Foundations of GenAI Orchestration: RAG, MLOps, and LLMOps

ZhengYuan Zhang, zengzheng@wustl.edu
(A paper written under the guidance of Prof. Raj Jain)

Download 

Abstract

Those who use GPT may have a deep understanding that GenAI, which was clumsy at the beginning, couldn't remember context or keep track of previous interactions, has now become almost all-powerful, no different from humans, and even smarter and with a better memory. That is because modern GenAI systems are no longer single models working in isolation; they rely on multiple connected components to work together in a coordinated manner. For this reason, orchestration has become one of the core challenges in building reliable AI applications. In this article, I will study the three main layers of orchestration, namely the knowledge orchestration layer RAG, the infrastructure orchestration layer MLOps, and the model-behavior orchestration layer LLMOps. Combining these ideas, this article outlines how GenAI orchestration has evolved step by step from a simple "retrieve and generate" to more proactive behaviors such as planning multi-step tasks and interacting with human feedback, becoming increasingly advanced. Also discuss several future directions, including zero-touch automation, system-level orchestration across networks and hardware, and emerging possibilities such as quantum-assisted orchestration. My goal is to provide a clear and structured view of how to build and maintain GenAI systems as they continue to grow in scale and complexity.

Keywords: RAG, MLOps, LLMOps, GenAI Orchestration, DevOps

Table of Contents

- 1. Introduction
- 2. Retrieval-Augmented Generation (RAG)
- 3. MLOps Oriented Orchestration
- 4. LLMOps-Oriented Orchestration
- 5. Summary and Conclusion
- 6. Future Work and Research Directions
- List of Acronyms
- References

1. Introduction

In the past few years, generative artificial intelligence has far exceeded the concept of "a large model answering questions". Modern applications typically involve several interconnected parts: a retrieval system that provides external knowledge, tools and APIs that models can call, agents that plan multi-step tasks, and an infrastructure layer that keeps everything running. It is precisely for this reason that our views on artificial intelligence systems have also changed. It is no longer enough to merely design a good model; we must also design how all the components are organized and coordinated. This is what I call orchestration.

However, orchestration is not a single concept. Different parts of the system face different challenges. In terms of knowledge, the retrieval-augmented generation (RAG) must determine when to retrieval, how many hops to perform, and how to combine information from different sources. At the infrastructure level, MLOps maintains the stability of the machine learning lifecycle by managing data pipelines, training efforts, deployment, monitoring, and version control. With the rise of large models, companies are now dealing with a new class of problems, such as prompt drift, context window management, hallucinations, multi-round consistency, and safe execution and output filtering, which are now referred to by many researchers as LLMOps.

Although these fields are often discussed separately, they are closely related. Together, they form a hierarchical orchestration stack. RAG is responsible for the orchestration of knowledge, including how to select, retrieval and organize external information. MLOps orchestrates the ML lifecycle. LLMOps coordinates the behavior of large models when they are running. GenAI applications orchestrate and coordinate tasks, tools, agents and retrievals.

The aim of this article is not to propose a new orchestration framework, but to combine these viewpoints and provide a clearer and more structured explanation of how modern GenAI systems work, so that those who come after, especially for readers who want to understand the relationship between RAG, MLOps, LLMOps and system-level orchestration, they can efficiently understand what GenAI orchestration is. First of all, I will study how RAG has evolved from a simple retrieval mechanism into a dynamic, agent-driven knowledge workflow. Then the structure of MLOps and how it provides an operational basis for reliable artificial intelligence systems were discussed. After that, I will introduce LLMOps and explain how large models can add new orchestration requirements that traditional ML has never considered. Finally, I summarized and discussed possible future directions, including zero-touch automation and quantum computers, etc.

2. Retrieval-Augmented Generation (RAG)

This section will start with the most basic RAG to explain how the GenAI system orchestrates "knowledge".

2.1 Classical RAG

The earliest RAG was actually very simple, roughly a simple pipeline of "retrieval and generation".

The model first queries the vector database based on the input, then inserts the found fragments into the prompt, and finally concatenates these fragments directly into the context of the LLM. As IBM basically defines it in its official introduction: RAG is mainly to fill in the knowledge that the model lacks [IBM RAG].

But there are actually many problems with this kind of classical RAG. Firstly, the search is fixed, and the model does not consider "Should I search now or not?" Secondly, the strategy is very inflexible, always top-k, always one-jump search, lacking judgment ability. The model doesn't know whether the obtained information is reliable or not, and lacks the ability to dynamically adjust according to the task. Moreover, it has insufficient adaptability to real tasks, resulting in it being inadequate when the tasks are slightly complex. For instance, once reasoning or judging the context is required, it is prone to crashing.

Therefore, classical RAG is more like the "starting point" of the entire system, providing the most fundamental knowledge interface for more complex orchestration, but it cannot meet the complex demands of multi-step reasoning or real-world scenarios.

2.2 Dynamic RAG

Later, people found that if RAG kept running according to a fixed routine, its performance would not improve at all. Thus, Dynamic RAG emerged. For instance, the "retrieval chaining" mentioned in IBM's LLM Orchestration documentation essentially means allowing the LLM to determine on its own whether to search again during the generation process [IBM LLM].

Another reason why Dynamic RAG has become more flexible is that many recent systems have begun to view retrievals, agents, tools and even data sources as "modules" that can be combined as needed. Hoveyda further advanced this idea by proposing a modular retrieval structure, enabling the model to dynamically select the retrieval method suitable

for the current task [Hoveyda 25]. Their table (Table 1) shows how a task can be a simple independent step or a complex multi-step operation, and how to mix actuators and tools to build the retrieval pipeline.

Table 1: Overview of the different module types and their subcategories in our framework.

Type	Subcategory	Description	Visualization
Tasks	Standalone	Simple operations that do not require interactions between multiple agents and tools.	Standalone task
	Complex	Multi-step operations requiring agents or interactions between multiple modules.	Complex task
Executor	Agents	General execution units (i.e., foundation models) that are widely applicable and can use tools.	Agent
	Tools	Specialized execution units designed for specific tasks with limited applicability.	Tool
Resources	(Properties)	Sources of data/information; categorized on the following properties: *Structure*, *Modality* and *Accessibility*.	Resource

Table 1. Overview of module types from Hoveyda et al. [Hoveyda 25]

2.3 Agentic RAG

Although Dynamic RAG is powerful, it is Agentic RAG that really pushes RAG to the "autonomous" stage. RAG lets the LLM decide for itself how to retrieve instead of waiting for the pipeline to do it.

LLM is not merely a consumer but a strategist [Xu 25]. It generates a search plan, and the model decides whether to search and what to search through self-reflection [Asai 24]. For instance, it might think: "Should I search structured data first and then web pages?" This autonomy has far exceeded that of classical RAG.

In real systems, there are often multiple types of retrieval sources, which usually need to be uniformly coordinated by an agent. The Agent will break down the entire task into a series of sub-retrieval steps and automatically combine and invoke these tools based on the task, enhancing the adaptability of cross-domain tasks. This is more like "knowledge planning" rather than "just check and it's done", making RAG no longer merely a knowledge patch but a knowledge process that can be planned, controlled and autonomous.

2.4 Human-in-the-Retrieval

As RAG becomes increasingly intelligent, people start to discover a deeper problem: a lot of truly important knowledge is not in the database at all, but in the human brain. Human-in-the-Retrieval proposed by Xu particularly emphasizes this point: the system should regard experts as another kind of "retrieval source" and can "invoke experts" just like invoking tools [Xu 25].

Why is it necessary for humans to search the source? Because many fields of knowledge are not structured at all, and what experts know often cannot be expressed by embedding. Some tasks require "explaining why" rather than "finding the facts". Then the system should know when to ask humans. The system needs to determine "whether human feedback is needed" just as it determines "whether to search". In many scientific tasks, the information provided by humans is actually a crucial step. In this structure, the search object expands from "database" to "humans", forming a true hybrid retrieval system, enabling RAG to leverage "living knowledge sources" rather than relying solely on databases.

3. MLOps Oriented Orchestration

3.1 Why MLOps Matters

Before discussing GenAI orchestration, it is necessary to remember that traditional ML systems are already striving to achieve production. Although most fail, the failure does not come from the model itself, but from everything around it. Without a coordinated lifecycle, even a good model can become unstable very quickly.

This is the gap that MLOps attempts to address. As Kreuzberger's research summarized, the behavior of ML systems is like that of long-running software products [Kreuzberger 23]. They constantly receive new data, deteriorate over time, and rely on many loosely connected components. Therefore, without data, model training, deployment, monitoring, and feedback, the system will become unmaintainable. MLOps provides an "operational backbone" to keep everything consistent and running continuously.

3.2 What MLOps Actually Is

Kreuzberger's work provides one of the most comprehensive definitions in this field. They did not view MLOps as a set of tools, but decomposed it into 9 principles, 9 components, 7 roles and end-to-end pipeline orchestration [Kreuzberger 23].

3.2.1 Nine principles and nine components

Kreuzberger identified nine MLOps principles, such as CI/CD automation, reproducibility, version control, workflow orchestration, continuous training, metadata logging, and feedback loops. Figure 2 directly maps these principles to the 9 technical components in the actual MLOps system:

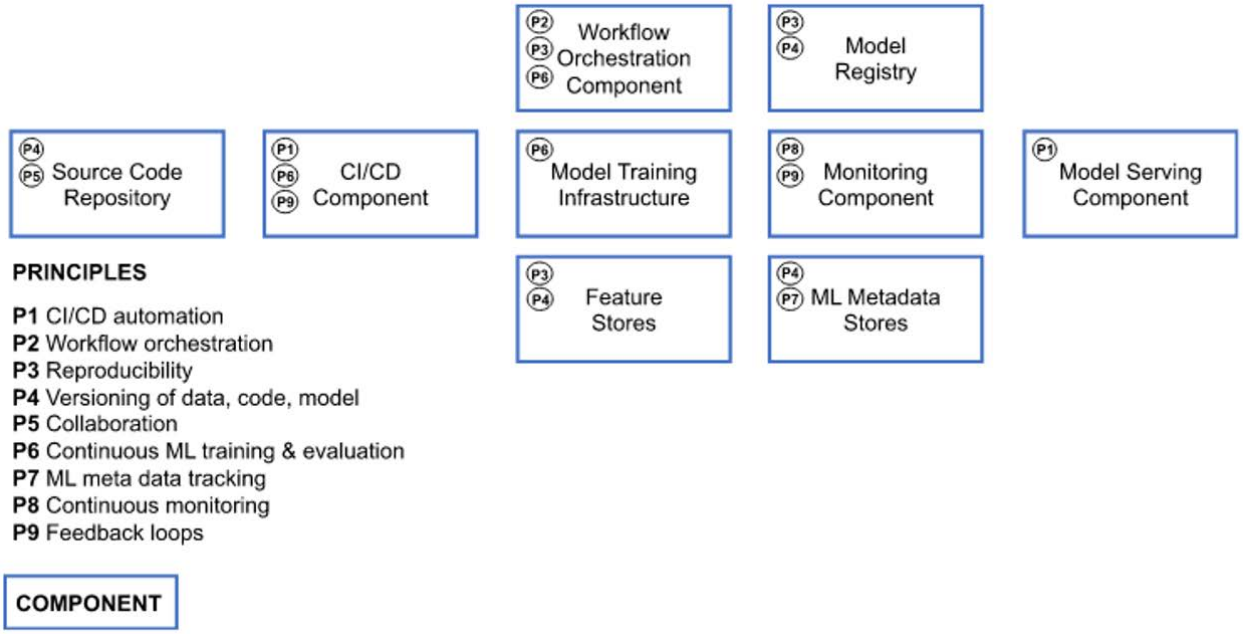


Figure 2. Nine principles and nine components [Kreuzberger 23]

This mapping clearly indicates that MLOps is not a set of isolated ideas. It is a coordinated and meticulously arranged pipeline that provides data through the pipeline to training, and then the training outcomes enter the registry. Then the registry connects to the deployment; Then to monitoring and deployment; Finally, the monitoring triggers feedback. This is an end-to-end lifecycle, in which each component plays a defined role.

3.2.2 Seven cross-functional roles

Because ML systems involve data engineering, modeling, software engineering and DevOps, Kreuzberger identified seven necessary roles, with ML/MLOps engineers at the center [Kreuzberger 23].

In [Figure 3], ML/MLOps engineers sit at the center, connecting data engineering, DevOps, backend/service engineering, software engineering, and data science. This highlights that orchestrating ML systems is not only about algorithms, it also requires coordinated human roles, because no single team can manage the entire lifecycle alone.

3.2.3 End-to-end pipeline orchestration

Their complete architecture (see Figure 4) basically shows an orchestrated pipeline, from business requirements to the data pipeline, then to experiments and then the automated workflow pipeline, finally to the model registry, deployment, monitoring, feedback loop, and then back to the early stages.

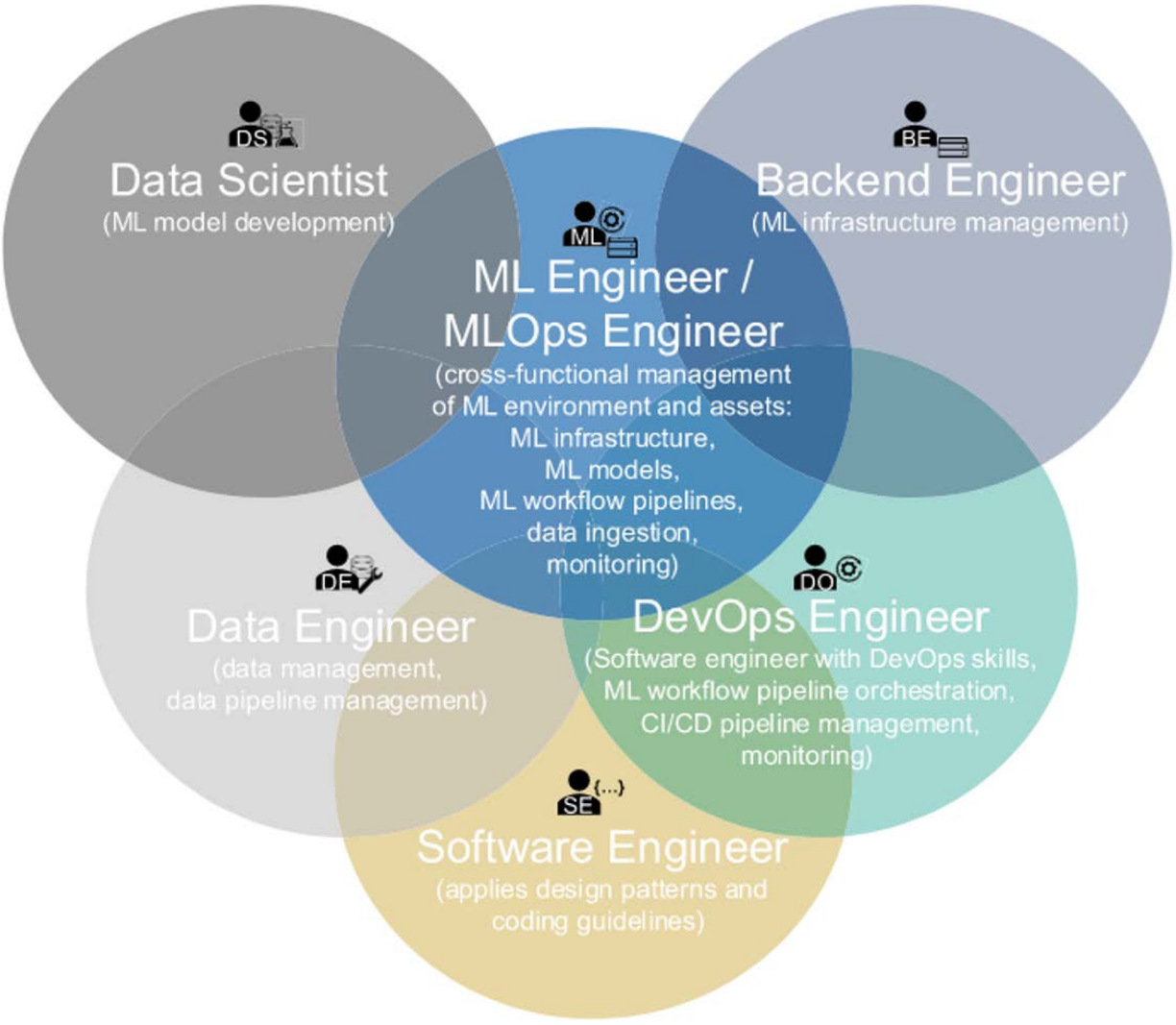


Figure 3. Seven cross-functional roles [Kreuzberger 23]

3.3 How MLOps Relates to GenAI Orchestration

With the background of 3.1 and 3.2 mentioned earlier, we can more clearly see how MLOps and GenAI orchestration are connected. It can be thought of as a natural three-tier structure: MLOps at the bottom, LLMOps in the middle, and GenAI's application orchestration at the top. At the beginning, MLOps was mainly responsible for managing the entire machine learning lifecycle, ensuring that data, training processes, versions, deployments, and monitoring could all run smoothly. Later, people began to use large models extensively, so a layer of LLMOps was added on top of MLOps. LLMOps will be explained in the next section. Next comes the application orchestration of GenAI. It not

only focuses on models but also simultaneously orchestrates tasks, retrievals, tools, agents, and various multi-step workflows. It even manages strategies and governance. So, no matter how fancy you are at the top, if the MLOps at the bottom doesn't lay a solid foundation, even the most powerful GenAI system won't run long, because MLOps for the entire layout system is like foundation for a tall building, bottom of the pyramid for the top, the people for the country.

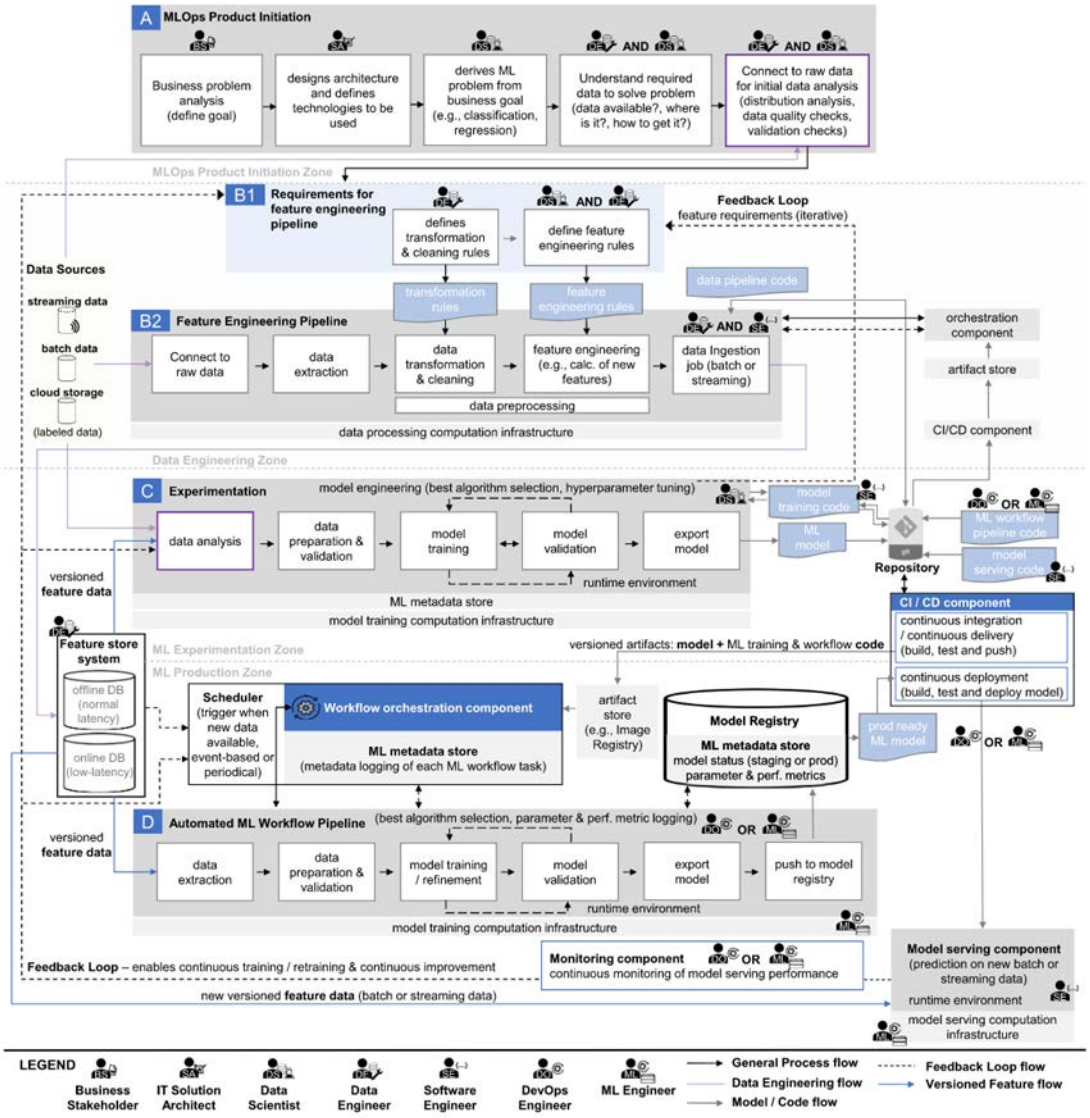


Figure 4. End-to-end MLOps architecture and workflow with functional components and roles [Kreuzberger 23]

4. LLMOps-Oriented Orchestration

Following the end of the previous section. If MLOps focuses on managing the ML lifecycle, then LLMOps is more about managing how a large model behaves while it is running.

4.1 LLMOps lifecycle

Sinha's work gives a more practical view of LLMOps. Instead of describing the ideas only in abstract terms, it lays out a full end-to-end lifecycle, from requirement definition to monitoring and responsible AI checks. Their diagram (Figure 5) is especially helpful because it shows how each stage connects to the next. It also makes it clear that LLMOps is not a single pipeline but a set of coordinated workflows that need to be orchestrated depending on how the application evolves.

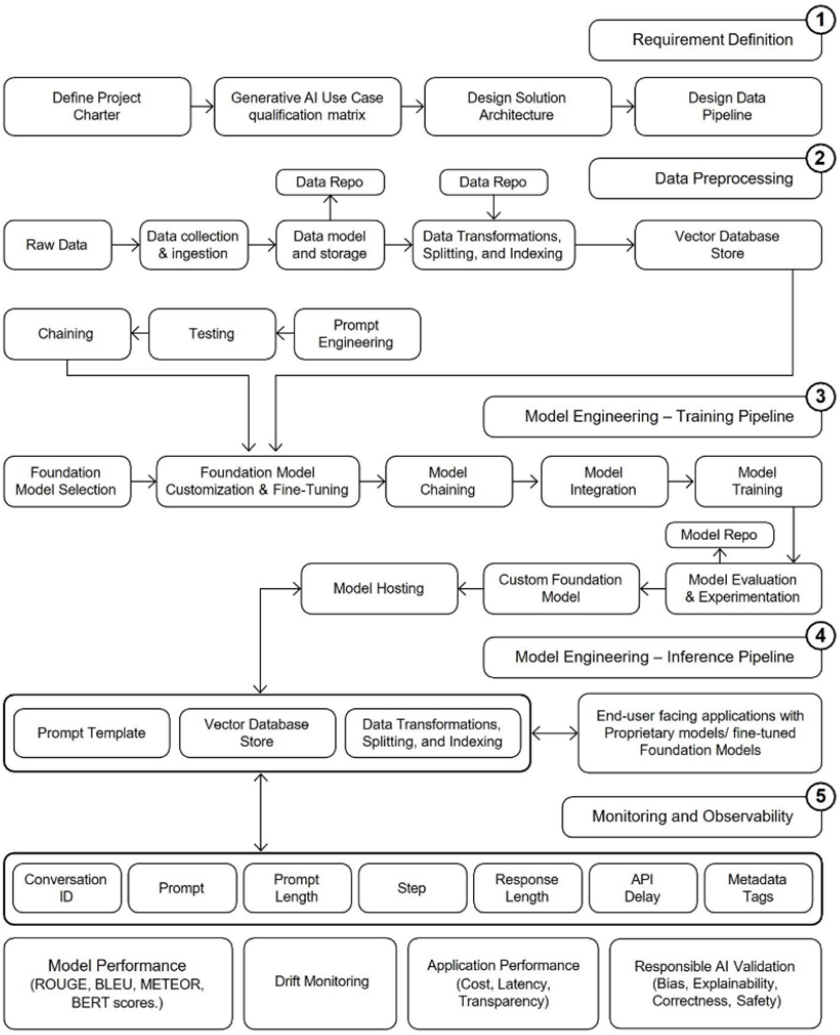


Figure 5. LLMOps framework [Sinha 24]

Compared with MLOps, the weight of governance in LLMOps has significantly increased. In practice, LLMOps is not just an extension of MLOps. It forms its own pipeline built around several elements: large foundation models, prompts, human feedback, and a much stronger emphasis on security and governance.

This life cycle includes the following key stages. First, LLMOps involves security, governance, and risk control from the very beginning, rather than focusing only on technical processes like MLOps. Moreover, the evaluation of LLMs is completely different from the logic of traditional ML. The data flow in LLMOps includes but is not limited to collecting internal context data of enterprises, building vector databases, handling noise, anonymization, prompt tuning, and domain adaptation, as well as LLM-specialized evaluation.

Sinha regards "monitoring" as one of the cores of LLMOps. He believes that the monitoring of LLMs is not about observing accuracy, but about observing behavior [Sinha 24]. Because the main monitoring contents are the degree of hallucination, prompt injection attack detection, continuity of multi-round conversations, factual consistency, user ratings, and other behavioral indicators.

Then, as mentioned at the beginning, LLMOps places more emphasis on governance. The governance part of LLMOps must undergo audits, manage model versions, usage permissions, and sensitive-content policies. Together, the project charter, data processing, monitoring, and governance constitute the full lifecycle of LLMOps.

In other words, LLMOps acts as the orchestrator inside the model layer. Its job is to make sure the model remains controllable and predictable even when the context and tasks keep changing.

4.2 What New Orchestration Requirements Have LLMs Introduced

As large language models gradually become core components of enterprise-level systems, they bring many new challenges that traditional ML systems do not have, and thus also put forward brand-new orchestration requirements.

First is prompt drift. Since humans constantly change prompt words and the model's own version may also be updated, as the system runs for a long time, the prompt will gradually "drift", and the behavior of the same task will eventually deviate. This means that an orchestration requirement is needed that can track prompt history, monitor and respond to offsets, and then trigger rollback or re-evaluation.

Then there is context management. The system needs to know which content should be placed in the context, which must be trimmed, and how to avoid forgetting, etc. This is regarded as one of the most core scheduling elements in LLM systems [IBM LLM].

When the system is connected to multiple data sources and multiple retrievers, a routing layer is needed to make decisions, such as when to search, how many hops to search, which retriever to select, and how to combine the search results, etc.

In Sinha's research [Sinha 24], what enterprise systems are most concerned about is the security of the output, such as whether it contains sensitive data, whether it is attacked by prompt injection, and whether it generates harmful content. Therefore, LLMOps must orchestrate an additional layer of security policies and model output filtering.

At the same time, like other orchestrates, LLMs require a stable human feedback process. To reduce hallucinations and biases, it is necessary to know which outputs require human intervention and how to record feedback, etc.

5. Summary and Conclusion

Looking back at the previous sections, we can see that all three layers are actually trying to solve the same fundamental problem: As artificial intelligence systems become increasingly complex, how do we control everything and ensure that the systems operate in the way we expect?

In Section 2, the evolution of RAG was discussed. The initially simple "search and generate" model has gradually transformed into a more intelligent one. Today's RAG systems can determine when to retrieve, from where to retrieve, how many hops there are, and even involve proxies or human reviewers when necessary. Therefore, RAG itself becomes a kind of "knowledge coordinator", and it is no longer passive. It actively organizes information for the model.

Then in Section 3, MLOps took this idea to a different level. MLOps is not about orchestrating knowledge, but about orchestrating the ML lifecycle. It enables all the basic components to operate in a stable and predictable manner. Without this foundation, even the best model will collapse very quickly.

Section 4 points to LLMOps. Large models introduce a whole set of new problems and stricter security requirements that traditional ML has never dealt with before. LLMOps attempts to coordinate these behaviors and ensure that the model remains reliable when tasks and contexts are constantly changing. Compared with MLOps, the focus has clearly

shifted from “managing the lifecycle of a model” to “managing how the model behaves while it is running.”

RAG, MLOps, and LLMOps are not substitutes for each other; they are interdependent. Without a strong MLOps foundation, LLMOps cannot run well, and without a stable LLMOps, GenAI orchestration cannot reliably coordinate tasks or agents. With the continuous development of artificial intelligence systems and the increasing number of tasks, in the near future, we may witness more orchestration demands.

Therefore, as the system grows more complicated, it becomes increasingly important to think not only about each layer individually, but also about how to orchestrate these layers in a more coherent and practical way. It becomes one of the key challenges for the next stage of GenAI applications.

6. Future Work and Research Directions

No matter how advanced the current GenAI orchestration is, it is still mainly about coordinating models, tools and tasks. However, it is not difficult to see from the trend of previous papers that orchestration itself is transforming from "managing AI applications" to "AI-driven, automatically evolving system intelligence".

Nowadays, there are also some newer and more powerful models demonstrating this trend. Zero-touch DevOps [Utham 2023] shows us that in the future, orchestration will no longer be about humans writing processes, but rather AI automatically generating, fixing, and executing the entire software lifecycle based on context. This means that GenAI orchestration will transform into an automatic inference of intent, automatically creating and updating workflows, debugging issues, and adjusting policies, becoming a fully automatic and dynamic generative model.

NetGPT [Chen 24] also opens an interesting direction for future research, which is that the future network itself will become an "AI-native system", where nodes, paths and resources can all be coordinated and scheduled through Agents. It is also indicated in Rail-Only [Wang 24] that in the future, artificial intelligence orchestration must be optimized in combination with the underlying computing power topology rather than building models on traditional data centers. This indicates that in the future, GenAI orchestration will sink from the application layer to the infrastructure layer. Maybe in the future, network paths, GPU topologies, fault detection and recovery, etc. will also be managed.

Zeydan's quantum computer running the GenAI orchestration concept also opens a door to the future. He proposed that workflows are not written but "searched" out, routes are

not calculated but obtained through "quantum search", and topologies are not designed by humans but jointly generated by AI and quantum [Zeydan 24]. This is a very cutting-edge yet reasonable future direction, using what quantum computers excel at to solve large-scale combinatorial optimization problems and accelerate operational speed.

From this, it can be seen that the future of GenAI orchestration is bright. There are still many different paths to develop in this field and many mysteries waiting for people to explore.

List of Acronyms

- **AI** — Artificial Intelligence
- **API** — Application Programming Interface
- **CI/CD** — Continuous Integration / Continuous Deployment
- **GenAI** — Generative Artificial Intelligence
- **GPU** — Graphics Processing Unit
- **LLM** — Large Language Model
- **LLMOps** — Operations for Large Language Models
- **MLOps** — Machine Learning Operations
- **RAG** — Retrieval-Augmented Generation
- **SDLC** — Software Development Life Cycle

References

1. "What is AI orchestration?" https://www.ibm.com/think/topics/ai-orchestration | IBM Think — Explains the coordination and management of AI models, systems, and integrations, covering deployment, integration, automation, and governance.
2. "What is LLM orchestration?" https://www.ibm.com/think/topics/llm-orchestration | IBM Think — Explains the concept of LLM orchestration, including its role in prompting, chaining, managing, and monitoring LLMs.
3. "What is RAG (retrieval augmented generation)?" https://www.ibm.com/think/topics/retrieval-augmented-generation | IBM Think — Explains how RAG connects LLMs to external knowledge bases, its components, benefits, and use cases.
4. Y. Chen, R. Li, Z. Zhao, C. Peng, J. Wu, and E. Hossain, "*NetGPT: An AI-Native Network Architecture for Provisioning Beyond Personalized Generative Services*," IEEE Network, vol. 38, no. 6, pp. 404–413, Nov. 2024. Available: https://ieeexplore.ieee.org/abstract/document/10466747
5. W. Wang, M. Ghobadi, K. Shakeri, Y. Zhang, and N. Hasani, "*Rail-only: A Low-Cost High-Performance Network for Training LLMs with Trillion Parameters*," arXiv

preprint arXiv:2307.12169, 2024. Available: https://doi.org/10.48550/arXiv.2307.12169

6. M. Hoveyda, H. Oosterhuis, A. P. de Vries, M. de Rijke, and F. Hasibi, "*Adaptive Orchestration of Modular Generative Information Access Systems*," in Proc. 48th Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval, 2025, pp. 3899–3910. Available: https://doi.org/10.1145/3726302.3730351
7. A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "*SELF-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*," in Proc. Int. Conf. Learn. Representations (ICLR), Vienna, Austria, May 2024. Available: https://selfrag.github.io
8. F. Cuconasu, G. Trappolini, F. Siciliano, S. Filice, C. Campagnano, Y. Maarek, N. Tonellotto, and F. Silvestri, "*The Power of Noise: Redefining Retrieval for RAG Systems*," in Proc. 47th Int. ACM SIGIR Conf. Res. Dev. Inf. Retrieval, 2024, pp. 719–729. Available: https://doi.org/10.1145/3626772.3657834
9. U. K. A. Sethupathy, "*Zero-touch DevOps: A GenAI-orchestrated SDLC automation framework*," World Journal of Advanced Engineering Technology and Sciences, vol. 08, no. 02, pp. 420–433, 2023. Available: https://doi.org/10.30574/wjaets.2023.8.2.0119
10. M. Sinha, S. Menon, and R. Sagar, "*LLMOps: Definitions, Framework and Best Practices*," in Proc. 2024 Int. Conf. Electrical, Computer and Energy Technologies (ICECET), Sydney, Australia, Jul. 2024. Available: https://doi.org/10.1109/ICECET61485.2024.10698359
11. D. Kreuzberger, N. Kühl, and S. Hirschl, "*Machine Learning Operations (MLOps): Overview, Definition, and Architecture*," IEEE Access, vol. 11, pp. 31866–31879, Mar. 2023. Available: https://doi.org/10.1109/ACCESS.2023.3262138
12. K. Parthasarathy, K. Vaidhyanathan, R. Dhar, V. Krishnamachari, A. Kakran, and S. Akshathala, "*Engineering LLM Powered Multi-Agent Framework for Autonomous CloudOps*," in Proc. 2025 IEEE/ACM 4th Int. Conf. AI Engineering – Software Engineering for AI (CAIN), Ottawa, Canada, Apr. 2025. Available: https://doi.org/10.1109/CAIN66642.2025.00031
13. X. Xu, D. Zhang, Q. Liu, Q. Lu, and L. Zhu, "*Agentic RAG with Human-in-the-Retrieval*," in Proc. 2025 IEEE 22nd Int. Conf. Software Architecture Companion (ICSA-C), Odense, Denmark, Mar.–Apr. 2025. Available: https://doi.org/10.1109/ICSA-C65153.2025.00074
14. E. Zeydan, L. Blanco, J. Mangues-Befaluy, S. S. Arslan, and Y. Turk, "*Next-Generation Orchestration: Quantum Computing for Network Services*," in Proc. 2024 Int. Conf. on Computer, Information and Telecommunication Systems (CITS), Girona, Spain, Jul. 2024. Available: https://doi.org/10.1109/CITS61189.2024.10608025

15. S. Joshi, "*Introduction to Generative AI and DevOps: Synergies, Challenges and Applications*," International Journal of Advanced Research in Science, Communication and Technology (IJARSCT), vol. 5, no. 1, pp. 205–214, Mar. 2025. Available: https://doi.org/10.48175/IJARSCT-23634

Last modified on November 17, 2025
This and other papers on recent advances in networking are available online at
http://www.cse.wustl.edu/~jain/cse5700-25/index.html
Back to Raj Jain's Home Page