A Systematic Analysis of MLOps Features and Platforms

Leonhard Faubel
University of Hildesheim
Institute of Computer Science
Software Systems Engineering
Hildesheim, Germany, 31141
faubel@uni-hildesheim.de

Klaus Schmid
University of Hildesheim
Institute of Computer Science
Software Systems Engineering
Hildesheim, Germany, 31141
faubel@uni-hildesheim.de

Abstract—**While many companies aim to use Machine Learning (ML) models, transitioning to deployment and practical application of such models can be very time-consuming and technically challenging. To address this, MLOps (ML Operations) offers processes, tools, practices, and patterns to bring ML models into operation. A large number of tools and platforms have been created to support developers in creating practical solutions. However, specific needs vary strongly in a situation-dependent manner, and a good overview of their characteristics is missing, making the architect's task very challenging. We conducted a systematic literature review (SLR) of MLOps platforms, describing their qualities, features, tactics, and patterns. In this paper, we want to map the design space of MLOps platforms. We are guided by the Attribute-Driven Design (ADD) methodology. In this way, we want to provide software architects with a tool to support their work in the platform area.**

Keywords-MLOps, Design space, ML platforms

I. INTRODUCTION (HEADING 1)

While machine learning (ML) becomes increasingly important to company success, the continuous transition from model development to deployment in the field becomes increasingly important. Companies may take a month or even more to deploy an ML model. One reason is that data scientists often lack knowledge of Software Engineering (SE) and computer science in general. While they are able to solve business problems with analytics and ML algorithms, they often struggle to deploy these algorithms into software systems in the real world. MLOps engineers fill this gap with additional knowledge in software engineering and automation [1]. On the other hand, SE processes have to be adapted to ML lifecycles [2] to work properly. ML brings in novel characteristics into existing SE methods: It requires trained models, is often unpredictable, and copes with constantly changing data.

MLOps provides processes, tools, practices, and patterns to close this gap. It aims at increasing quality attributes like correctness or reliability [3]. MLOps relies on DevOps principles, an attempt to automate the deployment processes of new software versions in software engineering and extents this by integrating ML relevant adaptations of the processes. MLOps provides best practices and guiding principles around ML, including collaborative work, reproducibility, continuous delivery, testing, and monitoring [4]. It aims to increase quality, simplify the management process, and automate the deployment process of bringing ML models into production [5]. Digital platforms can be defined as products, services, or technologies that serve as a basis for a large number of companies and offer complementary products, services products, services, and technologies [6]. MLOps platforms are platforms that manage ML pipelines [7] and try to satisfy the abovementioned gaps and capabilities. Since most platforms have many overall features in common, they are mainly different in their particular application [4]. This paper discusses the design space of such MLOps platforms from a technical software engineering perspective. It categorizes and provides an overview as decision support for platform developers and software architects who choose platforms as an integration component of more extensive applications. Initially, 70 MLOps platforms were compared based on a search on Google and Google Scholar to gain a general understanding and define terms. Based on that knowledge, an SLR was conducted as described in Section III. The results of our analysis led to a thorough description of characteristics of functions relevant to MLOps as well as example tools that implement them. Together with descriptions of relevant tactics this provides the core of our analysis of the design space of MLOps platforms as given in Section IV. In Section V, we particularly describe some major platforms, while Section VI provides an analysis and Section VII concludes.

A. Background

Traditional systems are static and often meet a company's needs in terms of speed, reliability, and the ability to predict its outcome [3]. ML models, in contrast, are different: They have inferences whose function cannot always be verified. Sometimes, their behavior is neither predictable nor fully repeatable. Data, code, and models are cross-dependent, and data, pre-processing, and training are interrelated in very complex ways. In production, input changes may cause inadvertent behavior. Changes in data or parameters cause changes in the inference [8], so focusing on mitigation strategies is especially important.

MLOps as a collection of techniques, tools, practices, or processes for ML deployment in production [9], [10], [11].

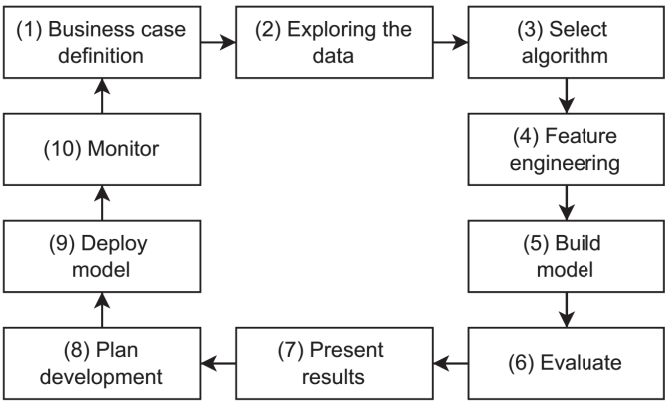

Figure 1: ML life-cycle adapted from [17].

aims to increase the level of automation [12], [13], [14]. A (1) business problem must be solved. Initially, the problem has to be fully understood. Further, the problem should be specified. Data should be acquired or made available for further steps [15]. For this, raw data must be processed to extract information, remove erroneous data, and bring it to a reasonable shape [16]. Then, the data needs to be (2) explored [15]. This involves finding correlations in the data by data scientists in collaboration with business domain experts to find technical challenges and first correlations in the data. Then, a fitting algorithm to the problem is (3) selected. (4) Feature selection means checking for similarities and impact on the prediction goals and eventually transforming raw data into other representations. A model is (5) built using the selected algorithm with appropriate features to use the correlation or patterns to predict. This model is (6) evaluated by checking how well the model performs using data that has not been used for training. The results are (7) presented and should be compared to other methods. This includes an analysis of the advantages: does its usage exceed the benefits or the cost? Then follows the (8) design of a pipeline that suits the needs of prior steps. After the model is (9) deployed as a module with interfaces (serving the model), it is (10) monitored by controlling the model behavior over time: often, there are underlying problems like drift under real-world
conditions, leading to anomaly and misbehavior of the model. Consequently, a typical ML life-cycle contains the activities shown in Figure 1.

II. Related Work

This section reviews literature related to the design space of MLOps platforms. We focus on architectural descriptions, implemented MLOps platforms, and systematic reviews. Several publications discuss MLOps architecture based on functional components, tools, and software infrastructures [18], [19], [20], [21]. Some implementations of MLOps platforms consider functional aspects as components to showcase the benefits of introducing MLOps tools [22], [23]. The architectures of such implementations are often domain-specific, e.g., Nia et al. [24] present an MLOps infrastructure for model deployment for telecom data that has been implemented and evaluated using metrics. Furthermore, authors reveal their design decisions by comparing features offered by different platforms, e.g., Zarate et al. [25] compare different tools based on predefined features, and Recupito et al. [26] conducted a multivocal literature review of the most common MLOps tools in which they examined features and functionalities. Compared to other literature reviews that address features, in this paper we look at 70 MLOps platforms and consider features from an SE perspective as a solution approach for software architects. Architectural design solutions are described. These include reference architectures based on components [18], [27], software architecture patterns [28], and cloud computing design patterns for MLOps [29].

III. Method

The method for this systematic literature review (SLR) followed the guidelines for performing SLRs in Software Engineering [30] and is described in more detail in a technical report [31]. At first, a search on the topic was carried out via Google and Google Scholar. Based on the results, an overview of MLOps platforms and an initial selection of published studies were obtained. This initial automated search helped to develop an initial understanding of the topic and to define terms and expressions that served as the basis for the search in this SLR. A comprehensive search strategy was developed to identify relevant studies. The search was performed in academic search engines including ACM Digital Library, IEEE Explore, and Science Direct using the keywords related to "MLOps" or "Machine Learning Operations". As a result, we included 47 white and grey literature studies and 95 self-published documents.

A. Research Questions

RQ1 How do MLOps platforms differ in their design?

RQ1.1 What features are commonly found in MLOps platforms?

RQ1.2 How do MLOps platforms support different qualities or characteristics in the context of their design and implementation?

RQ2 What are the current MLOps platforms?

RQ2.1 What can be learned from the current research results?

B. Selection Criteria

We included published research studies in English, such as conferences, journals, magazine papers, books, reports, and white papers using the term MLOps or Machine Learning Operations and dealing with MLOps platforms, automation, or software engineering published between January 2012 and May 2022. Self-published documents such as vendor homepages and blogs support our findings if no studies are available. Excluded are duplicate versions of studies, studies in languages other than English or German, studies without

reference to the research questions, talks without available information like protocols or notes, and posters.

Studies were excluded if they:

- Did not meet the inclusion criteria
- Were duplicates
- Were published in languages other than English

Different studies related to MLOps or Machine Learning Operations were excluded from the analysis. This was because MLOps can have various meanings, such as minimum linear operations or Multi-objective Lexicographic Optimization Problems. Additionally, the term Machine Learning Operations refers to supporting methods for ML, processing operations, or ML methods.

C. Data Extraction, Analysis, and Synthesis

Data extracted from the selected studies include item type, author(s), publication year, title, DOI, abstracts, and references. A thematic analysis approach was used to synthesize the findings and identify patterns across the selected studies. The quality of the included studies was assessed by checking whether the results were based on evidence and arguments, whether the study presented a research project, and whether the study's objectives were clearly defined. Studies that met these quality criteria were included in the final analysis.

D. Threats to Validity

Search on Google utilizes user preferences and factors such as search history and location, especially when no action is taken. Since MLOps is in the author's area of interest and he regularly searches for it concerning his work, we assume the bias could benefit our case. The Google and Google Scholar searches were incomplete, as there were too many results to review. However, the most relevant articles are listed first. During the assessment process, we identified several potential sources of bias that might have impacted the integrity of the findings. First, the identification of studies relied exclusively on selecting electronic databases. The initial search on Google and Google Scholar enabled us to check whether we had omitted any relevant studies. Further, bias could arise from only one author reviewing the articles, leading to subjective decisions. Another source of bias is the initial search of MLOps platforms, which helped select keywords and created an understanding of the study's vocabulary but did not exclude the inclusion of developer and commercial vendor articles that were not based on scientific methods. However, the information gathered there was purely technical and was supported by information from the subsequent SLR and our experiences on the topic.

IV. Design Space

This chapter describes features and design decisions in MLOps platforms. Features, which we consider as function groups, represent design decisions, while the quality aspects and criteria provide a rationale for these decisions.

Here we cover the quality aspects, technical features, and functionalities required in MLOps platforms. These include functionalities that support the activities of the MLOps lifecycle and features that increase the quality of MLOps. In some platforms, these features are implemented directly, while in others, they are made possible by openness and adaptability.

A. Quality Attributes

This section outlines context-specific quality attributes of MLOps platforms. Significantly more quality attributes are described in our technical report [31]. Figure 2 provides an overview of the exemplary properties described in detail below.

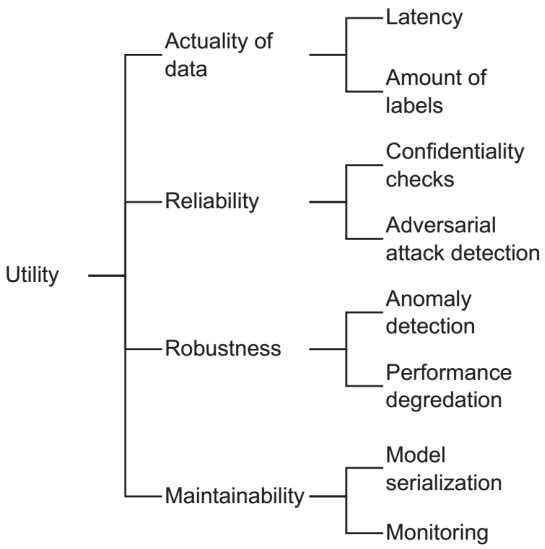

Figure 2: Context-specific overview of qualititesbased on a utility tree.

1) Actuality of Data: The higher the amount of labeled data, the higher the model's quality. Automatic labeling or label UIs can help gather new data. It may be beneficial to check if the correct labels exist and add a category for data that is not supposed to be predicted by the model to allow completeness. Additionally, the data should include other representations of each label. Standards in the data collection and labeling process allow consistency in terms of the expected data format and volume. However, labeling mechanisms have weaknesses since data must meet the intended needs and requirements to be relevant. This is not always granted due to human or machine error and must be checked frequently. That is why data cleaning is crucial. It helps to avoid typos, duplicate entries, measuring inaccuracies, missing features or values, and inaccurate labels. There may be a latency until the data appears in the database when information about a particular data point is recorded. Timeliness is essential when the system requires the latest and up-to-date data [17], [32]. Here, real-time environments for distributed processing of large datasets, data warehouses, and distributed search- engines [33] can help. Some of these environments support the high-performance processing of semi-structured in-memory data [34].

2) Reliability: Reliability is granted when a system continues to work correctly [35] and perform at the desired level, even in the face of adversarial attacks, hardware, software, or human errors [36]. Integrity and confidentiality checks and adversarial attack detection can face some of these challenges. CI, CD, and CT can avoid human errors [37].

3) Robustness: Robustness [38] is granted when a model can cope with an anomaly, performance degradation, and/or misbehavior.

4) Maintainability: Easing adapting to new data and maintaining the system behavior makes a system maintainable. Incorporating user feedback [39], [40], model serialization, monitoring, logging, model formatting, and documentation can help here [18].

B. Functional Groups

Several features must be available in the form of functional groups to support the MLOps lifecycle. We consider process, analytics, deployment, operations, and data storage as general function groups. These are described below for specific functions in design concept catalogs [41].

1) Process and Analytics: Data exploration, algorithm selection, feature engineering, model creation, and model performance evaluation are mandatory steps in machine learning, and they need to be supported by an MLOps platform (Table 1).

Table 1: Process and Analytics Catalog.

Specific Function	Duties	Example Tools
Data exploration	Visualization	Superset
		Grafana
	Data cleaning	pandas
	Data labeling	Doccano
		Label studio
Algorithm selection	AutoML	pycaret
		AutoGluon
		TransmogrifAI
	Hyperparameter tuning	Hyperopt
		Katib
		talos
Feature engineering	Feature extraction	Featuretools
		tsfresh
Model building	Machine learning	sklearn
		OpenCV
		MediaPipe
		Tensorflow
		Pytorch
		OpenNN

Data exploration: Identifying correlations in data and ensuring its suitability for analysis.

- **Visualization:** Representing data effectively to identify correlations and patterns. Transforming raw data for feature engineering and inference [15].
- **Data cleaning:** Detecting and correcting corrupt or inaccurate data values [42].
- **Data labeling:** Annotating or tagging data to enhance prediction accuracy [43].

Algorithm selection: Choosing an appropriate algorithm for the given problem.

- **AutoML:** Building, optimizing, and evaluating machine learning algorithms and pipelines, often with automated algorithm selection [44].
- **Hyperparameter tuning:** Automating the optimization of parameters [9].

Feature engineering: Identifying and extracting meaningful features from data.

- **Feature extraction:** Utilizing automated toolboxes to extract features from various types of datasets [45].

Model building: Constructing models using selected algorithms and learned features.

- **Machine learning:** Leveraging machine learning libraries for building and evaluating models.

2) Deployment and Operations: Further, there are deployment and operational features (Table 2). Some platforms allow models to be deployed in the cloud on the web, while others allow models to be deployed on-premise or on edge.

Table 2: Deployment and Operations Catalog.

Specific Function	Duties	Example Tools
Model as a Service	Model serialization	MLeap
		MLflow models
		pickle
		H20
	Model formatting	PMML
		PFA
		ONNX
	Model serving	spring
		tomcat
		django
		flask
Platform as a Service	Packaging and containers	Docker
		Kubernetes
		Kubeflow
Infrastructure as a Service	CPU cluster	CUDA
	GPU cluster TPU	OpenCL
	FPGA	Vitis AI
Communication	Service-oriented architecture	flask-RESTful
		gRPC
	Distributed event streaming	kafka
		RabbitMQ
Observability	Monitoring and Logging	Wavefront
		splunk
		datadog

Model as a Service (MaaS): Providing a fully functioning service for deploying models.

- **Model serialization:** Converting a trained model into a format or execution engine that can be easily saved, transferred, and reloaded without retraining.
- **Model formatting:** Structuring and arranging a serialized model to include metadata and apply necessary formats.
- **Model serving:** Serving a model as a service or dependency [46].

Platform as a Service (PaaS): Providing a platform for software development.

- **Packaging and containers:** Utilizing microservices for data pipelines [13]. Containers package code and depen-

dencies in an isolated environment, often managed by container orchestration platforms.

Infrastructure as a Service (IaaS): Providing computing resources as a foundation for cloud computing.

- **CPU Cluster:** Enabling CPUs to collaborate for computationally intensive ML tasks.
- **GPU Cluster:** Providing dynamic compute power like GPU clusters for training and scaling specific ML algorithms [47], [48].
- **TPU support:** Accelerating ML tasks with Tensor Processing Units (TPUs) for faster processing times [46].
- **FPGA support:** Using Field Programmable Gate Arrays (FPGAs) for ML tasks with faster processing times [49].

Communication: Enabling components and services to communicate effectively.

Service-oriented architecture (SOA): Creating loosely coupled services [50].

- **Distributed event streaming:** Managing the storage, distribution, and balancing of event streams across services [51].

Observability: Understanding system components for performance monitoring and issue detection.

- **Monitoring and logging:** Tracking model behavior over time, detecting data drift, anomalies, performance degradation, or misbehavior [15].

3) Data Storage: Repositories store data, code, models, and metadata. Specific databases are frequently used for various kinds of data and performance requirements (Table 3).

Table 3: Data storage catalog.

Specific Function	Duties	Example Tools
Versioning	Data repository	DVC
	Model repository	MLflow model repository
		DVC
	Code repository	GitHub
	Metadata repository	DVC
	Feature store	feast
NoSQL Databases	Reasoning and semantic	Neo4j
	Timescale	Cassandra
		MongoDB
		InfluxDB
	Caching and Queuing	Redis

Versioning: Tracking data, models, code, and metadata due to cross-dependencies.

- **Data repository:** Storing data along with its origin, acquisition details, and aggregations, often using different databases based on data type and performance requirements [15].
- **Model repository:** Storing mathematical models in machine-readable formats, possibly including validity checks [52].
- **Code repository:** Using distributed version control systems to manage code changes [37].

Metadata repository: Storing crucial metadata such as model lineage, versioning, aliasing, tagging, and annotations. Functions to store metadata are often included in data, code, and model registries [53].

- **Feature store:** Aiding feature engineering by storing and selecting suitable features, often including information about raw data transformations, aggregation, and management of feature values [54], [15].

Databases: Utilizing specialized databases for performancecritical applications.

- **Reasoning and semantic database:** Storing and interpreting relationships and meanings in data.
- **Timescale database:** Scaling queries for time-series data flexibly.
- **Caching and queuing database:** Enhancing performance and managing resource utilization.

C. Tactics

Tactics such as proven design strategies can improve various qualities [41]. We collected data on the quality aspects mentioned in the reviewed publications. These qualities can be categorized into data, models, and software. We could also extract tactics to address quality issues for most of the qualities. We summarized the qualities and constraints in Table 4. Attributes for which no reliable tactics were found in the literature are listed without naming suitable tactics. More detailed descriptions are provided in our report [31].

Table 4: Overview of quality attributes and constraints of the platforms.

Attribute	Tactics
Data Quality	
Accuracy	Avoid typos and duplicate entries
Completeness	
Consistent	Fill/remove missing entries, check for outliers
Currentness	Stream/Batch processing, update datasets frequently
Relevance	Collect data for the specific use case
Model Quality	
ML Performance	Monitor metrics for model evaluation
Explainability	
Software Quality	
Reliability	Monitoring, container management (dealing with errors in the distributed system)
Scalability	Distribution: containerization
Security	Encryption, software infrastructure updates
Performance	Latency: in-memory operations, model compression, reduce hardware requirements via model compression or frequency and number of calculations
Evolvability	Feature store, model repository
Maintainability	ML pipelines
Operability	Monitoring, retraining
Traceability	Storing and versioning of data, models, code, metadata, and their relationship
Constraints	
Cost	Costs can be reduced via software performance optimization
Development Time	Use automated setups
Model update time	ML pipelines, AutoML

Table 5: Platform categories.

Category	Description
Platform	ML supporting platforms and fully managed ML platforms.
Operating System	Entire operating system that is optimized for the operation and performance of ML.
Model-based	End-to-end platforms providing high-level scripting languages or graphical user interfaces.
Workbench	Offering managed programming environments with built-in integrations that help set up end-to-end MLOps production environments.
Package/IDE	Focused on the ML part, providing capabilities to put models into production.

V. Machine Learning Platforms

This section overviews infrastructure categories, licensing, and domain-specific platforms. In the initial search, we identified 70 MLOps infrastructures and categorized them as described in Table 5.

1) Licensing:: Commercial platforms such as *SageMaker*, *Azure MLOps*, *GCP*, and *valohai* are offered and managed by cloud providers. These platforms frequently cover the whole MLOps life-cycle and have AutoML tools embedded [15]. Models developed on the platforms of such cloud providers can be deployed on-demand using serverless technology, such as *Google Cloud Functions* and *Azure Functions*. However, commercial platforms often have the disadvantage of not being particularly customizable and open to technology variants. In addition, developers often struggle with vendor lock-in.

On the other hand, open-source licenses are free of charge and allow the software to be adapted and extended [19]. Further, developers who are concerned about putting their data in the cloud or being dependent on another company are implementing their own in-house MLOps platforms. Usually, they compose open-source platforms, tools, and libraries for that purpose, e.g., *MLFlow*, *Sacred*, and *DVC* (data version control) [55]. Tool overviews [56], [57], [34] can be used to find adequate options for such a compilation.

2) Domain-specific platforms:: Depending on the requirements, domain-specific platforms for container provisioning, automated ML, and big data can be useful. Many platforms use containerized workflows. Specialized container platforms for **container provisioning** platforms [58] are, e.g., *Docker*, *Kubernetes*, *OpenShift*, *AWS Elastic Container*, and *AWS Lambda*. Some platforms cover and **automate machine learning** development and deliver ready-to-production models [59], e.g., *Lobe AI*, *Google teachable machine*, and *Clarifai*. For **big data** processing, some of the reviewed platforms built up on platforms like *Databricks*, *Hadoop/Spark*, *Snowflake*, *Amazon Elastic Map Reduce* and *Google Big Query* [15].

VI. Discussion

Many developers use fully managed MLOps platforms provided by cloud service providers as those care for the entire infrastructure, servers, networks, security, and updates, allowing MLOps personnel to focus on their applications and data without worrying about the infrastructure. These platforms frequently offer services like virtualization, storage, and machine learning. Users can easily scale their resources up or down as needed but pay only for the resources they use [60].

In cases where companies have concerns about cost or about outsourcing data and using it in a cloud, in-house solutions are used to perform these tasks. This is also sometimes the case for addressing latency or bandwidth issues, e.g., in the case of Industry 4.0 scenarios [61]. With today's open-source libraries and tools, organizations can effectively compose platforms for their own purpose.

In our study, we identified features, tactics, design patterns, and reference architectures that are covered to varying degrees by the literature. In the existing literature, features for individual MLOps tools [25], [26] are described. Past literature, however, lacks features for MLOps platforms in specific. Therefore, our study summarizes MLOps **features** for platforms to fill that gap. Furthermore, we have extracted novel **tactics** to build such platforms. Not all quality attributes were addressed in the literature, so we could not provide a tactic for each of them, which leaves room for refinement in the future. **Design patterns** have been described by several sources [62] in the past. Moreover, reference architectures for MLOps [18], [27] and MLOps with specific requirements such as explainable AI and feedback in industrial use cases [63] are present in the existing literature. In summary, we provide an overview of the design space that can help in the design of MLOps platforms, but it does not cover all types of problems. Area-specific problems may occur that are not fully covered by this overview.

In our study, we found that the existing literature partially addresses challenges related to hardware and software optimization and containerization. Workflow management platforms grant scalability and help manage, scale, and deploy complex infrastructure. Searching the existing literature, we could not find any platform design concepts that reflect specific Machine Learning issues of large language models, e.g., encapsulating the multi-tier nature of fine-tuning aspects. We assume that a second tier is necessary for this purpose in addition to the training tier, which also requires high computing power and scalability at the user interface level. The first tier would be base model training with extensive datasets on high-performance clusters with GPU support. The second tier specializes the models for specific tasks and datasets. For example, scalable cloud instances could be accessed for webbased interfaces.

There is already some basic understanding and templates [64] on how to build MLOps platforms. These templates need adaptions for specific requirements. However, based on the existing literature, the question of how to build MLOps platforms is only partially answered from a software engineering perspective. With our analysis, we contribute to this knowledge and open up possibilities for future research. Future studies should extend this knowledge concerning integrating explainability and feedback into MLOps platforms and explore how to evaluate them.

VII. Conclusion

MLOps platforms frequently have a similar architecture [3]. This indicates that there are common qualities, features, and tactics relevant to their design. In this paper, we analyzed features of existing platforms as well as relevant design concepts like existing platforms and tools that can be used in the systematic creation of specialized MLOps platforms. We also discussed relevant qualities and potential tactics to achieve them.

We believe that the various design concepts that we identified are rather useful both for designers of innovative or customized MLOps platforms, as well as for potential users, who can use it as a basis to identify appropriate platform characteristics to select among potential alternative platform providers.

Acknowledgements

This work is supported by the project EXPLAIN, funded by the German Federal Ministry of Education under grant 01—S22030E. Any opinions expressed herein are solely by the authors and not the funding agency.

References

[1] M. Przybyla, "Data scientist vs machine learning ops engineer. here's the difference.", [Online]. Available: https://towardsdatascience.com

[2] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 2019, pp. 291–300.*

[3] V. Maya and A. Felipe, "The state of MLOps," 2021. [Online]. Available: https://repositorio.uniandes.edu.co/handle/1992/51495

[4] Valohai, "MLOps - machine learning operations." [Online]. Available: https://valohai.com/mlops/

[5] MLOps: What it is, why it matters, and how to implement it. [Online]. Available: https://neptune.ai/blog/mlops-what-it-is-why-itmatters-and-how-to-implement-it-from-a-data-scientist-perspective

[6] A. Hein, M. Schreieck, T. Riasanow, D. S. Setzke, M. Wiesche, M. Bohm, and H. Krcmar, "Digital platform ecosystems," ¨ *Electronic Markets*, vol. 30, no. 1, pp. 87–98, 2020.

[7] S. Oladele, "Best end-to-end MLOps platforms: Leading machine learning platforms that every data scientist need to know." [Online]. Available: https://neptune.ai/blog/end-to-end-mlops-platforms

[8] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young, "Machine learning: The high interest credit card of technical debt," in *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[9] J. George and A. Saha, "End-to-end machine learning using kubeflow," in *5th Joint International Conference on Data Science; Management of Data*. Association for Computing Machinery, 2022, pp. 336–338, event-place: Bangalore, India. [Online]. Available: https://doi.org/10.1145/3493700.3493768

[10] K. K. Gupt, M. A. Raja, A. Murphy, A. Youssef, and C. Ryan, "GELAB – the cutting edge of grammatical evolution," *IEEE Access*, vol. 10, pp. 38 694–38 708, 2022.

[11] Z. Sun, L. Sandoval, R. Crystal-Ornelas, S. M. Mousavi, J. Wang, C. Lin, N. Cristea, D. Tong, W. H. Carande, X. Ma, Y. Rao, J. A. Bednar, A. Tan, J. Wang, S. Purushotham, T. E. Gill, J. Chastang, D. Howard, B. Holt, C. Gangodagamage, P. Zhao, P. Rivas, Z. Chester, J. Orduz, and A. John, "A review of earth artificial intelligence," *Computers and Geosciences*, vol. 159, 2022.

[12] T. D. Akinosho, L. O. Oyedele, M. Bilal, A. Y. Barrera-Animas, A.-Q. Gbadamosi, and O. A. Olawale, "A scalable deep learning system for monitoring and forecasting pollutant concentration levels on UK highways," *Ecological Informatics*, vol. 69, 2022.

[13] D. De Silva and D. Alahakoon, "An artificial intelligence life cycle: From conception to production," *Patterns*, p. 100489, 2022.

[14] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas, "MLOps - definitions, tools and challenges," in *IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022, pp. 0453–0460.

[15] N. Gift and A. Deza, *Practical Mlops: Operationalizing Machine Learning Models*. O'Reilly Media, Inc, USA, 2021.

[16] S. Alla and S. K. Adari, *Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*, 1st ed. Apress, 2020.

[17] V. Lakshmanan, S. Robinson, and M. Munn, *Machine Learning Design Patterns: Solutions to Common Challenges in Data Preparation, Model Building, and MLOps*. O'Reilly UK Ltd., 2020.

[18] D. Kreuzberger, N. Kuhl, and S. Hirschl, "Machine learning operations ¨ (MLOps): Overview, definition, and architecture," *arXiv:2205.02302 [cs]*, 2022.

[19] S. Choudhary, "Kubernetes-based architecture for an onpremises machine learning platform," 2021. [Online]. Available: https://aaltodoc.aalto.fi:443/handle/123456789/110516

[20] A. B. Kolltveit and J. Li, "Operationalizing machine learning models: a systematic literature review," in *SE4RAI '22: Proceedings of the 1st Workshop on Software Engineering for Responsible AI*. Association for Computing Machinery, 2022, pp. 1–8.

[21] A. Lima, L. Monteiro, and A. P. Furtado, "Mlops: Practices, maturity models, roles, tools, and challenges-a systematic literature review." *ICEIS (1)*, pp. 308–320, 2022.

[22] Y. Zhou, Y. Yu, and B. Ding, "Towards MLOps: A Case Study of ML Pipeline Platform," in *International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. IEEE, 2020, pp. 23–25.

[23] R. Min˜on, J. Diaz-de Arcaya, A. I. Torre-Bastida, and P. Hartlieb, ´ "Pangea: An MLOps Tool for Automatically Generating Infrastructure and Deploying Analytic Pipelines in Edge, Fog and Cloud Layers," *Sensors*, vol. 22, no. 12, p. 4425, 2022.

[24] A. H. Nia, F. J. Kaleibar, F. Feizi, F. Rahimi, and H. Kashfi, "Unlocking the Power of Data in Telecom: Building an Effective MLOps Infrastructure for Model Deployment," in *2023 7th Iranian Conference on Advances in Enterprise Architecture (ICAEA)*. IEEE, 2023, pp. 15–16.

[25] G. Zarate, R. Mi ´ n˜on, J. D ´ ´ıaz-de Arcaya, and A. I. Torre-Bastida, "K2e: Building mlops environments for governing data and models catalogues while tracking versions," in *IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2022, pp. 206– 209.

[26] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. Di Nucci, F. Palomba, and D. A. Tamburri, "A Multivocal Literature Review of MLOps Tools and Features," in *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE, 2022, pp. 2022–02.

[27] I. Kumara, R. Arts, D. Di Nucci, W. J. Van Den Heuvel, and D. A. Tamburri, "Requirements and Reference Architecture for MLOps:Insights from Industry," *Authorea Preprints*, 2023.

[28] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Gueh ´ eneuc, "Studying ´ software engineering patterns for designing machine learning systems," in *2019 10th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2019, pp. 49–495.

[29] R. Subramanya, P. Rais ¨ anen, S. Sierla, and V. Vyatkin, "Cloud comput-¨ ing design patterns for mlops: applications to virtual power plants," in *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2023, pp. 01–07.

[30] B. Kitchenham, "Procedures for performing systematic reviews," *Keele, UK, Keele University*, vol. 33, pp. 1–26, 2004.

[31] L. Faubel and K. Schmid, "An mlops platform comparison," *Hildesheimer Informatik Berichte*, no. 1/2024, SSE 1/24/E, 2024.

[32] E. Raj, *Engineering MLOps: Rapidly build, test, and manage productionready machine learning life cycles at scale*. Packt Publishing, 2021.

[33] A. Choudhury, "Top 8 alternatives to apache spark." [Online]. Available: https://analyticsindiamag.com/top-8-alternatives-to-apache-spark/

[34] Y. Gavrilova, "The best open-source MLOps tools you should

[35] D. Meedeniya and H. Thennakoon, "Impact factors and best practices to improve effort estimation strategies and practices in DevOps," in

The 11th International Conference on Information Communication and Management. Association for Computing Machinery, 2021, pp. 11–17.
[36] S. Rahman and E. Kandogan, "Characterizing practices, limitations, and opportunities related to text information extraction workflows: A human-in-the-loop perspective," in *CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3491102.3502068
[37] S. Garg, P. Pundir, G. Rathee, P. Gupta, S. Garg, and S. Ahlawat, "On continuous integration / continuous delivery for automated deployment of machine learning models using MLOps," in *2021 IEEE Fourth International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2021, pp. 25–28.
[38] W. Wu and C. Zhang, "Towards understanding end-to-end learning in the context of data: Machine learning dancing over semirings codd's table," in *Proceedings of the Fifth Workshop on Data Management for End-To-End Machine Learning*. Association for Computing Machinery, 2021.
[39] H. Jayalath and L. Ramaswamy, "Enhancing performance of operationalized machine learning models by analyzing user feedback," in *4th International Conference on Image, Video and Signal Processing*. Association for Computing Machinery, 2022, pp. 197–203.
[40] A. Serban, K. van der Blom, H. Hoos, and J. Visser, "Adoption and effects of software engineering best practices in machine learning," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. Association for Computing Machinery, 2020.
[41] H. Cervantes and R. Kazman, *Designing Software Architectures: A Practical Approach*. Boston, MA, USA: Addison-Wesley Professional, 2016.
[42] S. Giannakopoulou, M. Karpathiotakis, B. Gaidioz, and A. Ailamaki, "CleanM: An optimizable query language for unified scale-out data cleaning," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1466–1477, 2022.
[43] C. Poss, T. Irrenhauser, M. Prueglmeier, D. Goehring, F. Zoghlami, V. Salehi, and O. Ibragimov, "Enabling robot selective trained deep neural networks for object detection through intelligent infrastructure," in *Proceedings of the 4th International Conference on Automation, Control and Robotics Engineering*. Association for Computing Machinery, 2019.
[44] N. O. Nikitin, P. Vychuzhanin, M. Sarafanov, I. S. Polonskaia, I. Revin, I. V. Barabanova, G. Maximov, A. V. Kalyuzhnaya, and A. Boukhanovsky, "Automated evolutionary approach for the design of composite machine learning pipelines," *Future Gener. Comput. Syst.*, vol. 127, pp. 109–125, 2022, place: NLD Publisher: Elsevier Science Publishers B. V.
[45] E. Raj, D. Buffoni, M. Westerlund, and K. Ahola, "Edge MLOps: An automation framework for AIoT applications," in *2021 IEEE International Conference on Cloud Engineering (IC2E)*, 2021, pp. 191–200.
[46] H. Liu, Q. Gao, J. Li, X. Liao, H. Xiong, G. Chen, W. Wang, G. Yang, Z. Zha, D. Dong, D. Dou, and H. Xiong, "JIZHI: A fast and costeffective model-as-a-service system for web-scale online inference at baidu," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery ; Data Mining*. Association for Computing Machinery, 2021, pp. 3289–3298.
[47] A. Bose and A. Aggarwal, "MLOps – "Why is it required?" and "What it is"? - KDnuggets," 2022. [Online]. Available: https://www.kdnuggets.com/2020/12/mlops-why-required-what-is.html
[48] Z. Li, X.-Y. Liu, J. Zheng, Z. Wang, A. Walid, and J. Guo, "FinRLpodracer: high performance and scalable deep reinforcement learning for quantitative finance," in *Proceedings of the Second ACM International Conference on AI in Finance*. Association for Computing Machinery, 2021, pp. 1–9.
[49] Z. Azad, R. Sen, K. Park, and A. Joshi, "Hardware acceleration for DBMS machine learning scoring: Is it worth the overheads?" in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2021, pp. 243–253.
[50] O. E. Oluyisola, S. Bhalla, F. Sgarbossa, and J. O. Strandhagen, "Designing and developing smart production planning and control systems in the industry 4.0 era: A methodology and case study," *J. Intell. Manuf.*, vol. 33, no. 1, pp. 311–332, 2022.
[51] M. Oplenskedal, P. Herrmann, and A. Taherkordi, "DeepMatch2: A comprehensive deep learning-based approach for in-vehicle presence detection," *Information Systems*, vol. 108, p. 101927, 2022.
[52] J. N. van Rijn, B. Bischl, L. Torgo, B. Gao, V. Umaashankar, S. Fischer, P. Winter, B. Wiswedel, M. R. Berthold, and J. Vanschoren, "OpenML: A collaborative science platform," in *Advanced Information Systems Engineering*. Springer Berlin Heidelberg, 2013, vol. 7908, pp. 645–649.
[53] B. Brik, K. Boutiba, and A. Ksentini, "Deep learning for b5g open radio access network: Evolution, survey, case studies, and challenges," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 228–250, 2022.
[54] M. van der Goes, "Scaling enterprise recommender systems for decentralization," in *Fifteenth ACM Conference on Recommender Systems*. Association for Computing Machinery, 2021, pp. 592–594.
[55] D. L. Visengeriyeva, A. Kammer, I. Bar, A. Kniesz, and M. Pl ¨ od, ¨ "ml-ops.org." [Online]. Available: https://ml-ops.org/
[56] "LF AI & Data Landscape," 2024, [Online; accessed 5. Apr. 2024]. [Online]. Available: https://landscape.lfai.foundation
[57] K. S. d. Prado, "Awesome MLOps," original-date: 2020-05-25T22:53:26Z. [Online]. Available: https://github.com/kelvins/awesomemlops
[58] D. Muiruri, L. E. Lwakatare, J. K. Nurminen, and T. Mikkonen, "Practices and infrastructures for ML systems – an interview study," 2021, publisher: TechRxiv.
[59] L. Silva and F. Osorio, "Flowi: A platform for ML development and management," 2021, [Online; accessed 14. May 2024]. [Online]. Available: https://proceedings.science/wmecai/wmecai-2021/papers/flowi-aplatform-for-ml-development-and-management?lang=en
[60] L. Cardoso Silva, F. Rezende Zagatti, B. Silva Sette, L. Nildaimon dos Santos Silva, D. Lucredio, D. Furtado Silva, and H. de Medeiros Caseli, ´ "Benchmarking machine learning solutions in production," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 626–633.
[61] L. Faubel, K. Schmid, and H. Eichelberger, "MLOps Challenges in Industry 4.0," *SN Comput. Sci.*, vol. 4, no. 6, pp. 828–11, Oct. 2023.
[62] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Gueh ´ eneuc, "Machine ´ learning architecture and design patterns," *IEEE Software*, vol. 8, p. 2020, 2020.
[63] L. Faubel, T. Woudsma, L. Methnani, A. G. Ghezeljhemeidan, F. Buelow, K. Schmid, W. D. van Driel, B. Kloepper, A. Theodorou, M. Nosratinia, and M. Bang, "Towards an mlops architecture for xai in ˚ industrial applications," 2023.
[64] Valohai, "Valohai | Take ML places it's never been," Feb. 2023, [Online accessed 23. Feb. 2023]. Available: https://valohai.com