that adheres to the standards of the software engineer (i.e., “production code”), pairing and code reviews are two non-technical solutions mentioned. While this can be performed after a restructuring within each of the teams, the speaker in M58 advises to also perform this kind of exchange between functional teams: “[..] a great way for us to upskill the data science team is to pair with them work on a project […] and show them how to do that correctly and why it’s important. I’ve never seen much success with a hostile engagement like blaming, finger-pointing. It’s not productive.” _M58_

(R4) Translation work between the different roles and common understanding of the goal: Different fields and values lead to different vocabularies used by their representatives. That is why practitioners recommend addressing this by either building a common dictionary with terms and a corresponding definition or introducing roles that have an understanding of both fields and act as a mediator between them. Having a common language across different roles and teams enables understanding of products as shared responsibility and fosters empathy for the individual challenges each of the fields has.

B. Data producer to consumer integration

Larger companies often have multiple departments and are organized in ways to treat them as distinct business units. Hence, there are organizational units that produce data (e.g., monitoring industrial processes, gathering customer data) and store them in their own backend systems. Especially ML teams require to consume that data to build new models. The separation of the two parties result in several anti-patterns.

(AP4) Requesting data is hard: Symptomatic for this pattern is the restricted access to data for the consumer by the producer. Practitioners describe that it is hard to convince the producing organizational entity to expose data or fetch specific data points that would be necessary for the consumer’s use case. Sometimes, the data producing team is not directly involved in these use cases and therefore fetching, exposing, or fixing data points are constantly weighed against the use case of the producing team. Hence, requests are often shut
down or unanswered since providing data often raises costs for the producer without any benefit.

(AP5) Tight technical coupling between consumer and producer team: An unclear handover process of data leads to close technological entanglement between teams. This results in code and data dependencies across organizational units. “Feature calculation against operational databases, this is a really horrible one, that I never want to repeat in my life again. Both from computational points of view but also from just separating out contracts.” M46. Further, implicit assumptions by the producer team can lead to silent changes of definitions or calculation of certain data points, which is not immediately visible in the outcome on the consumer’s side. These silent changes can lead to false predictions of data services further down the line and make debugging really hard.

(AP6) Partial data availability: Incomplete data available to the data science team may result in differences between training and production. If this is unknown, a deployed model may likely suffer a degraded performance as the distribution of training data does not reflect the data distribution in production. “He was creating a product, and when he was doing his training data, everything was fine. But then when it went live, this machine learning algorithm wasn’t getting the full story all of a sudden. Later he found out just by coincidence that his training data was just one stream of the whole picture. So, it was like ‘Oh wait a minute! That’s why everything’s going haywire when I put it out live!'” CS19

(C3) Lack of awareness and common goals: Practitioners mention that the organizational division between consumers and producers, where each one is following their own use case, is a major cause for the problems in collaboration between the two entities. The producer lacks awareness and concern about the data and does not see the value in spending time and effort on fetching and maintaining certain data points. Further, the consumer’s requests can be in conflict with the producer’s own product and success metrics. It is even possible that the producing part is not aware of any downstream dependencies of their data output which enforces silent failures: “How do you give them [the producer] the right feedback loop, so that they understand that if they change something it’s potentially a breaking change for a different unit somewhere else, because they [the consumer] use that data as an input data for some of their machine learning systems.” CS89

(C4) Missing documentation and unclear responsibilities: Especially in larger organizations with a centralized data structure (e.g. a data lake or a data warehouse), it can become intransparent who produced the data and who is therefore responsible for maintaining it. Furthermore, definitions might lack the necessary clarity, which leads consumers to either not use the data or produce results which can not be trusted. Such a lack of proper documentation makes it hard for consumers to trust the data and exposes them to the risk of having misbehaving models during production where only by chance one can find the cause in the data sources.

(R5) Raise awareness of data producers: Changing the mindset of data producers towards understanding how important their data can be for other teams is a common recommendation. Some practitioners mention that this remains an unsolved challenge within their company; education and feedback are mentioned as possible paths. Education focuses on emphasizing the role each team plays within the company’s wider scope and what kind of consumers exist, including their specific needs and risks. Further, a feedback mechanism that shows the utility of the data by the consumer and also enables the producers to see the influences of their changes directly.

(R6) Central platform as a contract between producer and consumer: A common recommendation was to use a central repository or platform to allow for discovering and accessing of data. Still, missing documentation and trust in the centralized data as well as the high level of structural independence needs to be addressed. Therefore, speakers suggest that the platform itself should act as a contract, including quality metrics and documentation of the ingested data.

IV. Communication

Missing communication has been reported to be a major problem for causing a redundant development of features, tools, and infrastructure in organizations as well as for a source of tension between the data science team and the management.

A. Redundant development

“Let’s say you have two models which are quite similar. Both use the same feature. Both are developed in different teams. And then, those teams start to develop exactly the same feature in slightly different ways but semantically with the same meaning.” M29

Companies with multiple teams and multiple use cases frequently report the situation that similar features (i.e., preprocessed raw data acting as inputs to the ML model) are subsumed by multiple models. Ideally, those features should be defined and preprocessed only once, but, in practice, are redundantly developed, wasting time and resources. Even worse, this situation is not unique to features, but redundant infrastructure and tool development, such as pipelines, database systems, scaling infrastructure, has been reported to potentially occur in every company with multiple use cases.

(AP7) Features are not discoverable, accessible, or reusable: Different teams are developing semantically similar features, but the features can not be reused. Even if they are used in different teams, there is no direct connection. That is, when a feature is updated or the constraints on the calculation change, every team has to correct them, leading to repetitive work. However, it has also been reported that features are given similar names, but have dissimilar definitions. So, the interpretation of domain concepts that may result in ML features is subjective to the team’s perspective and use case.

(AP8) Redundant ML infrastructure and tools: “I’m sure the tradition is: I have a problem, I build a model. I want to keep updating my model. I’m going to build some infrastructure to update my model. Now, I’m an infrastructure team for my model and so is everyone else.” CS23. Building an ML model usually requires an infrastructure for obtaining
and preprocessing raw data in a scalable way as well as conducting experiments and training runs. This leads to redundant infrastructure and the accompanied tools in the organization.

(AP9) Shadow IT: Since every team is developing their own infrastructure, there is a chance that the development happens although there is a central infrastructure provider who is normally responsible for hosting and maintaining infrastructure. The disconnect between the teams and the infrastructure provider leads to ungovernable self-hosted IT within the teams. Sometimes, this means that organizational security standards, such as protection of APIs, are ignored and, therefore, especially in the context of data, holds a major risk to privacy and security.

(C5) Missing intersections between teams, documentation, and trust: In this decentralized scenario, all teams work on different use cases independent from each other. The absence of a natural intersection between the teams leads to little or no communication, meaning that the teams do not know about each other’s developments (e.g., features or tools). Even if the developments of a team are potentially discoverable by other teams, it does not imply their usability. The isolation of the teams and use cases does not set the required incentives to build and document in a reusable fashion. This leads to missing trust by potential (re-)users of the features or infrastructure: “Can I actually reuse all those existing features that other team has developed? How do I get that trust and confidence that my use case can also be served? Right now many times, a team would be ‘Hey I need to get my use case out very soon. So, I don’t really care about documenting well, putting the right description about what the features are’. So, everybody will be a bit lazy in making sure they have done a good enough job so that people can understand.” CS26.

(R7) Unification and discovery through centralization: Practitioners mention centralization as a key concept to unify different tools as well as enable discovery of features. The central management of tools and infrastructure takes the burden from individual teams and gives them more time to focus on their use cases as well as enables easier communication between teams. Feature stores are again mentioned to be key to a central feature management. While it enables not only a clean translation of features for training and production, it further enables reusability and shareability across organizational entities. Besides feature stores, any other concept that involves central management of features and corresponding managing infrastructure can help.

(R8) Showcase meetings: For better understanding of each other’s work and progress, some practitioners propose regular meetings in which a team presents its work, their use case, and their progress and learnings. This fosters understanding of use cases, the development of intersecting interests and infrastructures, and education of new technologies.

B. Management vs. Data Science

“All of the higher-ups were just looking at the data science team as a money suck. All they were doing was walking around. They weren’t putting anything into production. They weren’t doing much: looking at data, asking for more data. And, they weren’t actually producing anything. And so the C-level executives: ‘Hey what are we paying these guys for? What is going on here?’ And they couldn’t figure out the value.” M29

Naturally, the management needs to be informed about the progress and state of development, much like in traditional software development. Tensions between management and development are, among others, one reason for introducing self-organizing teams and agile principles. However, the progress and outcome in ML development is inherently uncertain, which further amplifies communication hurdles.

(AP10) Data scientists struggle in their roles and get burned out: Data scientist struggle in their roles, feeling like they are not delivering enough value and are not producing tangible results for the company. Practitioners link this to feelings of burnout due to never ending iterations that finally do not lead to valuable results. Without documentation, practitioners find it hard to look back on their own work and see their contribution.

(AP11) Tension between management and data science: In the analyzed videos, data scientists as well as management report that communication can be difficult with each other. While management expects the data scientists to deliver value to the company, data scientists follow their research-oriented workflow. This increases tensions as the open research workflow may not be aligned directly with the business value. Usually, metrics do not map directly and the effects on the ML model are not directly visible. The frustration managers report goes even further into the accusation that data scientists neither want to be managed nor want to be confronted with their own value in the company: “When you talk about data scientists: if I use the letters ROI, I get hate mails. And I’ve done that before! I’ve posted: ‘You have to have ROI, you have to be connected to business problems.’ I get hate mails. Data scientists hate it. [...] A lot of data scientists come straight out of academia and it’s a different paradigm.” M30

(C6) Missing data science process: A stated cause for the anti-patterns reported above are missing standard development processes, such as SCRUM that the management knows and can align with its expectations about the development progress. The research-driven approach of model building contains too many uncertainties: “Software is very results driven, where it’s very easy to be like: ‘Hey, I build this piece of code and it should be x+x equals y. […]’ The input and the output should be very clear. Sometimes data science isn’t that way […] where it’s more research-based and kind of long-term projects, there’s a likelihood of failure.” CS21. The combination of inexperienced management that does not fully understand the development process of data science projects and the absence of standard development frameworks for data science lead to unstructured approaches that are often not accompanied by documentation and reporting.
Communication mismatch between technical and non-technical people: The communication mismatch between technical and non-technical people is seen as a major cause for diverse problems not only for data science but also for software development. The inability of technical people explaining their problems and achievements in a way that is understandable and relevant for the business is the root of tensions between the two groups. “I’m like ‘Hey, check out this cool RMSE graph!’ And with almost no labels on it and show it to a marketing team. And, they look at it as if they’re meeting an alien for the first time. They’re like ‘what is this?’.”

Strong processes: Building strong development processes for data science in general is a key recommendation from practitioners. They additionally recommend defined processes on how to intersect and communicate to management and other business-oriented entities. Tailoring the agile framework to the experiment-driven building process of ML models might be a way, but research is missing on this part.

Documentation and reporting: Documentation has been recommended as a main tool to communicate with the management and to enable retrospective analysis of the progress made so far by the data scientists themselves. Documentation is seen as the foundation for writing reports and preparing presentations, in which data scientists should focus more on communicating in terms of business goals than specific technical metrics. This is an interesting insight as this, at least partially, diverges from the low documentation approach of agile practices in software development. “Having that ability to always look back, there’s a lot of value there. And a lot of people just are ‘Oh, I just analyze this data and write up a powerpoint and then be done with it’.”

Leadership vacuum: We identified that many stated problems are due to a leadership vacuum. That is, there is no authority that either acts at all or with sufficient knowledge about data science.

“Everyone kept thinking that the solution would be to hire more data scientists. So, for me it was going back to leadership and the head of engineering and saying: ‘No, please stop! You can add [...] as many as you want, you’re not going to solve the major issue that we have.’ [...] And this title conundrum came up with every single one of them. It was continuously ‘Well, we can just hire a data scientist. Because we need ETL pipelines and we need somebody to make sure that they architect a very good relational database and then push all that reporting out to our multiple users’, and I was like: ‘[...] Is that the right person that you should be hiring?’ And they’re like: ‘It has data in the title’.”

Putting more people in a team, especially late before deadlines, has already been debunked for traditional software engineering. This ineffective management decision becomes even worse when the wrong people are hired, which have not the skills that are actually needed. Uninformed and non-fitting hiring has been reported by multiple practitioners. This situations seems to be more common in start-ups or companies that are just starting to build their first data-science use cases.
Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Developing and productionizing an ML model involves several stages at which decisions are made about which tools and technologies are used. Ideally, decisions are driven by the business goal, the available resources in terms of infrastructure and developer skills, and organization aspects. Résumé-driven-development is a phenomenon that just recently was defined as an interaction between recruiters and software professionals in which knowledge and experience with new and trending technologies is overemphasized. Practitioners in the MLOps community mention this to further influence decision processes within teams.

Identifying a use case that benefits from ML is challenging and should not be driven by hypes to generate profit. The anti-patterns emphasize the importance of proper requirements engineering, including suitable product metrics.

(A15) Stuck with proof of concepts: The organization keeps making proof of concepts (PoC) for algorithms but most of these never make it into an actual product. The speaker in M62 calls this phenomenon “proof-of-concept-hell”, where even successful PoCs never make it into production. The reason is that the product itself is not relevant for the user or not presented in a proper way. Sometimes, data scientists get stuck in endless iterations and optimizations of a single metric (e.g. accuracy). Ultimately, this process is not tied to the actual business value: “You’re really excited about technology. You just want to build something in one direction. It’s like: ‘Okay, cool, how deep can we go? […] How advanced can we make this model?’ It’s really important […] to really tie that back with exactly what you’re saying: You want to understand what’s the ultimate goal.”

(A16) Perception of ‘Everything can benefit from ML!’: Instead of a clear analysis of which aspects of a product can benefit from ML and the exploration of traditional development methods, there is a perception that everything should be done using ML. There is no clear roadmap or strategy of how ML will help the organization achieving a certain goal. This perception overrules fundamental business and project decisions, such as what is the value of an ML-enabled product feature for the customer, or how do we monetize ML.

(A17) Product is not feasible due to missing data or talent: Companies are very clear about what they want to do, but do not have the right data for their goal. Or, they cannot build it because they do not have the required staff. This seems to be a major problem of organizations that try to on-board a data product: “So, people will often think about what types of ML they want to do, but they won’t have the correct type of infrastructure to get the data that they need […] So that’s usually when people are rushing into something and then they realize: ‘I have zero training data’.”

(C13) Missing organizational strategy, governance, and structure: Speakers report that organizations are missing a data strategy when introducing an ML product for the first time. Missing suitable data is a major reason, which further relates to unmanaged data collection processes. They identify this as a management or leadership problem, since this cause manifests already before the development of the ML-enabled software. “For any company, of any size, when they’re trying to create that new data product, the majority of the time, they really haven’t collected the data they need. Because it’s either changed too much, they don’t have enough of it or the process
Managers report that the management lacks knowledge about ML. Practitioners have found that the management has no clear understanding of what ML is and how it works. They further complain that this leads to underestimating the differences between a data-science product and a traditional software product. Often, the engineering effort put into productizing a model and its influence on a return on investment is underestimated. This, in turn, makes the impression that every product can and should have an ML component. Simultaneously, companies seem to experience pressure to integrate ML into their products to stay competitive on the market, which enforces building products without the executive knowledge to manage them. “From 2016 to 2019 maybe, there was a big emphasis with CTOs, CIOs, and CEOs that we had to do machine learning, we had to do data science, because if we weren’t, competitors might be doing it, or I just won’t have anything interesting to talk about with my other CTO friends when we meet for coffee.”

Speakers emphasize how hard it is to find people that have both, the management and the technological skills, to successfully guide and develop machine learning projects. This especially enforces the disconnection between development teams and stakeholders, which is mentioned as a key challenge that most speakers do not have a clear answer to. A result of the poor translation of the business or user problem into an ML question are metrics that do not reflect the actual problem and this ultimately leads to the wrong metrics being optimized.

Missing production metrics: Missing production metrics causes feasible PoCs to be unsuitable in production. If data scientists are not evaluating their models on metrics that are meaningful for the product (and customer), they can not judge whether a PoC is actually successful.

(R13) Process to identify feasible use cases and product roadmap: Recommendations intend to raise awareness that ML will bring a lot of complexity to the product. So, most speakers recommend to rely on more traditional engineering methods and heuristics: “If you can avoid using machine learning, then do it!”

Finding a reliable process to identify and evaluate use cases is crucial before starting to iterate on the problem itself. Therefore, the interviewees suggest multi-step processes that must pass before the product idea is given to the data science team: “So, first you get sales, they go through sales. Then, you have a subject matter expert that looks at it, that knows a bit of data analysis. And then, if it passes through these filters, it goes through that AutoML to make sure that there’s possibility on that end. And then, it can go to the data scientist.”

A further suggestion is to write a proposal that states feasibility estimates of the final product and data availability. It should be evaluated by multiple organizational units to result in a full roadmap that can be given to the data scientists implementing the product.

(R14) Understand customers and keep them close: Practitioners recommend to keep problem owners and other stakeholders close to the data science team and start by understanding the actual requirements.

(R15) Education: The management needs the ability to evaluate and identify use cases for ML. Education through workshops is recommended for emphasizing data availability. If a product is evaluated to lack data, setting up a governed data collection process is required.

VI. DISCUSSION & CONCLUSION

We have found a large range of anti-patterns and causes from different organizational viewpoints. Although our study is not confirmatory by design, we could reproduce several findings from related work and match...
them with ours. The result of this process is shown in Table III where each heading indicates the related paper with its section and context, in which challenges have been reported. Column challenge corresponds to a heading or paragraph within the corresponding section of related work, and the confirmed findings column refers to the exact finding we could reproduce.

Overall, we found in all papers overlap, demonstrating the generality of our results. Amershi and others [27] focus on software engineering teams with ML components at Microsoft. Experts consistently mention challenges around data availability and reusability. The authors synthesize their findings into three major differences between engineering traditional and ML-enabled software: Data discovery and management, Customization and Reuse, and ML Modularity. While the challenges around data clearly correspond to several of our findings (AP4-5), the picture for customization and reuse is more complex. The authors focused solely on ML models’ reusability for which we could not find any anti-pattern. However, we found additional issues of reusability of features and infrastructure, leading to redundant development (AP7-9). Finally, the authors describe challenges for close entangling between model teams. We also find anti-patterns related to organizational silos between model and product team as well as between data producer and consumer (AP1-3, AP4-6).

Kim and others [25] investigate the role, work, and background of data scientists also at Microsoft. Their findings motivate further research in different directions, one being centralized data and standardized nomenclature, a recommendation that was also made by our speakers (R6-7). Apte et al. and others [26] investigate challenges of deep learning within seven projects and find organizational challenges to be one of three main categories. Challenges include tensions between product management and data scientists and different values between engineers and data scientists, an observation that was frequently shared by the MLOps community (AP11, C1, C6).

Consistent with Nahar and others [11], we see the clash of cultures (C1) and a lack of engineering competence in data scientists (C2) in combination with organizational siloing as root causes for tensions between teams. We share several observations for anti-patterns (AP2, AP4-6) as they also studied socio-technical aspects. In addition, we contextualized some of their findings not as anti-patterns, but as causes to challenges (e.g., C14, C16), which provides a new perspective on which practitioners can act on.

For instance, the authors report back that hiring people with mixed skills in engineering and data science seems beneficial, but often absent. We were able to complement this observation by describing anti-patterns and tracing their causes back to unclear roles and titles (C8) and a general skill shortage on the market (C10), which amplifies the struggles of uneducated hirers (C9) to recruit the right staff.

Compared to other papers, our sample holds unique characteristics (i.e., mainly leaders and managers with business and technical expertise). This way, we can abstract from single teams to the whole organization and make novel findings related to lack of communication and leadership. Specifically, these findings relate to missing decision processes within teams (C12) that finally lead to unmaintainable projects (AP14), and a lack of product management (C14-16) that enforces endless iterations on proof-of-concepts with meaningless metrics (AP15).

As a notable methodological contribution, we have shown how practitioner communities can be leveraged to provide a valuable source of qualitative data next to time-consuming (semi-)structured interviews, especially in areas where practitioners are hard to recruit.

The presented anti-patterns can be starting points for future research endeavours. In this regard, a notable open question is whether the found anti-patterns are specific to the development of ML-enabled systems or are, in fact, artifacts of more general socio-technical challenges in software engineering, which at least partly, have been studied before [28], [29].

Finally, this paper is not exclusively targeting academics, but enables transfer to industry by providing clear actionables for practitioners. We have gathered a list of 17 anti-patterns that managers and organizational leaders can use to spot socio-technical issues. The identified causes and suggested recommendations help practitioners to act on these anti-patterns. Overall, our findings call for more research on development processes, guidelines, and general organization structures to support the development of ML-enabled software.

**ACKNOWLEDGMENT**

The authors work has been supported by the Federal Ministry of Education and Research of Germany and by the Sächsische Staatsministerium für Wissenschaft Kultur und Tourismus in the program Center of Excellence for AI-research “Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig”, project identification number: ScAD.AI. Siegmund’s work has been funded by the German Research Foundation (SI 2171/2-2).

**REFERENCES**


[38] MLOps.community “Path to Production and Monetizing Machine Learning”, YouTube, Aug 18, 2020. https://www.youtube.com/watch?v=SNk0B0j8SrQ


[53] MLOps.community “MLOps 1 Year Anniversary!”, YouTube, Apr 9, 2021. https://www.youtube.com/watch?v=Wmnj1WYXUE