SkeltyMLOps: Orchestrating Collaborative MLOps Activities

Charbel Daoud[1], Danielle Azar[2], Julie Boiché[1], Christelle Urtado[1] and Sylvain Vauttier[1]

[1]*EuroMov Digital Health in Motion, Univ. Montpellier and IMT Mines Ales, Ales, France*
[2]*Department of Computer Science & Mathematics, Lebanese American University, Byblos, Lebanon*

Abstract

With the increasing adoption of machine learning (ML), the need to manage, in a tailored manner, the complete lifecycle of this new type of software arises. MLOps (Machine Learning Operations) extends DevOps to address the specific challenges of development and lifelong management of ML-powered software. Despite the increasing attention to this domain, practitioners still lack an accessible, modular architecture to support MLOps projects. To address this lack, this paper proposes SkeltyMLOps, a reference architecture designed to promote collaboration between the diverse actors involved in MLOps processes. The architecture is derived from a thorough literature review, from which we extracted and clustered, using a Large Language Model (LLM), a comprehensive list of MLOps actors and activities. These clusters were then used to guide the architectural decomposition and component design of SkeltyMLOps. The originality of our proposed reference architecture is (i) that it clearly aligns its components with the responsibilities of MLOps actors, covering the identified dimensions of MLOps processes, and (ii) mediates collaboration by orchestrating interactions between the different types of actors. This architecture provides a foundation for building collaborative ML-powered software.

Keywords

Software Engineering, Machine Learning Operations, MLOps Actors, MLOps Activities, MLOps Reference Architecture, Collaborative MLOps, Orchestration

1. Introduction

The advent of machine learning (ML) has led to the adaptation of DevOps practices to manage ML-powered software, resulting in the rise of Machine Learning Operations (MLOps) which is heavily inspired by DevOps [1, 2]. DevOps, a set of established practices within modern software engineering, integrates development and operational practices to streamline and shorten software release cycles through systematic build and deployment processes heavily dependent on automation [3]. Specifically, the `Dev` segment in DevOps consists of the Plan, Code, Build, and Test phases, while the `Ops` segment consists of the Release, Deploy, Operate, and Monitor phases [4]. As software systems evolve, whether for corrective or evolutionary maintenance, DevOps emphasizes iterative feedback loops that connect the Monitor phase back to the Plan phase. MLOps emerged around 2015, significantly influenced by the foundational work of Sculley *et al.* [5]. Consequently, the software engineering community is increasingly addressing ML-specific challenges through MLOps practices. Nevertheless, despite this increased awareness and adoption, MLOps practitioners continue to face numerous unaddressed challenges. First, Kolar *et al.* [6] note the absence of a consolidated MLOps architecture and the complexity of coordinating multiple technological components. Second, black-box tools with limited modularity hinder adoption by organizations [7, 8]. A preliminary investigation of open-source contributions on GitHub confirms this adoption barrier, revealing few accessible frameworks or reference architectures for practitioners[9]. Third, collaboration is a critical challenge that affects the success of ML-powered software which typically relies on multidisciplinary team structures [10]. However, many MLOps implementations currently lack the tools and practices needed to effectively

MLOps25: Workshop on Machine Learning Operations. October 25, 2025, Bologna, Italy
charbel.daoud@mines-ales.fr (C. Daoud); danielle.azar@lau.edu.lb (D. Azar); julie.boiche@umontpellier.fr (J. Boiché); christelle.urtado@mines-ales.fr (C. Urtado); sylvain.vauttier@mines-ales.fr (S. Vauttier)
0009-0009-6364-638X (C. Daoud); 0000-0002-6159-3714 (D. Azar); 0000-0002-1676-0528 (J. Boiché); 0000-0002-6711-8455 (C. Urtado); 0000-0002-5812-1230 (S. Vauttier)

facilitate such collaboration. Uysal *et al.* [11] emphasize the need for a tailored and holistic project management framework for ML-powered software. Finally, the engineering rigor in ML-powered software is still maturing, especially in eliciting requirements, which often depends on project-specific practices and therefore varies in consistency [12]. This paper proposes SkeltyMLOps, a reference architecture intended to facilitate the practical adoption of MLOps practices. SkeltyMLOps is structured around a set of well-scoped actors, each defined by their responsibilities and interactions, and focuses on orchestrating the core MLOps activities reported in the literature. The approach for designing SkeltyMLOps includes three main steps: (1) identifying, analyzing, and mapping the different actors and dimensions of MLOps; (2) identifying the set of activities that constitute an MLOps process; and (3) clustering these activities into software components to create a reference architecture. In summary, the contributions of this work are twofold: (a) identification and formalization of the key actors and activities involved in MLOps; and (b) introduction of SkeltyMLOps, a reference architecture designed to support the orchestration of these activities and to strengthen the collaboration between actors.

The remainder of this paper is structured as follows. Section 2 outlines the research questions that guide this work and the methodology used to address them. Section 3 identifies the main MLOps actors and dimensions, mapping and naming them through a systematic literature review and semantic clustering via a Large Language Model. Section 4 examines the activities performed by each actor and the interactions among them within these dimensions. Section 5 discusses threats to the validity and reliability of the results. Section 6 presents SkeltyMLOps, a reference architecture that orchestrates these activities and promotes collaboration among actors. Section 7 positions SkeltyMLOps within the existing literature, and highlights its novel contributions. Finally, Section 8 describes directions for future work.

2. Research Questions & Methodology

This section presents the research questions (RQs) that guide this study, describes the methodology adopted to address them, and explains the rationale for using a Large Language Model (LLM) within this research.

2.1. Research Questions

This research aims to identify the dimensions, actors, and activities of MLOps and to extract the fundamental components necessary for designing a reference architecture. The formulated RQs that refine and guide our study are listed below:

- **RQ1:** Who are the main actors involved in MLOps?
- **RQ2:** What are the dimensions of MLOps?
  A dimension is a distinct functional area that groups together related actors and activities.
- **RQ3:** What are the different activities that constitute MLOps processes?
- **RQ4:** How should an MLOps architecture that effectively integrates the activities identified in RQ3 be designed to facilitate collaboration among the actors identified in RQ1?

2.2. Methodology for RQ1 and RQ3

To address RQ1 and RQ3, a systematic literature review (SLR) is conducted.

Search Strategy. The literature search was carried out via Google Scholar. Using Google Scholar reduces bias towards specific publishers as suggested by prior comparisons [13, 14]. The search query was initially formulated from key terms appearing in the RQs, including *MLOps*, *actors*,

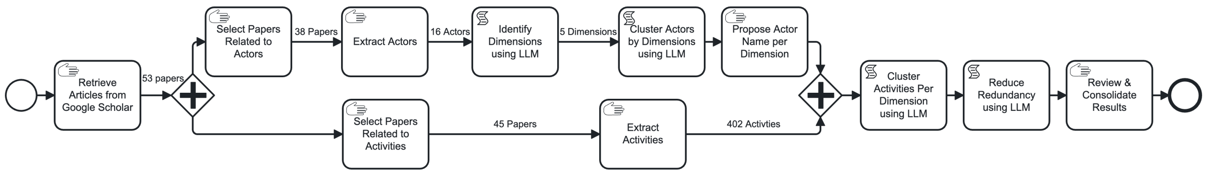
Retrieve Articles from Google Scholar
53 papers
Select Papers Related to Actors
38 Papers
Extract Actors
16 Actors
Identify Dimensions using LLM
5 Dimensions
Cluster Actors by Dimensions using LLM
Propose Actor Name per Dimension
Select Papers Related to Activities
45 Papers
Extract Activities
402 Activties
Cluster Activities Per Dimension using LLM
Reduce Redundancy using LLM
Review & Consolidate Results

Figure 1: Overview of the systematic literature review process.

activities, *processes*, and *architecture*. To enhance coverage and precision, the query was extended with relevant synonyms and semantically related terms. The final version of the search string is: `allintitle:(MLOps OR "machine learning operations") AND (actor OR actors OR pipeline OR pipelines OR architecture OR architectures OR architecting OR activity OR activities OR workflow OR workflows OR process OR processes).`

Selection Criteria. To ensure consistency and reliability in the selection process, two authors independently reviewed a random sample of 10 papers. A consensus meeting was held to align their interpretations, resolve ambiguities, and finalize the inclusion and exclusion criteria. It was decided that a paper will be included if it explicitly presents at least one of the following: (i) a list of MLOps actors, (ii) a figure describing MLOps processes, or (ii) an enumeration of related activities. A paper will be excluded if it belongs to: (a) non-English publications, (b) citation-only entries (*e.g.*, bibliographic listings without content), (c) unpublished manuscripts, (d) inaccessible documents (*e.g.*, full-text unavailable online or behind paywalls without institutional access), and (e) papers retrieved from subscription-only sources not accessible to the authors at the time of review.

Execution. The Google Scholar search initially[1] retrieved 51 entries. After applying the inclusion and exclusion criteria, the dataset was reduced to 35 primary studies. A random sample of 5 papers was selected from this set to refine the data extraction strategy. Data extraction was conducted independently by two authors and finalized through a consensus meeting. For RQ1, the set of actors, explicitly mentioned in each paper, were extracted. For RQ3, the described activities, either from process diagrams or enumerated descriptions in the text, were extracted. Following this pilot phase, the remaining papers were each assigned to both authors for independent extraction. The extracted data were then validated through bilateral discussions to reach consensus.

Snowballing. To complement the initial title-based search and address its limitations, backward and forward snowballing techniques were applied. Backward snowballing involves reviewing the reference lists of the initially selected papers to identify additional relevant studies. Forward snowballing, in turn, identifies studies that cite those papers. Applying the same inclusion and exclusion criteria, this process yielded 18 additional papers, resulting in a final set of 53 selected publications. Relevant data to RQ1 and RQ3 were extracted using the same extraction procedures detailed earlier.

Data Synthesis. Out of the 53 retained studies, 38 contribute to answering RQ1 and 45 to RQ3, with 30 papers addressing both. Figure 1 illustrates the review workflow. A data extraction sheet [15] is used to systematically collect information from the selected studies. The extracted data consists of: (i) study metadata such as title, authors, and citation; (ii) for RQ1, a list of explicitly mentioned actors; and (iii) for RQ3, a list of identified activities, whether extracted from described processes or explicitly enumerated in the text. The subsequent sections describe how this data was analyzed and mapped to RQ1 and RQ3.

[1]The search was executed in March 2025.

2.3. An LLM-powered Systematic Literature Review

The experiments in this study employ an LLM to systematically cluster and label data extracted from the systematic literature review. This approach is motivated by recent findings showing that LLMs can enhance unsupervised clustering by imposing semantic structure on unlabeled data [16, 17]. Zhang *et al.* [18] propose ClusterLLM, a framework that leverages ChatGPT's semantic capabilities to improve clustering outcomes, while maintaining low computational cost. Furthermore, LLMs have been shown to serve as efficient annotators, capable of producing meaningful semantic labels that approximate or replace human annotations [19, 20]. Following a comparative evaluation of multiple LLMs[2], we selected OpenAI's o3 model[3] based on its superior performance in terms of reasoning quality, completeness (*e.g.*, no missed entries during clustering), and labeling consistency [21, 22]. The prompts used throughout the clustering process were iteratively refined according to established prompt engineering practices [23, 24].

3. MLOps Actors & Dimensions

This section jointly addresses RQ1 and RQ2 by identifying actors and dimensions involved in MLOps. The analysis follows an iterative process in which the dimensions are inferred from the initially identified actors. These dimensions are then used to re-examine the actors, allowing for the refinement of the actor set. This process ensures that both RQs cross-validate each other.

3.1. MLOps Actors

The actors identified when addressing RQ1 (Section 2.2) are analyzed. Table 1 summarizes actors referenced in multiple sources along with their corresponding references. Actors mentioned only in a single study are excluded from the table (such as Data Provider [38] or ML Owner [56]), as their limited appearance suggests they are not widely recognized. During the extraction process, inconsistencies in how actors and their responsibilities were defined across studies were noted. For example, while most papers assign `Data Scientists` to data-centric tasks, others also associate them with model development responsibilities. Similarly, the role of `Software Engineers` varies significantly: some sources limit their involvement in software development, whereas others include operational and ML tasks. To systematically reduce such ambiguities and identify actor groups, we apply a clustering approach based on the LLM selected in Section 2.3. This approach is described in the next subsection.

3.2. MLOps Dimensions

To address the ambiguities stemming from unclear actor responsibilities, dimensions are formally identified and used to structure actors. We define an MLOps dimension as a conceptual axis that characterizes a specific concern in MLOps. Together, these dimensions form a conceptual space in which actors can be situated based on their responsibilities along each axis. This representation facilitates a structured understanding of collaboration across MLOps. This process is carried out using the o3-model through a three-step process: (1) identify relevant dimensions from a predefined actor set, (2) cluster actors according to these dimensions, and (3) propose representative actor name for each cluster. The exact prompt provided to the o3 model is included below:

You're an expert in machine learning operations (MLOps). You will find below a list that contains 16 Actors who appear in different activities in an MLOps process. I want you to:

- Identify dimensions from the provided actors.
- Cluster actors according to these dimensions.
- Propose a leading actor for each dimension.
- Format the final output as a structured table with three columns: Dimension, List of Actors, Proposed Leading Actor.

[2]See the replication package for full details of the LLM comparison.
[3]OpenAI's o3 model (see link: https://openai.com/index/introducing-o3-and-o4-mini/)

Table 1
Mentioned actors in reviewed papers.

Actor	Citations	Count
Business Stakeholder	[1, 38, 63]	3
Solution Architect	[1, 63]	2
Domain Expert	[32, 38, 45, 55, 59]	5
Manager	[38, 45]	2
Data Scientist	[1, 29, 31, 32, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 52, 53, 54, 55, 56, 57, 60, 61, 62, 63, 64]	31
Data Engineer	[1, 31, 32, 33, 36, 38, 39, 40, 43, 45, 47, 49, 50, 55, 57, 62, 63]	17
Data Steward	[43, 55]	2
ML Engineer / Developer	[60, 31, 34, 35, 40, 41, 42, 51, 56]	9
Scientist	[38, 58]	2
Software Developer	[35, 51, 50, 52]	4
Software Engineer	[1, 32, 38, 50, 53, 55, 56, 63]	8
Application Developer	[45, 46, 59]	3
DevOps Engineer	[1, 31, 34, 38, 55, 56, 57, 63]	8
MLOps Engineer	[1, 38, 43, 53, 55, 63]	6
Operations Engineer	[32, 36, 39, 41, 48, 55, 56]	7
IT Operations / Professional	[34, 50, 57]	3

Table 2
Proposed dimensions and actor clusters derived from the LLM.

Dimension Name	Actor Cluster	Leading Actor
Plan	Business Stakeholder, Manager, Domain Expert	Product Owner
Data Engineering	Data Scientist, Data Engineer, Data Steward, Scientist	Data Engineer
Model Engineering	ML Engineer / Developer	Model Engineer
Software Engineering	Solution Architect, Software Engineer, Software Developer, Application Developer	Software Engineer
Operations	DevOps Engineer, MLOps Engineer, Operations Engineer, IT Operations / Professional	Operations Engineer

A manual review is conducted on the raw output generated by the prompt. One adjustment involves renaming the first dimension from Business & Domain to Plan, to align it with the standard DevOps phases (Plan, Code, Build, etc.). The actor clusters were also reviewed and updated: the leading actor in the Plan dimension was changed from Business Lead to Product Owner to reflect the inclusion of software engineering activities alongside business concerns. Importantly, the Product Owner actively collaborates, within this dimension, with all other identified actors ensuring alignment and collaboration during the planning phase. All manual modifications are documented in the replication package [15]. The final adjusted set of dimensions and leading actors is presented in Table 2.

Highlights (RQ1 & RQ2). In response to RQ1, we identify 5 actors that lead MLOps dimensions: Product Owner, Data Engineer, Model Engineer, Software Engineer, and Operations Engineer. Addressing RQ2, we propose 5 MLOps dimensions: Plan, Data Engineering (Data), Model Engineering (Model), Software Engineering (Soft), and Operations (Ops).

4. MLOps Activities

This section further addresses RQ3, investigating how MLOps activities can be systematically identified and structured. Given that MLOps has no widely accepted standardized set of activities, a six-step methodology is proposed to fill this gap:

- **Activity Extraction.** The extraction process, described in Section 2.2, produces a matrix with 402 activities (rows) across 45 publications (columns).

- **Grouping by Dimensions.** The 402 recorded activities are then clustered according to the 5 dimensions introduced in Section 3.2 (Plan, Data, Model, Soft, and Ops). To automate this mapping, the o3 model is reused. The complete list of activities is fed into the model, leveraging its language understanding capabilities to classify each activity under one of the 5 dimensions. The prompt used is:

You're an expert in machine learning operations (MLOps). You will find, at the end, a list that contains 402 Activities that appear in different MLOps processes. I want you to:

- Categorize these Activities into these 5 dimensions: Plan, Data engineering, Model engineering, Software engineering, and Operations.
- Ensure activities are not mistakenly categorized: *e.g.*, Differentiate what's Software and what's Operations.
- Format the final output as a structured table with two columns: Dimension, Activity. I should get 402 rows (other than the header)

- **Dimension-Specific Clustering.** After executing the prompt, the o3 model produces clusters for the Plan (26 activities), Data (101 activities), Model (113 activities), Soft (59 activities), and Ops (103 activities) dimensions.

- **Redundancy Reduction.** To refine the activity sets extracted per dimension, a dedicated prompt is applied to each one. This step aims to: (1) group semantically similar activities and eliminate redundancies, and (2) assign a representative label and generate a concise description for each resulting cluster. The procedure is repeated independently for each of the 5 dimensions. The prompt used in this step is as follows:

You're an expert in machine learning operations (MLOps). You will find, at the end, a list of activities related to <dimension_name> Dimension. You are asked to:

- Group similar activities and propose new activity name. The proposed activity name should start with a verb. Use concise verb phrase, for example I prefer "Review Code" instead of "perform code review".
- Ensure activities are not mistakenly merged if they represent distinct tasks
- Format the final output as a structured table with three columns: Proposed Activity Name, Grouped Activities, A brief explanation. Order the activities in the table in a chronological order.

This step reduces the total number of distinct activities in Plan, Data, Model, Soft, and Ops dimensions to 7, 8, 9, 5, and 9, respectively (38 activities in total).

- **Manual Review and Merging.** A manual review of these 38 activities is conducted to refine the activity set and reduce redundancy[4]. For example, Diagram Deployment Model was merged with Design Architecture.

- **Final Consolidation.** The consolidated list of activities, each with a description and frequency across the 45 analyzed publications, is included in the replication package. A visual summary of these activities is provided in Figure 2, which organizes them according to the 5 proposed MLOps dimensions. Each dimension is color-coded and annotated with the lead actors responsible for executing their associated activities.

Highlights (RQ3). In response to RQ3, 38 activities are identified, distributed across the five MLOps dimensions. These activities constitute the foundation of SkeltyMLOps 's architectural design.

5. Threats to Validity

Despite our methodical approach and the rigor with which the extraction, processing, and analysis of data were handled, there are still potential threats to validity regarding the obtained results.

External validity. Our study is based on a selection of academic research papers. This may omit relevant alternative sources, such as white papers and online articles published by practitioners. However, it is a deliberate

[4]The details of this manual refinement are provided in the replication package: https://anonymous.4open.science/r/mlops_ecai25-EB68/

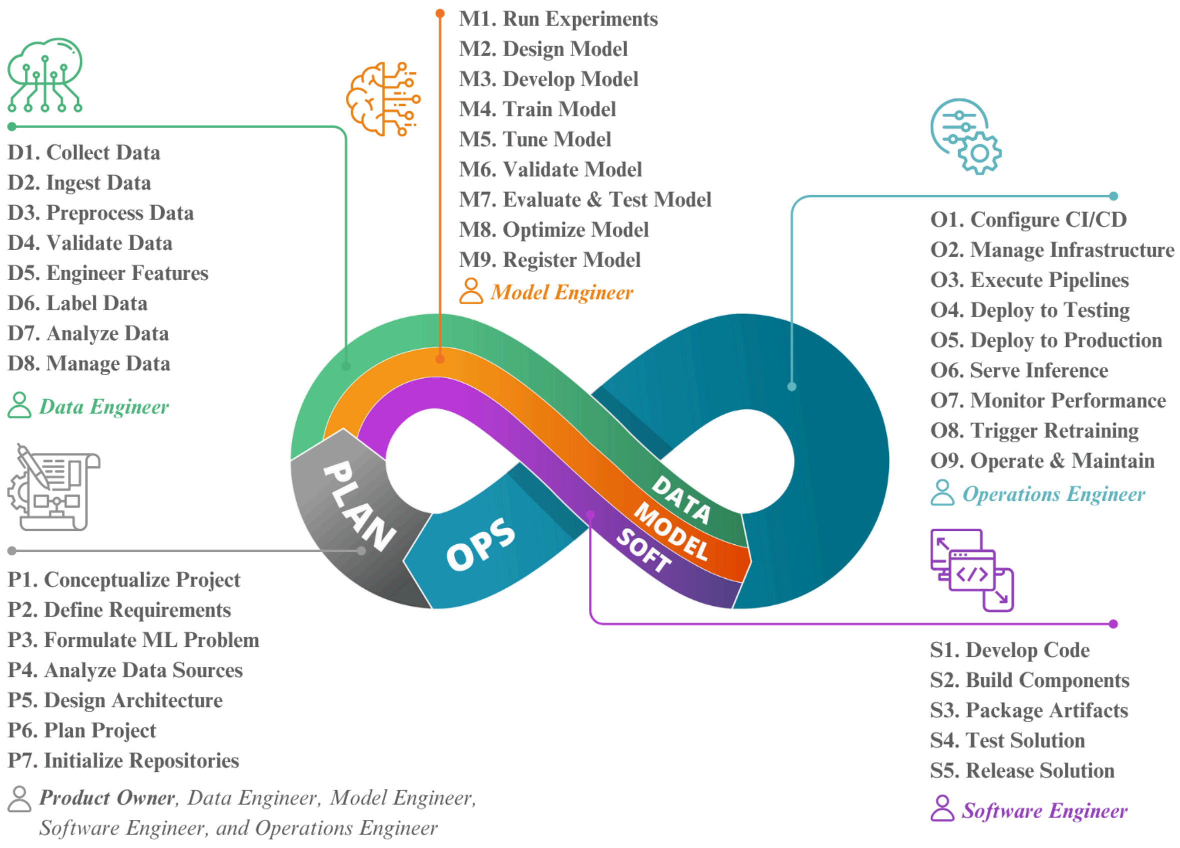

Figure 2: MLOps infinite loop showing dimensions, actors and associated activities.

design choice to only retain peer-reviewed papers, corresponding to established practices. Moreover, these papers are retrieved from a single search engine. We mitigate this threat by relying on Google Scholar, which is reputed for its agnostic and wide-ranging indexing of publications, and by applying backward and forward snowballing techniques to cover as many relevant studies as possible.

Internal validity. The proposed lists of actors, dimensions, and activities may reflect biases introduced by our backgrounds in software engineering, machine learning, and operations. To reduce these biases, we used an LLM to group the extracted information. While this approach minimizes subjective manual interpretations, it introduces new risks inherent to LLMs, such as hallucinations. We partially mitigate this by manually reviewing and refining the model's output to ensure coherence and accuracy.

Reliability. Our results depend on the choice of the LLM and are influenced by the design of the prompts. This is especially true for the activity clustering task, which involves a large and diverse set of items. LLM responses can vary across runs or model versions and may lack reproducibility if prompt designs are not well-documented. Furthermore, since LLMs are trained on vast and heterogeneous data, they might introduce contextually irrelevant groupings. To address these issues, we iteratively refined the prompts and ensured that outputs were reproducible under stable conditions. Our final manual adjustments were minimal and limited to coherence and clarity, not conceptual alterations.

6. A Reference MLOps Architecture Promoting Collaboration

This section introduces SkeltyMLOps, a reference architecture for MLOps, that explicitly supports the actors and activities identified in Sections 3 and 4. A Reference Architecture is a reusable blueprint that encapsulates essential design principles, best practices, and structural guidelines for a given domain [25, 26]. It serves as a foundation for developing concrete MLOps frameworks. SkeltyMLOps is a reference architecture for MLOps that integrates relevant domain-specific knowledge, and standardizes activities within the MLOps process. The motivation behind SkeltyMLOps is to emphasize and orchestrate collaboration among the various actors involved.

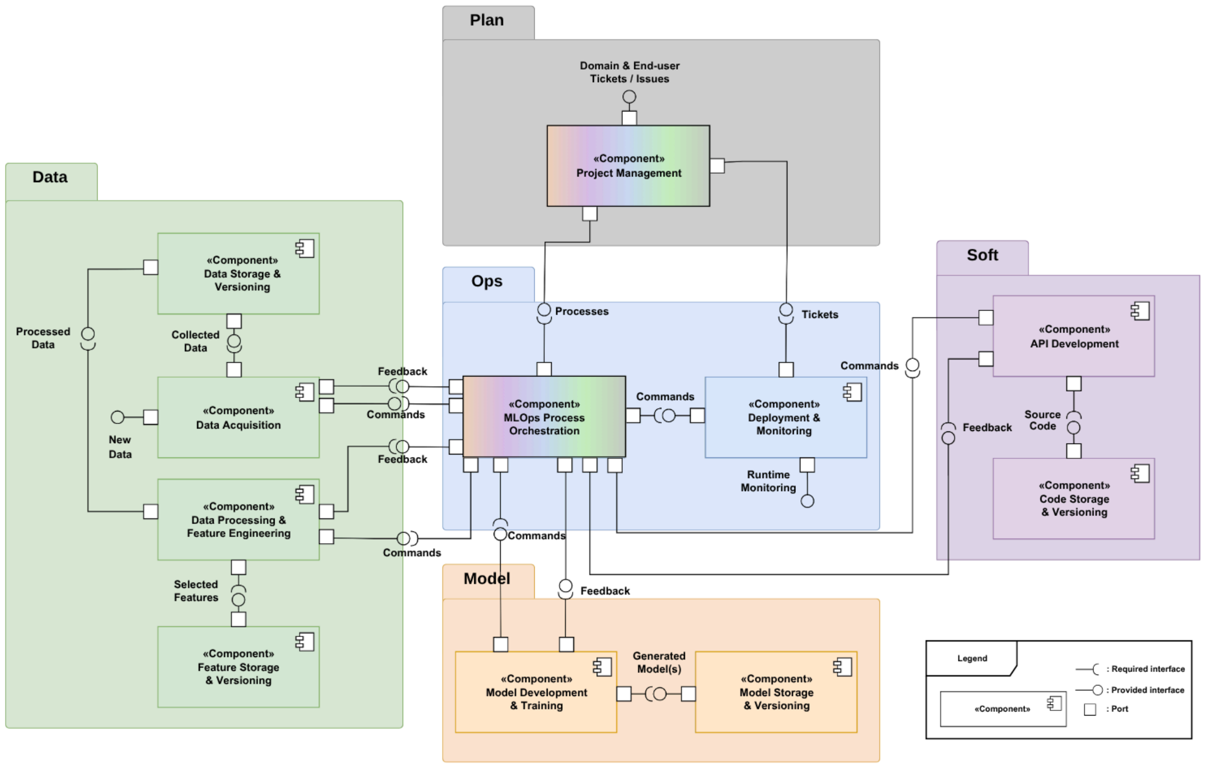
Plan
Domain & End-user Tickets / Issues
«Component» Project Management
Data
«Component» Data Storage & Versioning
Processed Data
Collected Data
«Component» Data Acquisition
New Data
«Component» Data Processing & Feature Engineering
Selected Features
«Component» Feature Storage & Versioning
Ops
Processes
Tickets
Feedback
Commands
Feedback
«Component» MLOps Process Orchestration
Commands
«Component» Deployment & Monitoring
Runtime Monitoring
Commands
Commands
Soft
«Component» API Development
Source Code
Feedback
«Component» Code Storage & Versioning
Model
Feedback
«Component» Model Development & Training
Generated Model(s)
«Component» Model Storage & Versioning
Legend
«Component»
: Required interface
: Provided interface
: Port

Figure 3: UML component diagram of SkeltyMLOps.

Table 3
Mapping between SkeltyMLOps components and activities.

Component	**Activities**
Project Management	Conceptualize Project, Define Requirements, Formulate ML Problem, Analyze Data Sources, Design Architecture, Plan Project, Initialize Repositories
Data Acquisition	Collect Data, Ingest Data
Data Processing & Feature Engineering	Preprocess Data, Validate Data, Engineer Features, Label Data, Analyze Data, Manage Data
Model Development & Training	Run Experiments, Design Model, Develop Model, Train Model, Tune Model, Validate Model, Evaluate & Test Model, Optimize Model, Register Model
API Development	Develop Code, Build Components, Package Artifacts, Test Solution, Release Solution
Deployment & Monitoring	Configure CI/CD, Manage Infrastructure, Deploy to Testing, Deploy to Production, Serve Inference, Monitor Performance, Operate & Maintain
MLOps Process Orchestrator	Execute Pipelines, Trigger Retraining

6.1. From Actors & Dimensions to an MLOps Reference Architecture

To design the architecture of SkeltyMLOps, a modular decomposition is adopted, aligned with MLOps identified dimensions. Each module corresponds to one of the 5 MLOps dimensions and is structured to serve the needs of its actors. This design follows a domain-driven approach, where each module represents a distinct sub-domain of MLOps. Within each module, finer-grained components are defined to support the execution of specific activities identified in Section 4. The overall architecture, with its modules and their internal components, is presented in Figure 3, while Table 3 details the correspondence between components and the activities they support. At the core of the architecture is the MLOps Process Orchestration (MPO) component. It manages both control and data flows among all components, and ensures coordinated execution of the MLOps process. The Project Management component, which handles requirements engineering and planning tasks, supplies the MPO with issue tickets [27, 28]. These tickets are then resolved automatically via a translation into executable commands that involve the appropriate components. A key aspect of SkeltyMLOps is that it versions not only code, but data and ML models as well. This versioning capability supports reproducibility, allows for rollback when needed, and can be used to support audits and explanations. Moreover, the architecture separates two complementary components: API Development and Deployment & Monitoring. The API Development component focuses on implementing and exposing model functionalities. In parallel, the Deployment & Monitoring component ensures that these models are monitored and maintained in production environments. This separation clarifies the boundary between development and operations. The architecture further incorporates components to manage Data and ML models. The Data Acquisition component provides interfaces for ingesting data from diverse sources. The Data Processing & Feature Engineering component manages the cleaning and transformations of raw data, and performs feature engineering. The Model Development & Training component groups services related to building, training, validating, and optimizing models. The proposed reference architecture allows updates and recovery procedures to have a minimal impact on other components. To support software evolution, SkeltyMLOps incorporates feedback loops at multiple levels. The Deployment & Monitoring component enables continuous monitoring and rollback mechanisms. In addition, it reports operational insights, such as data and concept drift, back to Project Management, where they are tracked as issue tickets. This feedback loop triggers maintenance actions within the MLOps process. In addition, end-user feedback collected through the Project Management interface is systematically integrated into the iterative development process. The Data Acquisition component further enhances adaptability by exposing interfaces for ingesting new data, thereby supporting feedback from production, enabling continuous training, and facilitating adaptation to evolving datasets. Finally, SkeltyMLOps offers client-facing interfaces that support integration with external tools, including IDEs. The modular nature of SkeltyMLOps makes it extensible and customizable. Components can be individually adapted, replaced, or omitted according to project-specific needs such as swapping the Model Storage implementation or disabling API Development in batch-only pipelines. This modularity enhances the architecture's maintainability, reusability, and adaptability.

6.2. Emphasis on MLOps Project Management & Process Orchestration

The core contribution of SkeltyMLOps lies in its explicit support for actor collaboration and activity orchestration across the MLOps processes. Unlike existing approaches that treat MLOps as a loosely connected set of tools and practices, the proposed design introduces structured components that enforce coordination and operational continuity between dimensions. At the core of this architecture are two central components: Project Management and MLOps Process Orchestration. The Project Management component initiates MLOps activities. It ingests BPMN processes and feeds them into the orchestration layer to automate process execution. The Product Owner, acting as a liaison across all actors, defines the MLOps processes by capturing requirements, aligning them with business goals, formulating the problem statement, and ensuring communication among actors. The Product Owner also supervises the MLOps backlog and issue tickets, ensuring alignment between evolving project needs and technical execution. Complementing this, the MPO component operationalizes the strategic decisions defined during project planning. Functioning as an execution layer, it continuously monitors process execution, handles feedback from runtime operations, and adapts system behavior when necessary. Through this mechanism, the MPO enforces ongoing coordination, ensuring that collaborative intentions are reflected in actual system behavior. Together, these components form the collaborative and orchestration backbone of the presented reference architecture. Their multicolored representation in Figure 3 specifically highlights the interaction and cooperation among the various actors involved within these two components.

Highlights (RQ4). SkeltyMLOps is an MLOps reference architecture designed to promote collaboration through the orchestration of activities. These activities are extracted from the literature, associated with identified MLOps actors from the literature review, and organized into components. The components are grouped into modules that reflect MLOps dimensions.

7. State of the Art on Reference Architectures for MLOps

This section examines recent MLOps reference architectures to contextualize SkeltyMLOps within the broader literature. Wozniak *et al.* [29] propose an MLOps reference architecture derived from a systematic literature review, identifying key components, tools, processes, and metrics. Although their proposed process explicitly integrates data-related activities, the final presented architecture itself lacks clearly defined data management components. This omission contrasts with our previous findings, which emphasize the importance of data management as critical to MLOps processes, a gap explicitly addressed by SkeltyMLOps. In contrast, Najafabadi *et al.* [30] conduct a systematic mapping study, identifying 35 architectural components. While their work shares motivations with SkeltyMLOps, it lists components without defining clear responsibilities or actors among them, causing overlap and potential confusion. Although Najafabadi *et al.* highlight the necessity of an orchestrator to manage interactions among components, their final architecture does not explicitly depict or detail this orchestrator. In contrast, SkeltyMLOps explicitly defines actors and their interactions based on collaboration, clearly integrating a central orchestrator into the architecture to coordinate the activities. Kreuzberger *et al.* [1] present an end-to-end MLOps architecture detailing various components and associated actors. However, their data versioning strategy primarily focuses on features while neglecting raw data and training/test dataset versioning, which are critical for reproducibility and traceability in MLOps. Additionally, their monitoring primarily covers model performance without addressing data drift, resource utilization, or overall infrastructure health, potentially hindering proactive issue detection and automated retraining processes. SkeltyMLOps addresses these limitations by providing explicit data versioning strategies across all stages of data lifecycle management and implementing comprehensive monitoring to proactively manage and adapt to performance, dataset, and model changes. In SkeltyMLOps, components are systematically organized by actors to facilitate clear and effective collaboration. At the core of SkeltyMLOps is a central orchestrator designed to coordinate interactions and ensure efficient execution across components. Additionally, SkeltyMLOps explicitly includes structured monitoring with clearly defined feedback loops, enabling systematic initiation of retraining or adaptation processes.

8. Conclusion & Perspectives

Recognizing the specific challenges posed by development and lifelong management of ML-powered software, we introduce SkeltyMLOps, a reference architecture tailored to address these challenges. The design of SkeltyMLOps follows a structured methodology: first, identifying and mapping the diverse actors and dimensions involved in MLOps, second, defining the core set of activities necessary for an MLOps process, and third, organizing these activities into software components assembled in a reference architecture. Compared to the existing literature, SkeltyMLOps stands out by focusing on actors and promoting a holistic project management and collaboration through its architecture design. We envision several extensions to this work. First, we aim to provide the LLM with the papers from the literature review as context to facilitate more precise reasoning. Second, we will design generic collaborative MLOps processes and their variants using BPMN. Third, we plan to implement a BPMN-based orchestrator for SkeltyMLOps to automate and coordinate the modeled MLOps processes. Fourth, we have initiated the development of an open-source version of SkeltyMLOps. This framework will integrate widely adopted tools and libraries. Finally, we intend to evaluate SkeltyMLOps in continuous training scenarios.

Acknowledgments. This research was partially supported by Région Occitanie (PLANETS project), which funds Charbel Daoud's PhD, and by the French National Research Agency (ANR) (AdaptiveMLOps, ANR-24-IAS2-0004), which supports the work of Christelle Urtado and Sylvain Vauttier.

Declaration on Generative AI

As discussed in Section 2.3, a large language model was used to cluster data extracted from the systematic literature review.

References

[1] D. Kreuzberger, N. Kühl, S. Hirschl, Machine Learning Operations (MLOps): Overview, Definition, and Architecture, IEEE Access 11 (2023) 31866–31879. doi:10.1109/ACCESS.2023.3262138.

[2] D. A. Tamburri, Sustainable MLOps: Trends and Challenges, in: The 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC, IEEE, Online, 2020, pp. 17–23. doi:10.1109/SYNASC51798.2020.00015.

[3] L. Zhu, L. Bass, G. Champlin-Scharff, DevOps and Its Practices, IEEE Software 33 (2016) 32–34. doi:10.1109/MS.2016.81.

[4] B. S. Farroha, D. L. Farroha, A Framework for Managing Mission Needs, Compliance, and Trust in the DevOps Environment, in: IEEE Military Communications Conference (MILCOM), IEEE, Baltimore, MD, USA, 2014, pp. 288–293. doi:10.1109/MILCOM.2014.54.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, Machine Learning: The High-Interest Credit Card of Technical Debt, in: SE4ML: Software Engineering for Machine Learning (NeurIPS 2014 Workshop), Montréal, QC, Canada, 2014. URL: https://research.google/pubs/machine-learning-the-high-interest-credit-card-of-technical-debt/.

[6] A. K. Narayanappa, C. Amrit, An Analysis of the Barriers Preventing the Implementation of MLOps, in: S. K. Sharma, Y. K. Dwivedi, B. Metri, B. Lal, A. Elbanna (Eds.), Transfer, Diffusion and Adoption of Next-Generation Digital Technologies, volume 697 of *IFIP Advances in Information and Communication Technology*, Springer, Cham, 2024, pp. 101–114. doi:10.1007/978-3-031-50188-3_10.

[7] A. M. Burgueño-Romero, A. Benítez-Hidalgo, C. Barba-González, J. F. Aldana-Montes, Toward an Open Source MLOps Architecture, IEEE Software 42 (2025) 59–64. doi:10.1109/MS.2024.3421675.

[8] A. Melde, M. Madan, P. Gavrikov, D. Hoof, A. Laubenheimer, J. Keuper, C. Reich, Tackling Key Challenges of AI Development—Insights from an Industry–Academia Collaboration, in: C. Reich, U. Mescheder (Eds.), The Upper-Rhine Artificial Intelligence Symposium (UR-AI 2022): AI Applications in Medicine and Manufacturing, Furtwangen University, Villingen-Schwenningen, Germany, 2022, pp. 112–121. URL: https://opus.hs-furtwangen.de/files/8624/Tackling.pdf.

[9] F. Calefato, F. Lanubile, L. Quaranta, A Preliminary Investigation of MLOps Practices in GitHub, in: 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM, ACM, Helsinki, Finland, 2022, pp. 283–288. doi:10.1145/3544902.3546636.

[10] A. Serban, K. van der Blom, H. Hoos, J. Visser, Software Engineering Practices for Machine Learning—Adoption, Effects, and Team Assessment, Journal of Systems and Software 209 (2024) 111907. doi:10.1016/j.jss.2023.111907.

[11] M. P. Uysal, E. Akturk, A Systemic Approach to Machine Learning Project Management, IEEE Engineering Management Review 52 (2024) 1–17. doi:10.1109/EMR.2024.3503677.

[12] M. Zarour, H. Alzabut, K. T. Al-Sarayreh, MLOps Best Practices, Challenges and Maturity Models: A Systematic Literature Review, Information and Software Technology 183 (2025) 107733. doi:10.1016/j.infsof.2025.107733.

[13] A. Ampatzoglou, S. Bibi, P. Avgeriou, M. Verbeek, A. Chatzigeorgiou, Identifying, Categorizing and Mitigating Threats to Validity in Software Engineering Secondary Studies, Information and Software Technology 106 (2019) 201–230. doi:10.1016/j.infsof.2018.10.006.

[14] A. Martín-Martín, E. Orduña-Malea, M. Thelwall, E. D. López-Cózar, Google Scholar, Web of Science, and Scopus: A systematic comparison of citations in 252 subject categories, Journal of Informetrics 12 (2018) 1160–1177. doi:10.1016/j.joi.2018.09.002.

[15] C. Daoud, D. Azar, J. Boiché, C. Urtado, S. Vauttier, SkeltyMLOps: Orchestrating Collaborative MLOps Activities, Zenodo, Version 0.1, 2025. doi:10.5281/zenodo.17094711, [Dataset].

[16] P. Trivedi, N. Choudhary, E. W. Huang, V. N. Ioannidis, K. Subbian, D. Koutra, Large Language Model Guided Graph Clustering, in: Learning on Graphs Conference (LoG), Virtual Event, 2024.

[17] V. Viswanathan, K. Gashteovski, C. Lawrence, T. Wu, G. Neubig, Large Language Models Enable Few-Shot Clustering, Transactions of the Association for Computational Linguistics 12 (2024) 321–333. doi:10.1162/tacl_a_00648.

[18] Y. Zhang, Z. Wang, J. Shang, ClusterLLM: Large Language Models as a Guide for Text Clustering, in: Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Singapore, 2023, pp. 13903–13920. doi:10.18653/v1/2023.emnlp-main.858.

[19] M. J. Mirza, L. Karlinsky, W. Lin, H. Possegger, M. Kozinski, R. Feris, H. Bischof, LaFTer: Label-Free tuning of Zero-shot classifier using language and unlabeled image collections, in: A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems 36 (NeurIPS), volume 36, Curran Associates, Inc., New Orleans, LA, USA, 2023, pp. 5765–5777. URL: https://proceedings.

neurips.cc/paper_files/paper/2023/file/123a18dfd821c8b440f42a00a27648d6-Paper-Conference.pdf.

[20] R. Zhang, Y. Li, Y. Ma, M. Zhou, L. Zou, LLMaAA: Making Large Language Models as Active Annotators, in: Findings of the Association for Computational Linguistics: EMNLP, Association for Computational Linguistics, Singapore, 2023, pp. 13088–13103. doi:10.18653/v1/2023.findings-emnlp.872.

[21] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, J. Song, R. Zhang, R. Xu, Q. Zhu, S. Ma, P. Wang, X. Bi, et al., DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, arXiv preprint (2025). doi:10.48550/arXiv.2501.12948. arXiv:2501.12948.

[22] S. Wu, Z. Peng, X. Du, T. Zheng, M. Liu, J. Wu, J. Ma, Y. Li, J. Yang, W. Zhou, Q. Lin, J. Zhao, Z. Zhang, W. Huang, G. Zhang, C. Lin, J. H. Liu, A Comparative Study on Reasoning Patterns of OpenAI's o1 Model, arXiv preprint (2024). doi:10.48550/arXiv.2410.13639. arXiv:2410.13639.

[23] M. Ggaliwango, H. Nakayiza, D. Jjingo, J. Nakatumba-Nabende, Prompt Engineering in Large Language Models, in: I. J. Jacob, S. Piramuthu, P. Falkowski-Gilski (Eds.), International Conference on Data Intelligence and Cognitive Informatics ICDICI, Algorithms for Intelligent Systems, Springer Singapore, Tirunelveli, India, 2024, pp. 387–402. doi:10.1007/978-981-99-7962-2_30.

[24] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith, D. C. Schmidt, A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, arXiv preprint (2023). doi:10.48550/arXiv.2302.11382. arXiv:2302.11382.

[25] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, M. Bone, The Concept of Reference Architectures, Systems Engineering 13 (2010) 14–27. doi:10.1002/sys.20124.

[26] E. Y. Nakagawa, Reference Architectures and Variability: Current Status and Future Perspectives, in: T. Männistö, M. A. Babar, C. E. Cuesta, J. E. Savolainen (Eds.), WICSA/ECSA Companion Volume, number 704 in ACM International Conference Proceeding Series, ACM, Helsinki, Finland, 2012, pp. 159–162. doi:10.1145/2361999.2362032.

[27] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillère, J. Klein, Y. L. Traon, Got Issues? Who Cares About It? A Large Scale Investigation of Issue Trackers from GitHub, in: IEEE 24th International Symposium on Software Reliability Engineering (ISSRE), IEEE, Pasadena, CA, USA, 2013, pp. 188–197. doi:10.1109/ISSRE.2013.6698918.

[28] D. Falessi, F. Hernandez, F. Khosmood, Issue Tracking Systems: What Developers Want and Use, in: The 13th International Conference on Software Technologies (ICSOFT, INSTICC, SciTePress, Porto, Portugal, 2018, pp. 543–548. doi:10.5220/0006818405430548.

[29] A. P. Woźniak, M. Milczarek, J. Woźniak, MLOps Components, Tools, Process and Metrics—A Systematic Literature Review, IEEE Access 13 (2025) 22166–22175. doi:10.1109/ACCESS.2025.3534990.

[30] F. A. Najafabadi, J. Bogner, I. Gerostathopoulos, P. Lago, An Analysis of MLOps Architectures: A Systematic Mapping Study, in: M. Galster, P. Scandurra, T. Mikkonen, P. O. Antonino, E. Y. Nakagawa, E. Navarro (Eds.), Software Architecture: 18th European Conference, ECSA, Luxembourg City, Luxembourg, volume 14889 of *Lecture Notes in Computer Science*, Springer, Cham, 2024, pp. 69–85. doi:10.1007/978-3-031-70797-1_5.