Received 20 February 2023, accepted 16 March 2023, date of publication 27 March 2023, date of current version 3 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3262138

RESEARCH ARTICLE

Machine Learning Operations (MLOps): Overview, Definition, and Architecture

DOMINIK KREUZBERGER[1], NIKLAS KÜHL[1,2], AND SEBASTIAN HIRSCHL[1]

[1]IBM, 71139 Ehningen, Germany

[2]Information Systems and Human-Centric Artificial Intelligence, University of Bayreuth, 95447 Bayreuth, Germany

Corresponding author: Niklas Kühl (kuehl@uni-bayreuth.de)

This work was supported in part by Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Grant 491183248, and in part by the Open Access Publishing Fund of the University of Bayreuth.

ABSTRACT The final goal of all industrial machine learning (ML) projects is to develop ML products and rapidly bring them into production. However, it is highly challenging to automate and operationalize ML products and thus many ML endeavors fail to deliver on their expectations. The paradigm of Machine Learning Operations (MLOps) addresses this issue. MLOps includes several aspects, such as best practices, sets of concepts, and development culture. However, MLOps is still a vague term and its consequences for researchers and professionals are ambiguous. To address this gap, we conduct mixed-method research, including a literature review, a tool review, and expert interviews. As a result of these investigations, we contribute to the body of knowledge by providing an aggregated overview of the necessary principles, components, and roles, as well as the associated architecture and workflows. Furthermore, we provide a comprehensive definition of MLOps and highlight open challenges in the field. Finally, this work provides guidance for ML researchers and practitioners who want to automate and operate their ML products with a designated set of technologies.

INDEX TERMS CI/CD, DevOps, machine learning, MLOps, operations, workflow orchestration.

I. INTRODUCTION

Machine Learning (ML) has become an important technique to leverage the potential of data and allows businesses to be more innovative [1], efficient [2], and sustainable [3]. However, the success of many productive ML applications in real-world settings falls short of expectations [4]. A large number of ML projects fail—with many ML proofs of concept never progressing as far as production [5]. From a research perspective, this does not come as a surprise as the ML community has focused extensively on the building of ML models, but not on (a) building production-ready ML products and (b) providing the necessary coordination of the resulting, often complex ML system components and infrastructure, including the roles required to automate and operate an ML system in a real-world setting [6]. For instance, in many industrial applications, data scientists still manage ML workflows manually to a great extent, resulting in many issues during the operations of the respective ML solution [7].

To address these issues, the goal of this work is to examine how manual ML processes can be automated and operationalized so that more ML proofs of concept can be brought into production. In this work, we explore the emerging ML engineering practice "Machine Learning Operations"—MLOps for short—precisely addressing the issue of designing and maintaining productive ML. We take a holistic perspective to gain a common understanding of the involved components, principles, roles, and architectures. While existing research sheds some light on various specific aspects of MLOps, a holistic conceptualization, generalization, and clarification of ML systems design are still missing. Different perspectives and conceptions of the term "MLOps" might lead to misunderstandings and miscommunication, which, in turn, can lead to errors in the overall setup of the entire ML system. Thus, we ask the research question:

RQ: What is MLOps?

The associate editor coordinating the review of this manuscript and approving it for publication was Alberto Cano.

To answer that question, we conduct a mixed-method research endeavor to (a) identify important principles of MLOps, (b) carve out functional core components, (c) highlight the roles necessary to successfully implement MLOps, and (d) derive a general architecture for ML systems design. In combination, these insights result in a definition of MLOps, which contributes to a common understanding of the term and related concepts.

Therefore, we hope to positively impact academic and practical discussions by providing clear guidelines for professionals and researchers alike with precise responsibilities. These insights can assist in allowing more proofs of concept to make it into production by having fewer errors in the system's design and, finally, enabling more robust predictions in real-world environments.

The remainder of this article is structured as follows. We will first elaborate on the necessary foundations and related work in the field. Next, we will give an overview of the utilized methodology, consisting of a literature review, a tool review, and an interview study. We then present the insights derived from the application of the methodology and conceptualize these by providing a unifying definition. We conclude the paper with a short summary, limitations, and outlook.

II. FOUNDATIONS OF DEVOPS

In the past, different software process models and development methodologies surfaced in the field of software engineering. Prominent examples include waterfall [8] and the agile manifesto [9]. Those methodologies have similar aims, namely to deliver production-ready software products. A concept called "DevOps" emerged in the years 2008/2009 and aims to reduce issues in software development [10], [11]. DevOps is more than a pure methodology and rather represents a paradigm addressing social and technical issues in organizations engaged in software development. It has the goal of eliminating the gap between development and operations and emphasizes collaboration, communication, and knowledge sharing. DevOps promotes automation through the tactic of continuous integration, continuous delivery, and continuous deployment (CI/CD), enabling fast, frequent, and reliable releases. Moreover, it is designed to ensure continuous testing, quality assurance, continuous monitoring, logging, and feedback loops. Due to the commercialization of DevOps, many DevOps tools are emerging, which can be differentiated into six groups [12], [13]: collaboration and knowledge sharing (e.g., Slack, Trello, GitLab wiki), source code management (e.g., GitHub, GitLab), build process (e.g., Maven), continuous integration (e.g., Jenkins, GitLab CI), deployment automation (e.g., Kubernetes, Docker), monitoring and logging (e.g., Prometheus, Logstash). Cloud environments are increasingly equipped with ready-to-use DevOps tooling that is designed for cloud use, facilitating the efficient generation of value [14]. With this novel shift towards DevOps, developers need to care about what they develop, as they need to operate it as well. As empirical results demonstrate, DevOps ensures better software quality [15]. People in the industry, as well as academics, have gained a wealth of experience in software engineering using DevOps. This experience is now being used to automate and operationalize ML.

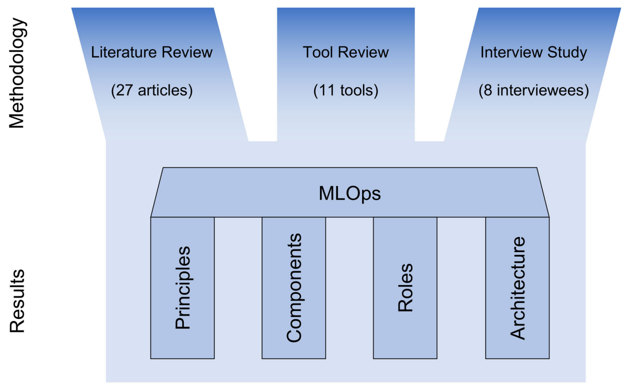

FIGURE 1. Overview of the methodology.

III. METHODOLOGY

To derive insights from the academic knowledge base while also drawing upon the expertise of practitioners from the field, we apply a mixed-method approach, as depicted in Figure 1. As a first step, we conduct a structured literature review [16], [17] to obtain an overview of relevant research. Furthermore, we review relevant tooling support in the field of MLOps to gain a better understanding of the technical components involved. Finally, we conduct semi-structured interviews [18], [19] with experts from different domains. On that basis, we conceptualize the term "MLOps" and elaborate on our findings by synthesizing literature and interviews in the next chapter ("Results").

A. LITERATURE REVIEW

To ensure that our results are based on scientific knowledge, we conduct a systematic literature review according to the method of Webster and Watson [16] and Kitchenham et al. [17]. After an initial exploratory search, we define our search query as follows: *((("DevOps" OR "CICD" OR "Continuous Integration" OR "Continuous Delivery" OR "Continuous Deployment") AND "Machine Learning") OR "MLOps" OR "CD4ML")*.

We query the scientific databases of Google Scholar, Web of Science, Science Direct, Scopus, and the Association for Information Systems eLibrary. It should be mentioned that the use of DevOps for ML, MLOps, and continuous practices in combination with ML is a relatively new field in academic literature. Thus, only a few peer-reviewed studies are available at the time of this research. Nevertheless, to gain experience in this area, the search included non-peer-reviewed literature as well. The search was performed in May 2021 and resulted in 1,864 retrieved articles. Of those, we screened 194 papers in detail. From that group,

TABLE 1. List of evaluated technologies.

	Technology Name	**Description**	**Sources**
Open-source examples	TensorFlow Extended	TensorFlow Extended (TFX) is a configuration framework providing libraries for each of the tasks of an end-to-end ML pipeline. Examples are data validation, data distribution checks, model training, and model serving.	[7], [23], [28], [δ, θ]
	Airflow	Airflow is a task and workflow orchestration tool, which can also be used for ML workflow orchestration. It is also used for orchestrating data engineering jobs. Tasks are executed according to directed acyclic graphs (DAGs).	[7], [26], [27], [α, β, ζ, η]
	Kubeflow	Kubeflow is a Kubernetes-based end-to-end ML platform. Each Kubeflow component is wrapped into a container and orchestrated by Kubernetes. Also, each task of an ML workflow pipeline is handled with one container.	[6], [7], [23], [26], [27], [α, β, γ, δ, ζ, η, θ]
	MLflow	MLflow is an ML platform that allows for the management of the ML lifecycle end-to-end. It provides an advanced experiment tracking functionality, a model registry, and model serving component.	[6], [31], [39], [α, γ, ε, ζ, η, θ]
Commercial examples	Databricks managed MLflow	The Databricks platform offers managed services based on other cloud providers' infrastructure, e.g., managed MLflow.	[6], [7], [27], [31], [α, ζ]
	Amazon CodePipeline	Amazon CodePipeline is a CI/CD automation tool to facilitate the build, test, and delivery steps. It also allows one to schedule and manage the different stages of an ML pipeline.	[66] [γ]
	Amazon SageMaker	With SageMaker, Amazon AWS offers an end-to-end ML platform. It provides, out-of-the-box, a feature store, orchestration with SageMaker Pipelines, and model serving with SageMaker endpoints.	[6], [33], [39], [66], [α, β, γ, ζ, θ]
	Azure DevOps Pipelines	Azure DevOps Pipelines is a CI/CD automation tool to facilitate the build, test, and delivery steps. It also allows one to schedule and manage the different stages of an ML pipeline.	[36], [66], [γ, ε]
	Azure ML	Microsoft Azure offers, in combination with Azure DevOps Pipelines and Azure ML, an end-to-end ML platform.	[6], [35]–[37], [45], [α, γ, ε, ζ, η, θ]
	GCP - Vertex AI	GCP offers, along with Vertex AI, a fully managed end-to-end platform. In addition, they offer a managed Kubernetes cluster with Kubeflow as a service.	[6], [26], [27], [37], [α, γ, δ, ζ, θ]
	IBM Cloud Pak for Data (IBM Watson Studio)	IBM Cloud Pak for Data combines a list of software in a package that offers data and ML capabilities.	[26], [γ]

27 articles were selected based on our inclusion and exclusion criteria (e.g., the term MLOps or DevOps and CI/CD in combination with ML was described in detail, the article was written in English, etc.). All 27 of these articles were peer-reviewed.

B. TOOL REVIEW

After going through 27 articles and eight interviews, various open-source tools, frameworks, and commercial cloud ML services were identified. These tools, frameworks, and ML services were reviewed to gain an understanding of the

TABLE 2. List of interview partners.

Interviewee pseudonym	Job Title	Years of experience with DevOps	Years of experience with ML	Industry	Company Size (number of employees)
Alpha (α)	Senior Data Platform Engineer	3	4	Sporting Goods / Retail	60,000
Beta (β)	Solution architect / Specialist for ML and AI	6	10	IT Services / Cloud Provider / Cloud Computing	25,000
Gamma (γ)	AI Architect / Consultant	5	7	Cloud Provider	350,000
Delta (δ)	Technical Marketing & Manager in ML / AI	10	5	Cloud Provider	139,995
Epsilon (ε)	Technical Architect - Data & AI	1	2	Cloud Provider	160,000
Zeta (ζ)	ML engineering Consultant	5	6	Consulting Company	569,000
Eta (η)	Engineering Manager in AI / Senior Deep Learning Engineer	10	10	Conglomerate (multi-industry)	400,000
Theta (θ)	ML Platform Product Lead	8	10	Music / audio streaming	6,500

technical components of which they consist. An overview of the identified tools is depicted in Table 1.

C. INTERVIEW STUDY

To answer the research questions with insights from practice, we conduct semi-structured expert interviews according to Myers and Newman [18]. One major aspect in the research design of expert interviews is choosing an appropriate sample size [20]. We apply a theoretical sampling approach [21], which allows us to choose experienced interview partners to obtain high-quality data. Such data can provide meaningful insights with a limited number of interviews. To get an adequate sample group and reliable insights, we use LinkedIn—a social network for professionals—to identify experienced ML professionals with profound MLOps knowledge on a global level. To gain insights from various perspectives, we choose interview partners from different organizations and industries, different countries, and nationalities, as well as different genders. Interviews are conducted until no new categories and concepts emerge in the analysis of the data. According to Glaser and Strauss [21], this stage is called "theoretical saturation." In total, we conduct eight interviews with experts (pseudonymized with α - θ), whose details are depicted in Table 2. All interviews are conducted between June and August 2021.

With regard to the interview design, we prepare a semi-structured guide with several questions, documented as an interview script [18]. During the interviews, "soft laddering" is used with "how" and "why" questions to probe the interviewees' means-end chain [19]. This methodical approach allowed us to gain additional insight into the experiences of the interviewees when required. All interviews are recorded and then transcribed. To evaluate the interview transcripts, we use an open coding scheme [20]. The open coding process allows the data to be broken down in an analytical manner so that conceptually similar topics can be grouped into categories and subcategories. These categories are called "codes". Concepts were identified when they appeared multiple times in different interviews [21].

IV. RESULTS

We apply the described methodology and structure our resulting insights into a presentation of important principles, their resulting instantiation as components, the description of necessary roles, as well as a suggestion for the architecture and workflow resulting from the combination of these aspects. Finally, we derive the conceptualization of the term and provide a definition of MLOps.

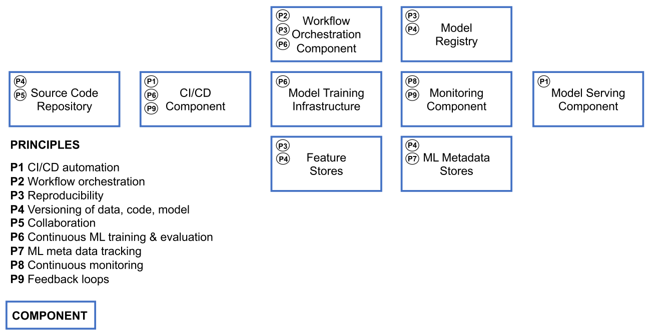
P2 P3 P6 Workflow Orchestration Component
P3 P4 Model Registry
P4 P5 Source Code Repository
P1 P6 P9 CI/CD Component
P6 Model Training Infrastructure
P8 P9 Monitoring Component
P1 Model Serving Component
P3 P4 Feature Stores
P4 P7 ML Metadata Stores
PRINCIPLES
P1 CI/CD automation
P2 Workflow orchestration
P3 Reproducibility
P4 Versioning of data, code, model
P5 Collaboration
P6 Continuous ML training & evaluation
P7 ML meta data tracking
P8 Continuous monitoring
P9 Feedback loops
COMPONENT

FIGURE 2. Implementation of principles within technical components.

A. PRINCIPLES

A principle is viewed as a general or basic truth, a value, or a guide for behavior. In the context of MLOps, a principle is a guide to how things should be realized in MLOps and is closely related to the term ''best practices'' from the professional sector. Based on the outlined methodology, we identified nine principles required to realize MLOps. Figure 2 provides an illustration of these principles and links them to the components with which they are associated.

P1 CI/CD automation. CI/CD automation provides continuous integration, continuous delivery, and continuous deployment. It carries out the build, test, delivery, and deploy steps. It provides fast feedback to developers regarding the success or failure of certain steps, thus increasing the overall productivity. CI/CD puts ideas of DevOps into practice. Therefore, CI/CD can be seen as a DevOps tactic [6], [7], [22], [23], [α, β, θ].

P2 Workflow orchestration. Workflow orchestration coordinates the tasks of an ML workflow pipeline according to directed acyclic graphs (DAGs). DAGs define the task execution order by considering relationships and dependencies [7], [24], [25], [26] [α, β, γ, δ, ζ, η].

P3 Reproducibility. Reproducibility is the ability to reproduce an ML experiment and obtain the exact same results [23], [27] [α, β, δ, ε, η].

P4 Versioning. Versioning ensures the versioning of data, model, and code to enable not only reproducibility, but also traceability (for compliance and auditing reasons) [23], [27] [α, β, δ, ε, η].

P5 Collaboration. Collaboration ensures the possibility to work collaboratively on data, model, and code. Besides the technical aspect, this principle emphasizes a collaborative and communicative work culture aiming to reduce domain silos between different roles [7], [25], [27], [α, δ, θ].

P6 Continuous ML training & evaluation. Continuous training (CT) means periodic retraining of the ML model based on new feature data. Continuous training is enabled through the support of a monitoring component, a feedback loop, and an automated ML workflow pipeline. Continuous training always includes an evaluation run to assess the change in model quality [23], [24], [28], [29], [β, δ, η, θ]. In general, to manage costs of retraining, it should be carefully considered, which update frequency is necessary for the use case (e.g., daily vs. weekly). A powerful tool to decrease cost of retraining is the use of online learning in large scale web applications, benefiting from iterative training steps compared to a full training data set. This way the model can also reflect recent impactful events like catastrophes. There is a magnitude of online learning optimization algorithms available [30].

P7 ML metadata tracking/logging. Metadata is tracked and logged for each orchestrated ML workflow task. Metadata tracking and logging is required for each training job iteration (e.g., training date and time, duration, etc.), including the model specific metadata—e.g., used parameters and the resulting performance metrics, model lineage: data and code used—to ensure the full traceability of experiment runs [6], [7], [31], [32], [α, β, δ, ε, ζ, η, θ].

P8 Continuous monitoring. Continuous monitoring implies the periodic assessment of data, model, code, infrastructure resources, and model serving performance (e.g., prediction accuracy) to detect potential errors or changes that influence the product quality [7], [23], [28], [32], [33], [α, β, γ, δ, ε, ζ, η].

P9 Feedback loops. Multiple feedback loops are required to integrate insights from the quality assessment step into the development or engineering process (e.g., a feedback loop from the experimental model engineering stage to the previous feature engineering stage). Another feedback loop is required from the monitoring component (e.g., observing the model serving performance) to the scheduler to enable the retraining [7], [23], [24], [34], [35], [α, β, δ, ζ, η, θ].

B. TECHNICAL COMPONENTS

After identifying the principles that need to be incorporated into MLOps, we now elaborate on the precise components and implement them in the ML systems design. In the following, the components are listed and described in a generic way with their essential functionalities. The references in brackets refer to the respective principles that the technical components are implementing.

C1 CI/CD Component (P1, P6, P9). The CI/CD component ensures continuous integration, continuous delivery, and continuous deployment. It takes care of the build, test, delivery, and deploy steps. It provides rapid feedback to developers regarding the success or failure of certain steps, thus increasing the overall productivity [6], [7], [22], [23], [24], [28], [α, β, γ, ε, ζ, η]. Examples are Jenkins [7], [24], and GitHub actions (η). For implementing an MLOps workflow, this means the automated linting, assembly and registry of training inference and application code into a shippable format (e.g., Python Wheel) as well as execution of unit and integration test cases. This automation should be an idempotent process using dynamically assigned resources from the CI/CD tool, that results in a binary or archive-based package.

C2 Source Code Repository (P4, P5). The training, inference and application source code is versioned in a repository. It allows multiple developers to commit and merge their code [23], [24], [36], [37], [38] [α, β, γ, ζ, θ]. Examples include Bitbucket [39], [ζ], GitLab [24], [39], [ζ], GitHub [37], [ζ, η], and Gitea [23].

C3 Workflow Orchestration Component (P2, P3, P6). The workflow orchestration component offers task orchestration of an ML workflow via directed acyclic graphs (DAGs). These graphs represent execution order and artifact usage of single steps of the workflow. A workflow uses for example packaged code artifacts in the respective process step, like extracting data, training, inference or embedding of a model binary into an application [6], [7], [23], [26], [31], [α, β, γ, δ, ε, ζ, η]. Examples include Apache Airflow [α, ζ], Kubeflow Pipelines [ζ], Watson Studio Pipelines [γ], Luigi [ζ], AWS SageMaker Pipelines [β], and Azure Pipelines [ε]. In theory, CI/CD tools could also be used to schedule the triggering of specific tasks sequentially, however the complexity of data engineering- or ML-pipeline tasks has increased the need for a tool specifically designed for the purpose of workflow or task orchestration. These workflow orchestration tools make it easier to efficiently manage interrelated and interdependent tasks, because they are specifically designed to manage complex task chains [40].

C4 Feature Store System (P3, P4). A feature store system ensures central storage of commonly used features. It has two databases configured: One database as an offline feature store to serve features with normal latency for experimentation, and one database as an online store to serve features with low latency for predictions in production [25], [28], [α, β, ζ, ε, θ]. Examples include Google Feast [ζ], Amazon AWS Feature Store [β, ζ], Tecton.ai and Hopswork.ai [ζ]. This is where most of the data for training ML models will come from. Moreover, data can also come directly from any kind of data store. A feature store poses complex requirements, which are highly dependent on the use case. Databases of it can be hosted on on-premises infrastructure or in the cloud. However, scalability is typically realized with cloud infrastructure. Most use cases have a read-heavy workload, combined with a batch or streaming-based ingestion pattern on (very) large data sets. Such high scalability can be achieved on distributed file systems [41], [42], or distributed databases [43], [44] combined with parallel and distributed data processing algorithms (e.g., MapReduce or a more high-level API like Spark).

C5 Model Training Infrastructure (P6). The model training infrastructure provides the foundational computation resources, e.g., CPUs, RAM, and GPUs. The provided infrastructure can be either distributed or non-distributed. In general, a scalable and distributed infrastructure is recommended [7], [23], [27], [28], [32], [33], [37], [45], [46], [δ, ζ, η, θ]. Examples include local machines (not scalable) or cloud computation [33] [η, θ], as well as non-distributed or distributed computation (several worker nodes) [7], [37]. Frameworks supporting computation are Kubernetes [η, θ] and Red Hat OpenShift [γ]. Typically, deep learning workloads (training and inference) are matrix-multiplication-heavy and therefore computation bound. GPUs are optimized towards this type of workload and should be the primary focus for compute node specification. In edge devices, where storage and computation power are limited, Quantized Neural Nets with low-precision floating-point operations [47] in combination with pruning and Hofmann Coding should be investigated [48].

C6 Model Registry (P3, P4). The model registry stores centrally the trained ML models together with their metadata. It has two main functionalities: storing the ML artifact and storing the ML metadata (see C7) [7], [24], [25], [34], [35], [α, β, γ, ε, ζ, η, θ]. Advanced storage examples include MLflow [α, η, ζ], AWS SageMaker Model Registry [ζ], Microsoft Azure ML Model Registry [ζ], and Neptune.ai [α]. Simple storage examples include Microsoft Azure Storage, Google Cloud Storage, and Amazon AWS S3 [23].

C7 ML Metadata Stores (P4, P7). ML metadata stores allow for the tracking of various kinds of metadata, e.g., for each orchestrated ML workflow pipeline task. Another metadata store can be configured within the model registry for tracking and logging the metadata of each training job (e.g., training date and time, duration, etc.), including the model specific metadata—e.g., used parameters and the resulting performance metrics, model lineage: data and code used [7], [25], [31], [37], [49], [α, β, δ, ζ, θ]. Examples include orchestrators with built-in metadata stores tracking each step of experiment pipelines [α] such as Kubeflow Pipelines [α, ζ], AWS SageMaker Pipelines [α, ζ], Azure ML, and IBM Watson Studio [γ]. MLflow provides an advanced metadata store in combination with the model registry [6], [31].

C8 Model Serving Component (P1). The model serving component can be configured for different purposes. Examples are online inference for real-time predictions or batch inference for predictions using large volumes of input data. The serving can be provided, e.g., via a REST API. As a foundational infrastructure layer, a scalable and distributed model serving infrastructure is recommended [23], [27], [33], [37], [39], [46], [α, β, δ, ζ, η, θ]. One example of a model serving component configuration is the use of Kubernetes and Docker technology to containerize the ML model, and leveraging a Python web application framework like Flask [24] with an API for serving [α]. Other Kubernetes supported frameworks are KServing of Kubeflow [α], TensorFlow Serving, and Seldion.io serving [27]. Inferencing could also be realized with Apache Spark for batch predictions [θ]. Examples of cloud services include Microsoft Azure ML REST API [ε], AWS SageMaker Endpoints [α, β], IBM Watson Studio [γ], and Google Vertex AI prediction service [δ]. The actual deployment of the model depends on the use case and falls typically into one of these categories: real-time, batch, or serverless inference. Real-time inference can be achieved by hosting the model in a RESTful web-service, batch inference can be an idempotent MapReduce workflow, and serverless inference is used when cost-efficient and scalable serving is required.

C9 Monitoring Component (P8, P9). The monitoring component takes care of the continuous monitoring of the model serving performance (e.g., prediction accuracy). Additionally, monitoring of the ML infrastructure, CI/CD, and orchestration are required [7], [23], [24], [28], [32], [33], [50], [α, ζ, η, θ]. Examples include Prometheus with Grafana [η, ζ], ELK stack (Elasticsearch, Logstash, and Kibana) [α, η, ζ], and simply TensorBoard [θ]. Examples with built-in monitoring capabilities are Kubeflow [θ], MLflow [η], and AWS SageMaker model monitor or cloud watch [ζ].

C. ROLES

After describing the principles and their resulting instantiation of components, we identify necessary roles in order to realize MLOps in the following. MLOps is an interdisciplinary group process, and the interplay of different roles is crucial to design, manage, automate, and operate an ML system in production. In the following, every role, its purpose, and related tasks are briefly described:

R1 Business Stakeholder (similar roles: Product Owner, Project Manager). The business stakeholder defines the business goal to be achieved with ML and takes care of the communication side of the business, e.g., presenting the return on investment (ROI) generated with an ML product [7], [24], [45] [α, β, δ, θ].

R2 Solution Architect (similar role: IT Architect). The solution architect designs the architecture and defines the technologies to be used, following a thorough evaluation [7], [24], [α, ζ].

R3 Data Scientist (similar roles: ML Specialist, ML Developer). The data scientist translates the business problem into an ML problem and takes care of the model engineering, including the selection of the best-performing algorithm and hyperparameters [7], [25], [32], [33], [α, β, γ, δ, ε, ζ, η, θ].

R4 Data Engineer (similar role: DataOps Engineer). The data engineer builds up and manages data and feature engineering pipelines. Moreover, this role ensures proper data ingestion to the databases of the feature store system [25], [26], [32], [α, β, γ, δ, ε, ζ, η, θ].

R5 Software Engineer. The software engineer applies software design patterns, widely accepted coding guidelines, and best practices to turn the raw ML problem into a well-engineered product [32], [α, γ].

R6 DevOps Engineer. The DevOps engineer bridges the gap between development and operations and ensures proper CI/CD automation, ML workflow orchestration, model deployment to production, and monitoring [7], [22], [25], [51], [α, β, γ, ε, ζ, η, θ].

R7 ML Engineer/MLOps Engineer. The ML engineer or MLOps engineer combines aspects of several roles and thus has cross-domain knowledge. This role incorporates skills from data scientists, data engineers, software engineers, DevOps engineers, and backend engineers (see Figure 3). This cross-domain role builds up and operates the ML infrastructure, manages the automated ML workflow pipelines and model deployment to production, and monitors both the model and the ML infrastructure [7], [24], [25], [32], [α, β, γ, δ, ε, ζ, η, θ].

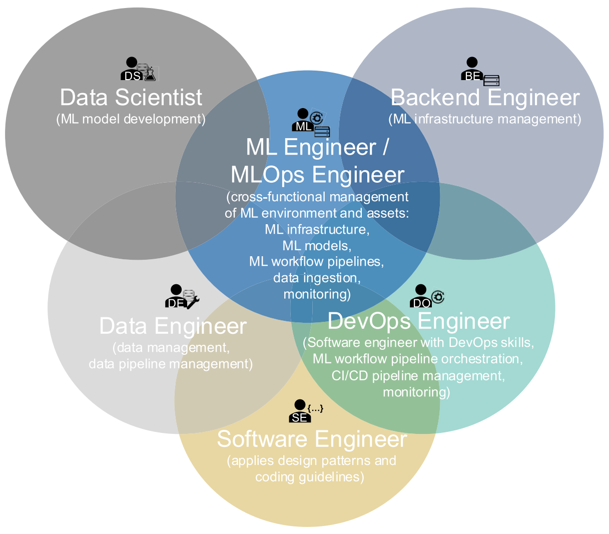
DS
Data Scientist
(ML model development)
BE
Backend Engineer
(ML infrastructure management)
ML
ML Engineer / MLOps Engineer
(cross-functional management of ML environment and assets: ML infrastructure, ML models, ML workflow pipelines, data ingestion, monitoring)
DE
Data Engineer
(data management, data pipeline management)
DO
DevOps Engineer
(Software engineer with DevOps skills, ML workflow pipeline orchestration, CI/CD pipeline management, monitoring)
SE
Software Engineer
(applies design patterns and coding guidelines)

FIGURE 3. Roles and their intersections contributing to the MLOps paradigm.

V. ARCHITECTURE AND WORKFLOW

On the basis of the identified principles, components, and roles, we derive a generalized MLOps end-to-end architecture to give ML researchers and practitioners proper guidance. It is depicted in Figure 4. Additionally, we depict the workflows, i.e., the sequence in which the different tasks are executed in the different stages. The artifact was designed to be technology-agnostic. Therefore, ML researchers and practitioners can choose the best-fitting technologies and frameworks for their needs. This means the MLOps process and components can either be built out of ''best-of-breed'' open-source tools, but also with enterprise solutions. Also, a mix and match combination of enterprise and open-source tools to realize MLOps is possible. Enterprise softwares / cloud services often allow the connection to open-source tools via their APIs and vice versa. Thus, it should be considered to have a look at newest developments as the open source tool market is growing rapidly. There are frequently new options for combinations possible. However, there are certainly also some constraints when it comes to API interfaces connections and combinations. In general, it is hard to say which technologies are good to combine and which aren't. However, with the newly introduced examples of applications and the precise mentioning of tools, we demonstrate possible combinations.

As depicted in Figure 4, we illustrate an end-to-end process, from MLOps product initiation to the model serving. It includes (A) the MLOps product initiation steps; (B) the feature engineering pipeline, including the data ingestion to the feature store; (C) the experimentation; and (D) the automated ML workflow pipeline up to the model serving.

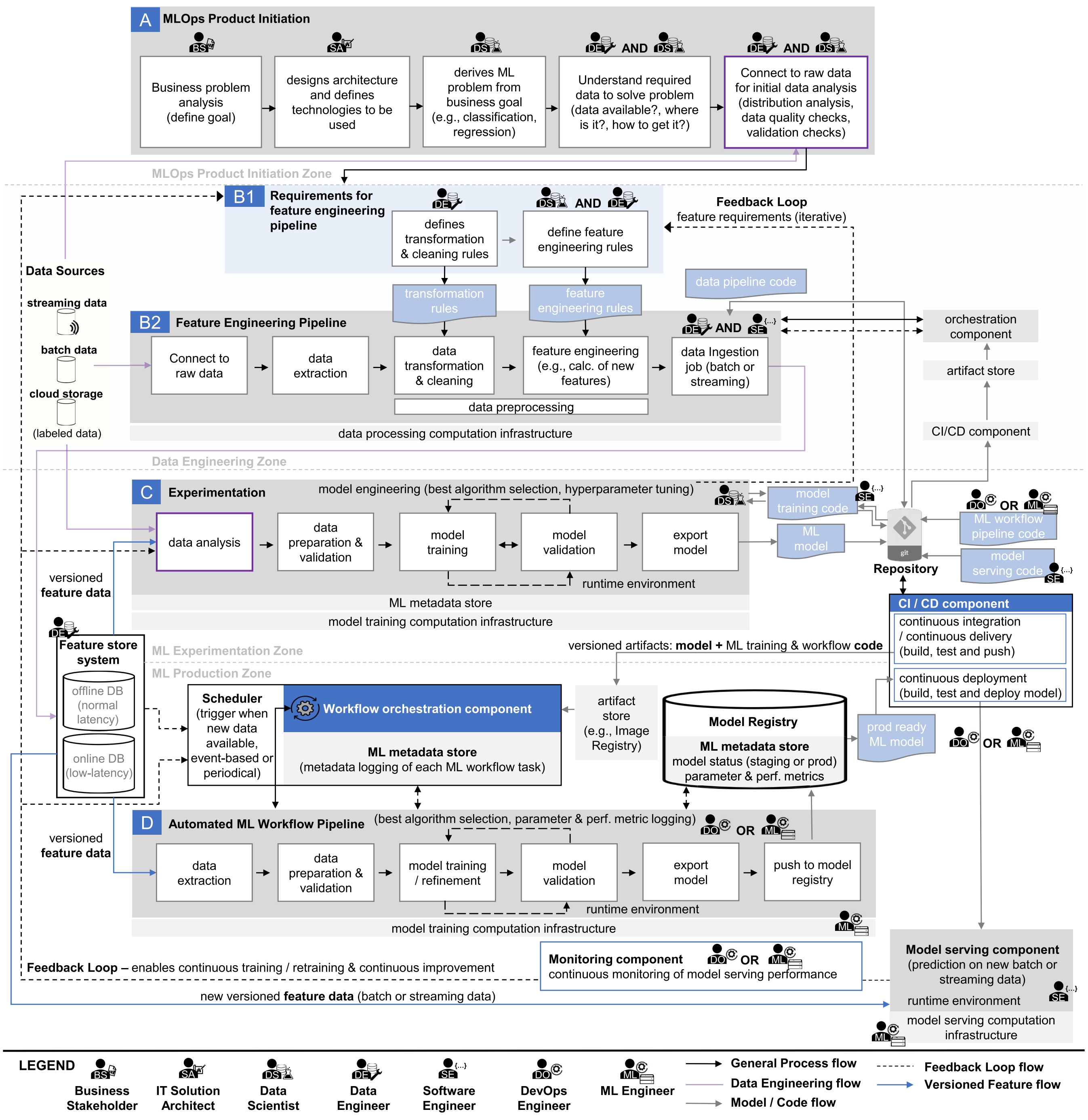
A MLOps Product Initiation
Business problem analysis (define goal)
designs architecture and defines technologies to be used
derives ML problem from business goal (e.g., classification, regression)
Understand required data to solve problem (data available?, where is it?, how to get it?)
Connect to raw data for initial data analysis (distribution analysis, data quality checks, validation checks)
MLOps Product Initiation Zone
B1 Requirements for feature engineering pipeline
defines transformation & cleaning rules
define feature engineering rules
Feedback Loop feature requirements (iterative)
Data Sources
streaming data
batch data
cloud storage
(labeled data)
transformation rules
feature engineering rules
data pipeline code
B2 Feature Engineering Pipeline
Connect to raw data
data extraction
data transformation & cleaning
feature engineering (e.g., calc. of new features)
data Ingestion job (batch or streaming)
data preprocessing
data processing computation infrastructure
orchestration component
artifact store
CI/CD component
Data Engineering Zone
C Experimentation
model engineering (best algorithm selection, hyperparameter tuning)
data analysis
data preparation & validation
model training
model validation
export model
runtime environment
ML metadata store
model training computation infrastructure
model training code
ML model
ML workflow pipeline code
model serving code
Repository
versioned feature data
Feature store system
offline DB (normal latency)
online DB (low-latency)
CI / CD component
continuous integration / continuous delivery (build, test and push)
continuous deployment (build, test and deploy model)
ML Experimentation Zone
ML Production Zone
versioned artifacts: model + ML training & workflow code
Scheduler (trigger when new data available, event-based or periodical)
Workflow orchestration component
ML metadata store (metadata logging of each ML workflow task)
artifact store (e.g., Image Registry)
Model Registry
ML metadata store model status (staging or prod) parameter & perf. metrics
prod ready ML model
D Automated ML Workflow Pipeline
(best algorithm selection, parameter & perf. metric logging)
data extraction
data preparation & validation
model training / refinement
model validation
export model
push to model registry
runtime environment
model training computation infrastructure
Model serving component (prediction on new batch or streaming data)
runtime environment
model serving computation infrastructure
Monitoring component continuous monitoring of model serving performance
Feedback Loop – enables continuous training / retraining & continuous improvement
new versioned feature data (batch or streaming data)
AND
OR
LEGEND
Business Stakeholder
IT Solution Architect
Data Scientist
Data Engineer
Software Engineer
DevOps Engineer
ML Engineer
General Process flow
Data Engineering flow
Model / Code flow
Feedback Loop flow
Versioned Feature flow

FIGURE 4. End-to-end MLOps architecture and workflow with functional components and roles.

(A) MLOps product initiation. (1) The business stakeholder (R1) analyzes the business and identifies a potential business problem that can be solved using ML. (2) The solution architect (R2) defines the architecture design for the overall ML system, and decides on the technologies to be used after a thorough evaluation. (3) The data scientist (R3) derives an ML problem—such as whether regression or classification should be used—from the business goal. (4) The data engineer (R4) and the data scientist (R3) work together to understand which data is required to solve the problem. (5) Once the answers are clarified, the data engineer (R4) and data scientist (R3) collaborate to locate the raw data sources for the initial data analysis. They check the distribution, and quality of the data, as well as performing validation checks. Furthermore, they ensure that the incoming data from the data sources is labeled, meaning that a target attribute is known, as this is a mandatory requirement for supervised ML. In this example, the data sources already had labeled data available as the labeling step was covered during an upstream process.

(B1) Requirements for feature engineering pipeline. The features are the relevant attributes required for model training. After the initial understanding of the raw data and the initial data analysis, the fundamental requirements for the feature engineering pipeline are defined, as follows: (6) The data engineer (R4) defines the data transformation rules (normalization, aggregations) and cleaning rules to bring the data into a usable format. (7) The data scientist (R3) and data engineer (R4) together define the feature engineering rules, such as the calculation of new and more advanced features based on other features. These initially defined rules must be iteratively adjusted by the data scientist (R3) either based on the feedback coming from the experimental model engineering stage or from the monitoring component observing the model performance.

(B2) Feature engineering pipeline. The initially defined requirements for the feature engineering pipeline are taken by the data engineer (R4) and software engineer (R5) as a starting point to build up the prototype of the feature engineering pipeline. The initially defined requirements and rules are updated according to the iterative feedback coming either from the experimental model engineering stage or from the monitoring component observing the model's performance in production.

As a foundational requirement, the data engineer (R4) defines the code required for the CI/CD (C1) and orchestration component (C3) to ensure the task orchestration of the feature engineering pipeline. This role also defines the underlying infrastructure resource configuration. (8) First, the feature engineering pipeline connects to the raw data, which can be (for instance) streaming data, static batch data, or data from any cloud storage. (9) The data will be extracted from the data sources. (10) The data preprocessing begins with data transformation and cleaning tasks. The transformation rule artifact defined in the requirement gathering stage serves as input for this task, and the main aim of this task is to bring the data into a usable format. These transformation rules are continuously improved based on the feedback. (11) The feature engineering task calculates new and more advanced features based on other features. The predefined feature engineering rules serve as input for this task. These feature engineering rules are continuously improved based on the feedback. (12) Lastly, a data ingestion job loads batch or streaming data into the feature store system (C4). The target can either be the offline or online database (or any kind of data store). An example of the implementation of an entire feature engineering pipeline can be found in Esmaeilzadeh et al. [52], who implemented an NLP pipeline with Apache Spark. As another example, Xu [53] demonstrates how a financial institution may use Spark to process and analyze large amounts of customer credit data, such as credit history, income, and demographics. The data is then transformed and cleaned using Spark's DataFrame and SQL functionality, and various feature engineering techniques are applied to create a set of relevant features for the credit risk model. These features can be then passed through an ML pipeline, also implemented in Spark, to train and evaluate a predictive model for assessing credit risk. In addition, Apache Kafka can be used for near real-time streaming data ingestion into the Spark-based feature engineering pipeline [54]. However, to some extent, a traditional ETL tool can be used to build a feature engineering pipeline [55].

(C) Experimentation. Most tasks in the experimentation stage are led by the data scientist (R3) including the initial configuration of the hardware and runtime environment. The data scientist is supported by the software engineer (R5). (13) The data scientist (R3) connects to the feature store system (C4) for the data analysis. (Alternatively, the data scientist (R3) can also connect to the raw data for an initial analysis.) In case of any required data adjustments, the data scientist (R3) reports the required changes back to the data engineering zone (feedback loop). (14) Then the preparation and validation of the data coming from the feature store system is required. This task also includes the train and test split dataset creation. (15) The data scientist (R3) estimates the best-performing algorithm and hyperparameters, and the model training is then triggered with the training data (C5). The software engineer (R5) supports the data scientist (R3) in the creation of well-engineered model training code. (16) Different model parameters are tested and validated interactively during several rounds of model training. Once the performance metrics indicate good results, the iterative training stops. The best-performing model parameters are identified via parameter tuning. The model training task and model validation task are then iteratively repeated; together, these tasks can be called "model engineering." The model engineering aims to identify the best-performing algorithm and hyperparameters for the model. (17) The data scientist (R3) exports the model and commits the code to the repository. As a foundational requirement, either the DevOps engineer (R6) or the ML engineer (R7) defines the code for the (C2) automated ML workflow pipeline and commits it to the repository. Once either the data scientist (R3) commits a new ML model or the DevOps engineer (R6) and the ML engineer (R7) commits new ML workflow pipeline code to the repository, the CI/CD component (C1) detects the updated code and triggers automatically the CI/CD pipeline carrying out the build, test, and delivery steps. The build step creates artifacts containing the ML model and tasks of the ML workflow pipeline. The test step validates the ML model and ML workflow pipeline code. The delivery step pushes the versioned artifact(s)—such as images—to the artifact store (e.g., image registry).

Typical technologies used for the experimentation step are notebook-based solutions like the ones from Jupyter. One example of an industry case where ML experiments are performed with a notebook-based environment is in the field of natural language processing (NLP) [56]. A company that provides NLP-based services such as sentiment analysis, text summarization, and named entity recognition, may use Jupyter notebooks to perform ML experiments on large amounts of text data. The company's data scientists use Jupyter notebooks to prepare the data. Then, they can train, evaluate, and optimize different ML models, such as deep learning models, and test the results. To track the experiments with the textual data, i.e., the tracking of meta data and storing the resulting models, common-used solutions in combination with Jupyter are, among others, MLflow (i.e., Obeid [57] for assessing the risk of COVID-19 based on health records) or Neptune.AI (i.e., Aljabri [58] for NLP-based fake news detection).

(D) Automated ML workflow pipeline. The DevOps engineer (R6) and the ML engineer (R7) take care of the management of the automated ML workflow pipeline. They also manage the runtime environments, the underlying model training infrastructure in the form of hardware resources and frameworks supporting computation such as Kubernetes (C5). The workflow orchestration component (C3) orchestrates the tasks of the automated ML workflow pipeline. For each task, the required artifacts (e.g., images) are pulled from the artifact store (e.g., image registry). Each task can be executed via an isolated environment (e.g., containers). Finally, the workflow orchestration component (C3) gathers metadata for each task in the form of logs, completion time, and so on.

Once the automated ML workflow pipeline is triggered, each of the following tasks is managed automatically: (18) automated pulling of the versioned features from the feature store systems (data extraction). Depending on the use case, features are extracted from either the offline or online database (or any kind of data store). (19) Automated data preparation and validation; in addition, the train and test split is defined automatically. (20) Automated final model training on new unseen data (versioned features). The algorithm and hyperparameters are already predefined based on the settings of the previous experimentation stage. The model is retrained and refined. (21) Automated model evaluation and iterative adjustments of hyperparameters are executed, if required. Once the performance metrics indicate good results, the automated iterative training stops. The automated model training task and the automated model validation task can be iteratively repeated until a good result has been achieved. (22) The trained model is then exported and (23) pushed to the model registry (C6), where it is stored e.g., as code or containerized together with its associated configuration and environment files.

For all training job iterations, the ML metadata store (C7) records metadata such as parameters to train the model and the resulting performance metrics. This also includes the tracking and logging of the training job ID, training date and time, duration, and sources of artifacts. Additionally, the model specific metadata called "model lineage" combining the lineage of data and code is tracked for each newly registered model. This includes the source and version of the feature data and model training code used to train the model. Also, the model version and status (e.g., staging or production-ready) is recorded.

Once the status of a well-performing model is switched from staging to production, it is automatically handed over to the DevOps engineer or ML engineer for model deployment. From there, the (24) CI/CD component (C1) triggers the continuous deployment pipeline. The production-ready ML model and the model serving code are pulled (initially prepared by the software engineer (R5)). The continuous deployment pipeline carries out the build and test step of the ML model and serving code and deploys the model for production serving. The (25) model serving component (C8) makes predictions on new, unseen data coming from the feature store system (C4). This component can be designed by the software engineer (R5) as online inference for real-time predictions or as batch inference for predictions concerning large volumes of input data. For real-time predictions, features must come from the online database (low latency), whereas for batch predictions, features can be served from the offline database (normal latency). Model-serving applications are often configured within a container and prediction requests are handled via a REST API. When deploying an ML/AI application, it's a good practice to use A/B testing to determine in a real-world scenario which model performs better compared to another model, for example, deploying a "challenger model" in addition to an existing "champion model" to find out which one performs better by collecting feedback, for example, when predicting hotel booking cancellations [61].

As a foundational requirement, the ML engineer (R7) manages the model-serving computation infrastructure. The (26) monitoring component (C9) observes continuously the model-serving performance and infrastructure in real-time. Once a certain threshold is reached, such as detection of low prediction accuracy, the information is forwarded via the feedback loop. The (27) feedback loop is connected to the monitoring component (C9) and ensures fast and direct feedback allowing for more robust and improved predictions. It enables continuous training, retraining, and improvement. With the support of the feedback loop, information is transferred from the model monitoring component to several upstream receiver points, such as the experimental stage, data engineering zone, and the scheduler (trigger). The feedback to the experimental stage is taken forward by the data scientist for further model improvements. The feedback to the data engineering zone allows for the adjustment of the features prepared for the feature store system. Additionally, the detection of concept drifts as a feedback mechanism can enable (28) continuous training. Concept drifts occur in real-world applications when the input data changes over time e.g., when a sensor breaks. Decreased prediction accuracy due to concept drift can be detected by a certain concept drift detection algorithm [59]. Once the model-monitoring component (C9) detects a drift in the data [60], the information is forwarded to the scheduler, which then triggers the automated ML workflow pipeline for retraining (continuous training). As explained, a change in adequacy of the deployed model can be detected using distribution comparisons to identify drift. Retraining is not only triggered automatically when a statistical threshold is reached; it can also be triggered when new feature data is available, or it can be scheduled periodically.

Typical technologies supporting the automated ML workflow pipeline are, among others, Apache Airflow, Kubeflow Pipelines, IBM Watson Studio Pipelines, or SageMaker Pipelines. One example of an industry use case for an automated machine learning workflow pipeline using Airflow is in the field of online advertising [62]. A company may use Airflow to automate the process of training and deploying machine learning models for ad targeting and optimization. The pipeline starts by extracting, transforming, and loading large amounts of data from various sources, such as website clickstream data, user demographics, and campaign performance data. This data is then passed through a series of preprocessing and feature engineering steps, implemented as Airflow operators. Next, different machine learning models are trained and evaluated on the processed data, also using Airflow operators. The best-performing model is then deployed to a production environment, where it is used to make real-time ad targeting decisions. In this case, Apache Airflow is used to automate the entire process, including scheduling, monitoring, and re-running failed tasks.

VI. CONCEPTUALIZATON

With the findings at hand, we conceptualize the literature and interviews. It becomes obvious that the term MLOps is positioned at the intersection of machine learning, software engineering, DevOps, and data engineering (see Figure 5). We define MLOps as follows:

MLOps (Machine Learning Operations) is a paradigm, including aspects like best practices, sets of concepts, as well as a development culture when it comes to the end-to-end conceptualization, implementation, monitoring, deployment, and scalability of machine learning products. Most of all, it is an engineering practice that leverages three contributing disciplines: machine learning, software engineering (especially DevOps), and data engineering. MLOps is aimed at productionizing machine learning systems by bridging the gap between development (Dev) and operations (Ops). Essentially, MLOps aims to facilitate the creation of machine learning products by leveraging these principles: CI/CD automation, workflow orchestration, reproducibility; versioning of data, model, and code; collaboration; continuous ML training and evaluation; ML metadata tracking and logging; continuous monitoring; and feedback loops.

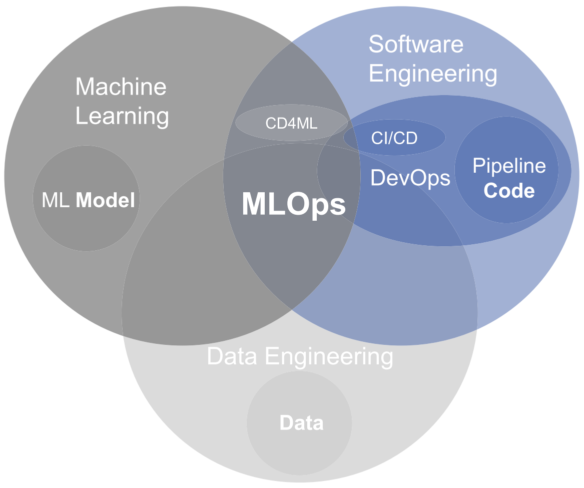

FIGURE 5. **Intersection of disciplines of the MLOps paradigm.**

VII. OPEN CHALLENGES

Several challenges for adopting MLOps have been identified after conducting the literature review, tool review, and interview study. These open challenges have been organized into the categories of organizational, ML system, and operational challenges.

A. ORGANIZATIONAL CHALLENGES

The mindset and culture of data science practice is a typical challenge in organizational settings [63]. As our insights from literature and interviews show, to successfully develop and run ML products, there needs to be a culture shift away from model-driven machine learning toward a product-oriented discipline [γ]. The recent trend of data-centric AI also addresses this aspect by putting more focus on the data-related aspects taking place prior to the ML model building. Especially the roles associated with these activities should have a product-focused perspective when designing ML products [γ]. A great number of skills and individual roles are required for MLOps (β). As our identified sources point out, there is a lack of highly skilled experts for these roles—especially with regard to architects, data engineers, ML engineers, and DevOps engineers [26], [32], [38], [α, ε]. This is related to the necessary education of future professionals—as MLOps is typically not part of data science education [33] [γ]. Posoldova [6] further stresses this aspect by remarking that students should not only learn about model creation, but must also learn about technologies and components necessary to build functional ML products.

Data scientists alone cannot achieve the goals of MLOps. A multi-disciplinary team is required [25], thus MLOps needs to be a group process [α]. This is often hindered because teams work in silos rather than in cooperative setups [α]. Additionally, different knowledge levels and specialized terminologies make communication difficult. To lay the foundations for more fruitful setups, the respective decision-makers need to be convinced that an increased MLOps maturity and a product-focused mindset will yield clear business improvements [γ].

B. ML SYSTEM CHALLENGES

A major challenge with regard to MLOps systems is designing for fluctuating demand, especially in relation to the process of ML training [33]. This stems from potentially voluminous and varying data [28], which makes it difficult to precisely estimate the necessary infrastructure resources (CPU, RAM, and GPU) and requires a high level of flexibility in terms of scalability of the infrastructure [7], [33], [δ].

C. OPERATIONAL CHALLENGES

In productive settings, it is challenging to operate ML manually due to different stacks of software and hardware components and their interplay as well as the selection of both ([64], [65]. Therefore, robust automation is required [24], [33]. Also, a constant incoming stream of new data forces retraining capabilities. This is a repetitive task which, again, requires a high level of automation [66], [θ]. These repetitive tasks yield a large number of artifacts that require a strong governance [27], [32], [45] as well as versioning of data, model, and code to ensure robustness and reproducibility [7], [32], [39]. Lastly, it is challenging to resolve a potential support request (e.g., by finding the root cause), as many parties and components are involved. Failures can be a combination of ML infrastructure and software within the MLOps stack [7].

VIII. CONCLUSION

With the increase of data availability and analytical capabilities, coupled with the constant pressure to innovate, more machine learning products than ever are being developed. However, only a small number of these proofs of concept progress into deployment and production. Furthermore, the academic space has focused intensively on machine learning model building and benchmarking, but too little on operating complex machine learning systems in real-world scenarios. In the real world, we observe data scientists still managing ML workflows manually to a great extent. The paradigm of Machine Learning Operations (MLOps) addresses these challenges. In this work, we shed more light on MLOps. By conducting a mixed-method study analyzing existing literature and tools, as well as interviewing eight experts from the field, we uncover four main aspects of MLOps: its principles, components, roles, and architecture. From these aspects, we infer a holistic definition. The results support a common understanding of the term MLOps and its associated concepts, and will hopefully assist researchers and professionals in setting up successful ML products in the future.

REFERENCES

[1] M. Aykol, P. Herring, and A. Anapolsky, "Machine learning for continuous innovation in battery technologies," *Nature Rev. Mater.*, vol. 5, no. 10, pp. 725–727, Jun. 2020.

[2] M. K. Gourisaria, R. Agrawal, G. M. Harshvardhan, M. Pandey, and S. S. Rautaray, "Application of machine learning in industry 4.0," in *Machine Learning: Theoretical Foundations and Practical Applications*. Cham, Switzerland: Springer, 2021, pp. 57–87.

[3] A. D. L. Heras, A. Luque-Sendra, and F. Zamora-Polo, "Machine learning technologies for sustainability in smart cities in the post-COVID era," *Sustainability*, vol. 12, no. 22, p. 9320, Nov. 2020.

[4] R. Kocielnik, S. Amershi, and P. N. Bennett, "Will you accept an imperfect AI?: Exploring designs for adjusting end-user expectations of AI systems," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2019, pp. 1–14.

[5] R. van der Meulen and T. McCall. (2018). *Gartner Says Nearly Half of CIOs Are Planning to Deploy Artificial Intelligence*. Accessed: Dec. 4, 2021. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-02-13-gartner-says-nearly-half-of-cios-are-planning-to-deploy-artificial-intelligence

[6] A. Posoldova, "Machine learning pipelines: From research to production," *IEEE Potentials*, vol. 39, no. 6, pp. 38–42, Nov. 2020.

[7] L. E. Lwakatare, I. Crnkovic, E. Rånge, and J. Bosch, "From a data science driven process to a continuous delivery process for machine learning systems," in *Product-Focused Software Process Improvement* (Lecture Notes in Computer Science), vol. 12562. Springer, 2020, pp. 185–201, doi: 10.1007/978-3-030-64148-1_12.

[8] W. W. Royce, "Managing the development of large software systems," in *Proc. IEEE WESCON*, Aug. 1970, pp. 1–9.

[9] K. Beck et al., "The agile manifesto," 2001. [Online]. Available: http://agilemanifesto.org/

[10] P. Debois. (2009). *Patrick Debois Devopsdays Ghent*. Accessed: Mar. 25, 2021. [Online]. Available: https://devopsdays.org/events/2019-ghent/speakers/patrick-debois/

[11] S. Mezak. (Jan. 25, 2018). *The Origins of DevOps: What's in a Name? DevOps.com*. Accessed: Mar. 25, 2021. [Online]. Available: https://devops.com/the-origins-of-devops-whats-in-a-name/

[12] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles, "A survey of DevOps concepts and challenges," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–35, Nov. 2020, doi: 10.1145/3359981.

[13] R. W. Macarthy and J. M. Bass, "An empirical taxonomy of DevOps in practice," in *Proc. 46th Euromicro Conf. Softw. Eng. Adv. Appl. (SEAA)*, Aug. 2020, pp. 221–228, doi: 10.1109/SEAA51224.2020.00046.

[14] M. Rütz, "DEVOPS: A systematic literature review," Inf. Softw. Technol., FH-Wedel, Aug. 2019. [Online]. Available: https://www.researchgate.net/publication/335243102_DEVOPS_A_SYSTEMATIC_LITERATURE_REVIEW

[15] P. Perera, R. Silva, and I. Perera, "Improve software quality through practicing DevOps," in *Proc. 17th Int. Conf. Adv. ICT Emerg. Regions (ICTer)*, Sep. 2017, pp. 1–6.

[16] J. Webster and R. Watson, "Analyzing the past to prepare for the future: Writing a literature review," *MIS Quart.*, vol. 26, no. 2, pp. 8–23, 2002. [Online]. Available: https://www.jstor.org/stable/4132319, doi: 10.1.1.104.6570.

[17] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—A systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, Jan. 2009, doi: 10.1016/j.infsof.2008.09.009.

[18] M. D. Myers and M. Newman, "The qualitative interview in IS research: Examining the craft," *Inf. Org.*, vol. 17, no. 1, pp. 2–26, Jan. 2007, doi: 10.1016/j.infoandorg.2006.11.001.

[19] U. Schultze and M. Avital, "Designing interviews to generate rich data for information systems research," *Inf. Org.*, vol. 21, no. 1, pp. 1–16, Jan. 2011, doi: 10.1016/j.infoandorg.2010.11.001.

[20] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative Sociol.*, vol. 13, no. 1, pp. 3–21, 1990, doi: 10.1007/BF00988593.

[21] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. London, U.K.: Aldine, 1967, doi: 10.4324/9780203793206.

[22] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen, "MLOps challenges in multi-organization setup: Experiences from two real-world cases," 2021, *arXiv:2103.08937*.

[23] Y. Zhou, Y. Yu, and B. Ding, "Towards MLOps: A case study of ML pipeline platform," in *Proc. Int. Conf. Artif. Intell. Comput. Eng. (ICAICE)*, Oct. 2020, pp. 494–500, doi: 10.1109/ICAICE51518.2020.00102.

[24] I. Karamitsos, S. Albarhami, and C. Apostolopoulos, "Applying devops practices of continuous automation for machine learning," *Information*, vol. 11, no. 7, pp. 1–15, 2020, doi: 10.3390/info11070363.

[25] A. Goyal, "MLOps machine learning operations," *Int. J. Inf. Technol. Insights Transformations*, vol. 4, no. 2, 2020. Accessed: Apr. 15, 2021. [Online]. Available: http://technology.eurekajournals.com/index.php/IJITIT/article/view/655

[26] D. A. Tamburri, "Sustainable MLOps: Trends and challenges," in *Proc. 22nd Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Sep. 2020, pp. 17–23, doi: 10.1109/SYNASC51798.2020.00015.

[27] O. Spjuth, J. Frid, and A. Hellander, "The machine learning life cycle and the cloud: Implications for drug discovery," *Expert Opinion Drug Discovery*, vol. 16, no. 9, pp. 1071–1079, 2021, doi: 10.1080/17460441.2021.1932812.

[28] B. Derakhshan, A. R. Mahdiraji, T. Rabl, and V. Markl, "Continuous deployment of machine learning pipelines," in *Proc. EDBT*, Mar. 2019, pp. 397–408, doi: 10.5441/002/edbt.2019.35.

[29] R. R. Karn, P. Kudva, and I. A. M. Elfadel, "Dynamic autoselection and autotuning of machine learning models for cloud network analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1052–1064, May 2019, doi: 10.1109/TPDS.2018.2876844.

[30] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.

[31] A. Molner Domenech and A. Guillén, "Ml-experiment: A Python framework for reproducible data science," *J. Phys., Conf. Ser.*, vol. 1603, no. 1, Sep. 2020, Art. no. 012025, doi: 10.1088/1742-6596/1603/1/012025.

[32] S. Makinen, H. Skogstrom, E. Laaksonen, and T. Mikkonen, "Who needs MLOps: What data scientists seek to accomplish and how can MLOps help?" in *Proc. IEEE/ACM 1st Workshop AI Eng. Softw. Eng. AI (WAIN)*, May 2021, pp. 109–112.

[33] L. C. Silva, F. R. Zagatti, B. S. Sette, L. N. dos Santos Silva, D. Lucredio, D. F. Silva, and H. de Medeiros Caseli, "Benchmarking machine learning solutions in production," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2020, pp. 626–633, doi: 10.1109/ICMLA51294.2020.00104.

[34] A. Banerjee, C. C. Chen, C. C. Hung, X. Huang, Y. Wang, and R. Chevesaran, "Challenges and experiences with MLOps for performance diagnostics in hybrid-cloud enterprise software deployments," in *Proc. OpML USENIX Conf. Oper. Mach. Learn.*, 2020, pp. 7–9.

[35] B. Benni, M. Blay-Fornarino, S. Mosser, F. Precioso, and G. Jungbluth, "When DevOps meets meta-learning: A portfolio to rule them all," in *Proc. ACM/IEEE 22nd Int. Conf. Model Driven Eng. Lang. Syst. Companion (MODELS-C)*, Sep. 2019, pp. 605–612, doi: 10.1109/MODELS-C.2019.00092.

[36] C. Vuppalapati, A. Ilapakurti, K. Chillara, S. Kedari, and V. Mamidi, "Automating tiny ML intelligent sensors DevOPS using Microsoft azure," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2020, pp. 2375–2384, doi: 10.1109/BigData50022.2020.9377755.

[37] Á. L. García, J. M. D. Lucas, M. Antonacci, W. Z. Castell, and M. David, "A cloud-based framework for machine learning workloads and applications," *IEEE Access*, vol. 8, pp. 18681–18692, 2020, doi: 10.1109/ACCESS.2020.2964386.

[38] C. Wu, E. Haihong, and M. Song, "An automatic artificial intelligence training platform based on kubernetes," in *Proc. 2nd Int. Conf. Big Data Eng. Technol.*, Jan. 2020, pp. 58–62, doi: 10.1145/3378904.3378921.

[39] G. Fursin, "Collective knowledge: Organizing research projects as a database of reusable components and portable workflows with common interfaces," *Phil. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 379, no. 2197, May 2021, Art. no. 20200211, doi: 10.1098/rsta.2020.0211.

[40] M. Schmitt, "Airflow vs. Luigi vs. Argo vs. MLFlow vs. KubeFlow," Tech. Rep., 2022. [Online]. Available: https://www.datarevenue.com/en-blog/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow

[41] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, May 2010, pp. 1–10.

[42] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th ACM Symp. Operating Syst. Princ.*, Oct. 2003, pp. 29–43.

[43] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *ACM SIGOPS Oper. Syst. Rev.*, vol. 44, no. 2, pp. 35–40, Apr. 2010.

[44] J. C. Corbett, "Spanner: Google's globally distributed database," *ACM Trans. Comput. Syst. (TOCS)*, vol. 31, no. 3, pp. 1–22, 2013.

[45] Y. Liu, Z. Ling, B. Huo, B. Wang, T. Chen, and E. Mouine, "Building a platform for machine learning operations from open source frameworks," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 704–709, 2020, doi: 10.1016/j.ifacol.2021.04.161.

[46] G. S. Yoon, J. Han, S. Lee, and J. W. Kim, *DevOps Portal Design for SmartX AI Cluster Employing Cloud-Native Machine Learning Workflows*, vol. 47. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-39746-3_54.

[47] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[48] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[49] L. E. Lwakatare, I. Crnkovic, and J. Bosch, "DevOps for AI—Challenges in development of AI-enabled applications," in *Proc. Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, Sep. 2020, pp. 1–6, doi: 10.23919/SoftCOM50211.2020.9238323.

[50] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlaš, W. Wu, and C. Zhang, "A data quality-driven view of MLOps," 2021, *arXiv:2102.07750*.

[51] W. J. van den Heuvel and D. A. Tamburri, *Model-Driven ML-Ops for Intelligent Enterprise Applications: Vision, Approaches and Challenges*, vol. 391. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-52306-0_11.

[52] A. Esmaeilzadeh, M. Heidari, R. Abdolazimi, P. Hajibabaee, and M. Malekzadeh, "Efficient large scale NLP feature engineering with Apache spark," in *Proc. IEEE 12th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2022, pp. 274–280.

[53] J. Xu, "MLOps in the financial industry: Philosophy, practices, and tools," in *Future and Fintech, the, Abcdi and Beyond*. Singapore: World Scientific, 2022, p. 451, doi: 10.1142/9789811250903_0014.

[54] F. Carcillo, A. D. Pozzolo, Y.-A. L. Borgne, O. Caelen, Y. Mazzer, and G. Bontempi. *SCARFF?: A Scalable Framework for Streaming Credit Card Fraud Detection With Spark 1*. Accessed: Feb. 17, 2023. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[55] J. Dhanalakshmi and N. Ayyanathan, "A dynamic web data extraction from SRLDC (southern regional load dispatch centre) and feature engineering using ETL tool," in *Proc. 2nd Int. Conf. Artif. Intell., Adv. Appl.* Springer, 2022, pp. 443–449, doi: 10.1007/978-981-16-6332-1_38.

[56] J. Foster and J. Wagner, "Naive Bayes versus BERT: Jupyter notebook assignments for an introductory NLP course," in *Proc. 5th Workshop Teaching (NLP)*, 2021, pp. 112–114.

[57] J. S. Obeid, M. Davis, M. Turner, S. M. Meystre, P. M. Heider, E. C. O'Bryan, and L. A. Lenert, "An artificial intelligence approach to COVID-19 infection risk assessment in virtual visits: A case report," *J. Amer. Med. Inform. Assoc.*, vol. 27, no. 8, pp. 1321–1325, Aug. 2020.

[58] M. Aljabri, D. M. Alomari, and M. Aboulnour, "Fake news detection using machine learning models," in *Proc. 14th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2022, pp. 473–477.

[59] L. Baier, N. Kühl, and G. Satzger, "How to cope with change?—Preserving validity of predictive services over time," in *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, 2019, pp. 1–10.

[60] L. Baier, T. Schlör, J. Schöffer, and N. Kühl, "Detecting concept drift with neural network model uncertainty," 2021, *arXiv:2107.01873*.

[61] N. Antonio, A. de Almeida, and L. Nunes, "Predicting hotel bookings cancellation with a machine learning classification model," in *Proc. 16th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2017, pp. 1049–1054, doi: 10.1109/ICMLA.2017.00-11.

[62] T. Cui, Y. Wang, and B. Namih, "Build an intelligent online marketing system: An overview," *IEEE Internet Comput.*, vol. 23, no. 4, pp. 53–60, Jul. 2019.

[63] L. Baier and S. Seebacher, "Challenges in the Deployment and," in *Proc. 27th Eur. Conf. Inf. Syst. (ECIS)*, Stockholm, Sweden, Jun. 2019, pp. 1–15. [Online]. Available: https://aisel.aisnet.org/ecis2019_rp/163

[64] P. Ruf, M. Madan, C. Reich, and D. Ould-Abdeslam, "Demystifying MLOps and presenting a recipe for the selection of open-source tools," *Appl. Sci.*, vol. 11, no. 19, p. 8861, Sep. 2021.

[65] N. Hewage and D. Meedeniya, "Machine learning operations: A survey on MLOps tool support," 2022, *arXiv:2202.10169*.

[66] B. Karlaš, M. Interlandi, C. Renggli, W. Wu, C. Zhang, D. M. I. Babu, J. Edwards, C. Lauren, A. Xu, and M. Weimer, "Building continuous integration services for machine learning," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 2407–2415, doi: 10.1145/3394486.3403290.

DOMINIK KREUZBERGER received the B.Sc. degree in business information systems as part of a dual study program from Baden-Wuerttemberg Cooperative State University (DHBW), Stuttgart, and the M.Sc. degree in information systems engineering and management with a focus on digital services and machine learning operations (MLOps) from the Karlsruhe Institute of Technology (KIT). He is currently an IT architect, specializing in hybrid cloud computing and artificial intelligence solutions. He is also with IBM with a focus on client success, where he designs and builds enterprise-grade data and machine-learning products based on IBM technology. Before joining IBM, he worked for nearly a decade with the multinational sports company adidas. During this time, he held various positions in the area of e-commerce and data and analytics.

NIKLAS KÜHL received the Ph.D. degree (summa cum laude) in information systems with a focus on applied machine learning. In his studies, he is working on conceptualizing, designing, and implementing artificial intelligence (AI) products with a focus on inter-organizational learning as well as fair and effective collaboration within human–AI teams. He is currently a Full Professor of information systems and human-centric AI with the University of Bayreuth. He is also a Group Lead with Fraunhofer FIT for business analytics as well as a Senior Expert in artificial intelligence with IBM. In the past, he was a Managing Consultant for Data Science with IBM, which complemented his theoretical knowledge with practical insights from the field. He has been working on machine learning (ML) and AI in different domains, since 2014. He is internationally collaborating with multiple institutions such as the University of Texas and the MIT–IBM Watson AI Laboratory.

SEBASTIAN HIRSCHL is currently a Senior Engineer/Architect with IBM and leads the ML engineering and platform activities for the machine learning practice in Germany. He combines his computer science background in machine learning and artificial intelligence. He developed the discipline of machine learning (ML) engineering within IBM over the last five years, including best practices, methods, roles, and tools. He designs and leads the implementation of enterprise-grade data and ML products for clients in Germany and Europe. Together with a team, he publishes the IBM Data Science Best Practices as well as shapes the IBM Data and AI reference architecture. In his role as an ML engineering expert, he drives the evolution of the paradigm of MLOps internally and externally.