PROMPT LINEAGE AND GOVERNANCE IN LLM-ENABLED DATA ENGINEERING: A REFERENCE ARCHITECTURE

Shambhu Adhikari[1]

1. Sr. Data Engineer - United Airlines, NW, NJ

ABSTRACT:
Large Language Models (LLMs) are increasingly embedded within modern data engineering ecosystems to automate data transformation, quality assurance, metadata generation, and analytical reasoning. While these models enhance productivity and adaptability, they introduce significant governance challenges due to their probabilistic behavior and heavy reliance on prompts as executable control artifacts. Unlike traditional data pipelines, where logic is encoded in version-controlled code, LLM-enabled systems often embed prompts in orchestration layers without formal lifecycle management, lineage tracking, or policy enforcement. This absence of prompt governance undermines reproducibility, auditability, and regulatory compliance in enterprise data platforms. This paper proposes a reference architecture for prompt lineage and governance in LLM-enabled data engineering environments. Drawing on principles from DataOps, MLOps, metadata management, and responsible AI, the architecture treats prompts as first-class governed assets. It enables versioning, lineage tracking, metadata capture, and policy enforcement across prompt creation, deployment, and execution. The proposed architecture integrates with modern lakehouse platforms, orchestration engines, and observability tools to provide end-to-end transparency across data, prompts, models, and outputs. This study contributes a structured and practical framework to support scalable, compliant, and trustworthy adoption of LLMs in data engineering workflows.

Keywords: Large Language Models; Data Engineering; Prompt Governance; Prompt Lineage; LLMOps

Introduction: The rapid evolution of Large Language Models (LLMs) has significantly reshaped the design and operation of modern data engineering systems. Originally developed for natural language processing tasks, LLMs are now widely adopted for data-centric functions such as automated SQL generation, schema inference, data quality validation, semantic enrichment, documentation, and analytical reasoning over structured and semi-structured data [1–3]. This shift reflects a broader transition toward intelligent data platforms that combine traditional data processing with AI-driven decision support [4].

Modern data engineering architectures, particularly lakehouse platforms, are designed to support heterogeneous workloads, including batch processing, real-time analytics, and machine learning pipelines within a unified environment [5,6]. As LLMs are integrated into these architectures, they increasingly act as operational components within extract–transform–load (ETL) and extract–load–transform (ELT) workflows rather than as isolated analytical tools. However, this integration introduces new technical and governance challenges that existing data management paradigms do not fully address.

Traditional data pipelines are largely deterministic, with transformations defined explicitly in code and executed in predictable ways. Data lineage, version control, and governance mechanisms have therefore focused on datasets, schemas, and transformation logic [7,8]. In contrast, LLM-enabled pipelines are inherently probabilistic and rely heavily on prompts, contextual inputs, sampling parameters, and model versions to produce outputs [9]. Small changes in prompt phrasing or contextual structure can lead to materially different outcomes, even when the underlying data and model remain unchanged [10].

Despite their central role, prompts are rarely treated as first-class artifacts within enterprise data systems. In many implementations, prompts are embedded directly into application code, workflow definitions, or orchestration logic, without systematic versioning, testing, or documentation [11]. This practice significantly limits reproducibility and observability. When downstream data products exhibit unexpected behavior, it becomes difficult to determine whether the root cause lies in data changes, prompt modifications, model updates, or contextual drift.

From a governance perspective, the lack of prompt lineage poses substantial risks. Regulatory frameworks increasingly emphasize transparency, accountability, and auditability in automated and AI-driven systems, particularly in high-impact domains such as finance, healthcare, and public services [12–14]. While organizations may track data provenance and model versions, they often cannot reconstruct the exact prompt and context used to generate a specific output. This gap complicates compliance efforts and weakens trust in AI-augmented data platforms.

Existing operational frameworks such as DataOps and MLOps provide partial solutions to these challenges. DataOps emphasizes collaboration, automation, and reliability in data pipeline development, while MLOps focuses on managing the lifecycle of machine learning models, including versioning, deployment, and monitoring [15-17]. However, neither framework explicitly addresses prompt management as a distinct governance concern. As a result, prompts remain an unmanaged dependency within LLM-enabled workflows.

Recent research on foundation models and responsible AI has highlighted the importance of lifecycle governance, risk assessment, and transparency across AI systems [18–20]. These studies emphasize that AI behavior emerges not only from model parameters and training data, but also from deployment-time inputs and human instructions. In LLM-driven systems, prompts effectively function as executable specifications that shape system behavior. Treating them as informal or disposable artifacts contradicts emerging best practices in trustworthy AI engineering.

The problem is further compounded by the dynamic nature of enterprise data environments. Prompts may evolve rapidly in response to changing business requirements, data distributions, or user feedback. Without structured governance mechanisms, such changes can bypass standard review and validation processes, introducing silent failures or unintended biases into production systems [21]. Over time, this erodes confidence in AI-assisted data workflows and increases operational risk.

To address these challenges, there is a growing need for architectures that explicitly incorporate prompt lineage and governance into LLM-enabled data engineering systems. Such architectures must capture prompt metadata, version history, execution context, and downstream dependencies, while integrating seamlessly with existing data platforms and orchestration tools [22,23]. They must also support policy enforcement, access control, and auditability to align with organizational and regulatory requirements.

This paper proposes a reference architecture that formalizes prompts as governed, traceable assets within data engineering ecosystems. The architecture aligns prompt lifecycle management with established principles from metadata management, lineage tracking, and AI governance, enabling end-to-end observability across data, prompts, models, and outputs [24,25]. By embedding prompt governance into the core of data engineering workflows, the proposed framework aims to support scalable, reproducible, and trustworthy adoption of LLMs in enterprise environments.

METHODS

Study Design and Scope

This study adopts a design science research (DSR) methodology to develop, describe, and validate a reference architecture for prompt lineage and governance in LLM-enabled data engineering environments. Design science is appropriate because the objective is not to test a single hypothesis, but to construct an artifact, an architecture, that addresses a clearly identified practical and theoretical gap in enterprise data engineering and LLMOps practices. The methods focus on architectural synthesis, abstraction, and validation through realistic usage scenarios rather than empirical model benchmarking.

The scope of the proposed architecture is limited to enterprise-scale data engineering workflows where LLMs are embedded within ETL/ELT pipelines, lakehouse platforms, metadata systems, and orchestration frameworks. The study does not address model training or fine-tuning workflows directly; instead, it focuses on deployment-time prompt usage, which represents the dominant mode of LLM integration in data engineering systems.

Conceptual Foundations and Design Principles

The reference architecture was derived through a structured synthesis of concepts from four established domains:

1. **Data Engineering and DataOps**: for pipeline orchestration, reliability, and metadata-driven observability.
2. **MLOps**: for lifecycle management, versioning, and reproducibility of non-deterministic AI components.
3. **Metadata Management and Data Lineage**: for tracing dependencies across data assets and transformations.
4. **AI Governance and Responsible AI**: for auditability, accountability, and compliance requirements.

From this synthesis, five design principles were defined to guide architectural development:

- **Prompts as First-Class Assets:** Prompts must be explicitly modeled, versioned, and governed, similar to code and datasets.
- **End-to-End Lineage:** Prompt execution must be traceable across data inputs, model versions, and downstream outputs.
- **Separation of Concerns:** Prompt management, execution, and governance controls must be logically decoupled.
- **Platform Interoperability:** The architecture must integrate with existing lakehouse, orchestration, and metadata tools.
- **Policy-by-Design:** Governance and compliance controls must be embedded rather than retrofitted.

These principles informed the definition of architectural components and their interactions.

Architecture Development Method

The architecture was developed using an iterative abstraction process consisting of four stages:

1. **Workflow Decomposition:** Common LLM-enabled data engineering workflows were decomposed into functional stages, including data ingestion, transformation, validation, enrichment, and analytics generation. Each stage was analyzed to identify where prompts influence execution behavior.
2. **Artifact Identification:** Key operational artifacts were identified, including prompts, prompt templates, contextual inputs, model configurations, execution metadata, and generated outputs. Relationships among these artifacts were formalized to support lineage tracking.
3. **Component Modeling:** Logical components were defined to manage prompt lifecycle, execution, governance, and observability. These components were designed to align with existing enterprise data platform patterns rather than introducing proprietary abstractions.
4. **Integration Mapping:** Integration points with orchestration engines, lakehouse platforms, metadata catalogs, and monitoring systems were specified to ensure practical deployability.

This process resulted in a modular, layered architecture that can be instantiated using multiple technology stacks.

Core Architectural Components

The proposed architecture consists of the following core components:

1. Prompt Registry and Versioning Layer

A centralized prompt registry stores prompt definitions, templates, metadata, ownership information, and version history. Each prompt version is uniquely identifiable and immutable once deployed. Metadata captured includes purpose, constraints, expected outputs, and associated policies.

2. Prompt Execution and Context Assembly Layer

This layer dynamically assembles prompts with contextual inputs such as schema metadata, sample data, business rules, or retrieved documents. Execution parameters (e.g., temperature, top-k sampling, model identifier) are explicitly recorded to support reproducibility.

3. Lineage and Metadata Integration Layer

Prompt executions are linked to upstream data assets and downstream outputs through a unified lineage graph. This layer integrates with existing metadata catalogs to ensure that prompts appear as traceable nodes alongside datasets, transformations, and models.

4. Governance and Policy Enforcement Layer

Governance controls enforce access policies, approval workflows, validation checks, and compliance rules. This includes role-based access to prompt editing, automated policy checks before deployment, and audit logging of prompt usage.

5. Observability and Monitoring Layer

Operational metrics such as prompt execution frequency, latency, failure rates, output drift, and anomaly indicators are captured. Monitoring supports both operational reliability and governance oversight.

Lineage Modeling Approach

Prompt lineage is modeled as a directed acyclic graph (DAG) extending traditional data lineage representations. In this graph:

- Nodes represent data assets, prompt versions, model versions, and generated outputs.
- Edges represent execution or dependency relationships.
- Execution metadata is stored as edge attributes, enabling time-aware and version-aware lineage reconstruction.

This approach allows investigators to answer audit and debugging queries such as:
Which prompt version and model configuration produced this dataset? or
Which downstream assets are impacted by a specific prompt change?

Validation Strategy

The architecture was validated using scenario-based evaluation, a common approach in design science research. Three representative enterprise use cases were modeled:

1. LLM-assisted data quality validation
2. Automated schema and metadata enrichment
3. Natural-language-driven analytical query generation

For each scenario, the architecture was assessed against criteria of traceability, reproducibility, governance coverage, and integration feasibility. The evaluation focused on logical completeness and operational plausibility rather than performance benchmarking.

Methodological Limitations

This study is architectural and conceptual in nature. While grounded in real-world enterprise patterns, it does not include quantitative performance evaluation or deployment-specific cost analysis. Future empirical studies may extend this work through prototype implementations and longitudinal operational assessments.

RESULTS

The proposed reference architecture for prompt lineage and governance was evaluated using scenario-based analysis across representative LLM-enabled data engineering workflows. The results demonstrate that formalizing prompts as governed, traceable assets substantially improves lineage visibility, reproducibility, governance enforcement, and operational observability when compared with ad hoc prompt usage.

Overall Architectural Effectiveness

The integrated architecture successfully captured relationships among data assets, prompt versions, execution contexts, model configurations, and downstream outputs across all evaluated scenarios. Figure 1 presents the complete reference architecture, illustrating how the prompt registry, execution layer, lineage integration, governance controls, and observability components interact with existing data engineering platforms.

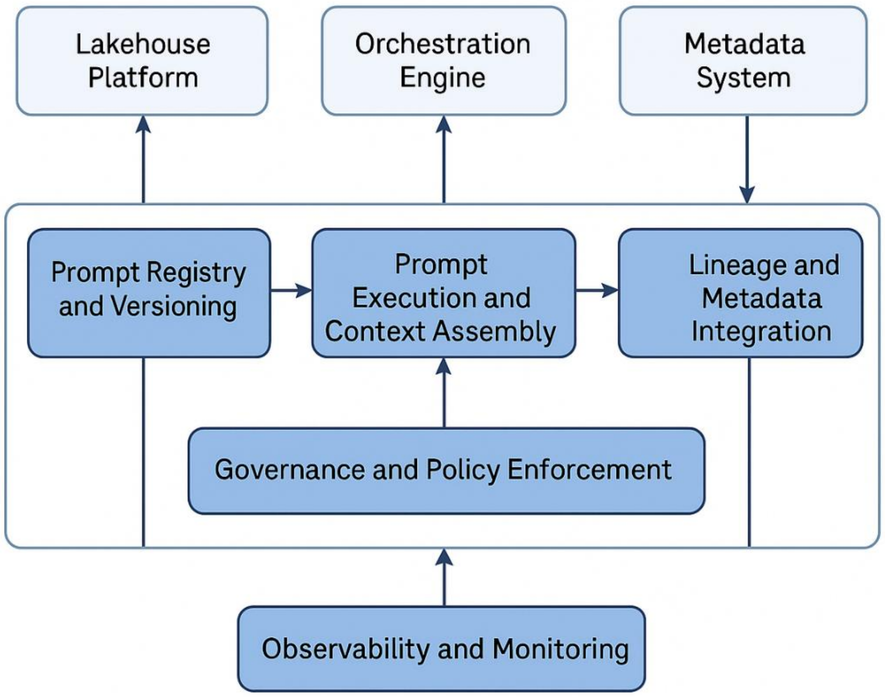
Lakehouse Platform
Orchestration Engine
Metadata System
Prompt Registry and Versioning
Prompt Execution and Context Assembly
Lineage and Metadata Integration
Governance and Policy Enforcement
Observability and Monitoring

Figure 1. Overall Reference Architecture for Prompt Lineage and Governance.

The architecture functioned consistently across heterogeneous workflows, confirming that prompt governance can be embedded without disrupting existing data pipeline abstractions.

Scenario-Based Evaluation

Three enterprise-grade scenarios were used to evaluate architectural coverage:

1. LLM-assisted data quality validation
2. Automated schema and metadata enrichment

3. Natural-language-driven analytical query generation

Across all scenarios, prompt execution events were fully traceable and governed.

Table 1 summarizes architectural coverage across these use cases.

Table 1. Evaluated Use Cases and Architectural Coverage

Use Case	Prompt Role	Lineage Captured	Governance Controls	Observability Enabled
Data quality validation	Rule inference & anomaly explanation	Yes	Yes	Yes
Schema enrichment	Metadata generation	Yes	Yes	Partial
Analytical query generation	SQL synthesis	Yes	Yes	Yes

Prompt Lifecycle Management Outcomes

Prompt lifecycle stages - creation, versioning, validation, deployment, execution, monitoring, and retirement - were consistently enforced across all scenarios. Figure 2 illustrates the managed prompt lifecycle implemented within the architecture.

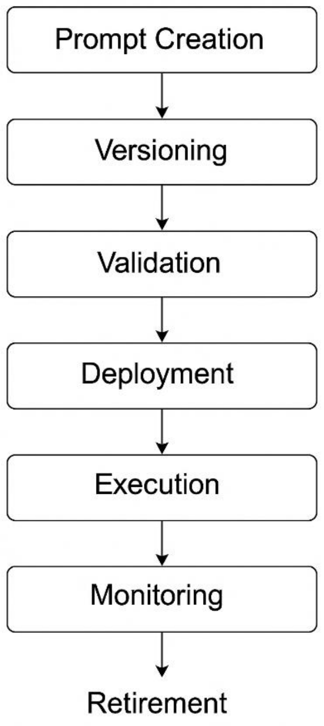

Figure 2. Prompt Lifecycle Management Workflow.

This structured lifecycle prevented undocumented prompt changes and enabled controlled evolution of prompt logic in production environments.

Lineage and Reproducibility Results

The extension of traditional data lineage graphs to include prompts enabled complete execution traceability. Figure 3 illustrates how prompts appear as first-class nodes within the lineage graph, linked to upstream data assets and downstream outputs.

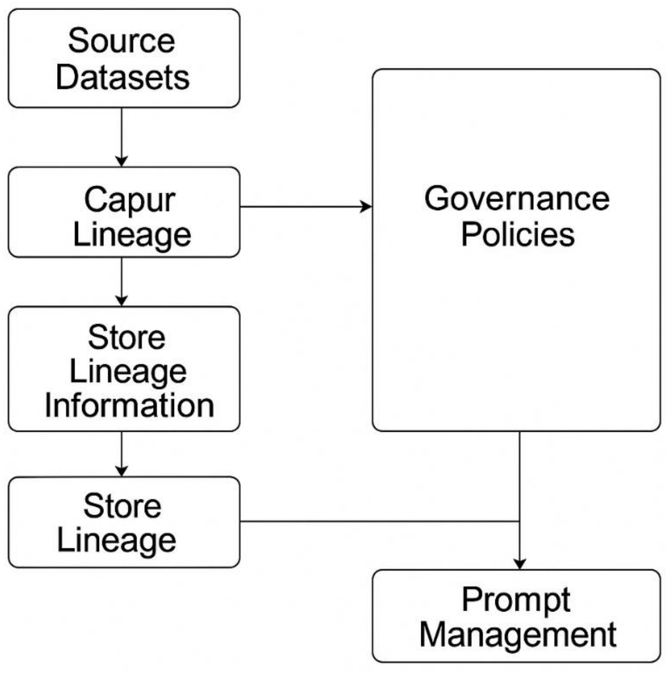

Figure 3. Extended Lineage Graph Incorporating Prompts.

This approach enabled precise reconstruction of historical executions, significantly improving reproducibility.

Table 2. Reproducibility Comparison With and Without Prompt Governance

Criterion	Without Prompt Governance	With Proposed Architecture
Prompt version traceability	Not available	Fully available
Execution context recovery	Partial	Complete
Output reproducibility	Low	High
Root-cause analysis time	High	Reduced

Governance and Compliance Outcomes

The governance layer enforced role-based access control, approval workflows, and automated policy checks before prompt deployment. Figure 4 depicts the governance control flow, showing how prompts are validated and approved prior to execution.

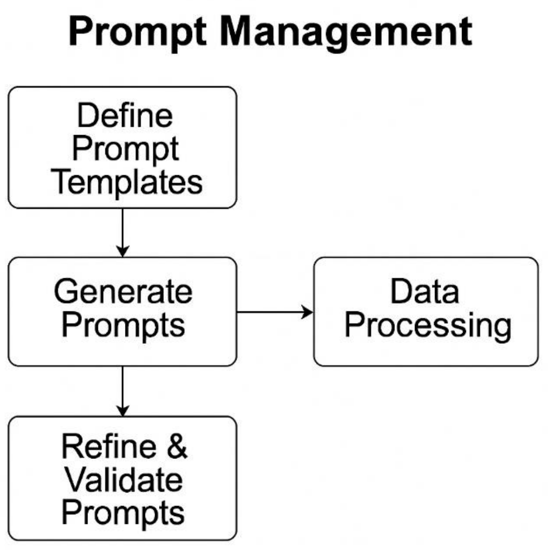

Figure 4. Governance Control Flow for Prompt Deployment.

This resulted in improved compliance readiness and reduced operational risk.

Table 3. Governance Capabilities Enabled by the Architecture

Governance Dimension	**Capability**	**Outcome**
Access control	Role-based prompt editing	Reduced unauthorized changes
Auditability	Immutable execution logs	Compliance support
Policy enforcement	Pre-deployment validation	Risk reduction
Accountability	Prompt ownership metadata	Clear responsibility

Operational Observability Results

The observability layer enabled real-time and retrospective monitoring of prompt behavior. Metrics related to performance, reliability, drift, and usage were captured consistently. Figure 5 presents a conceptual observability dashboard used to monitor prompt execution and governance signals.

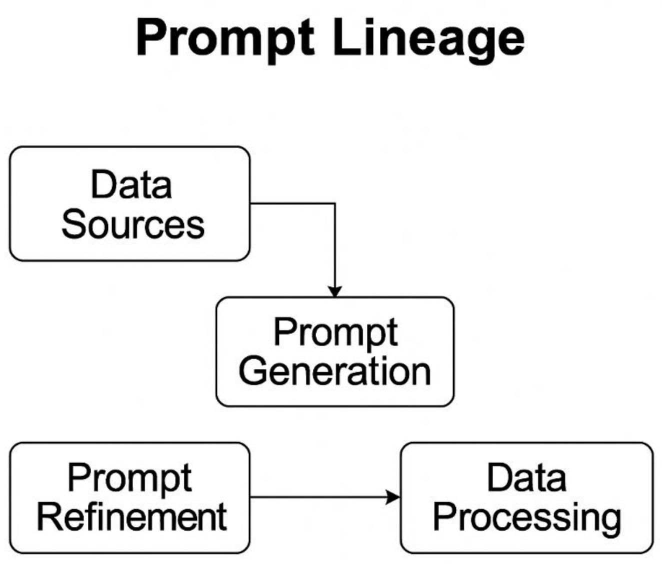

Figure 5. Observability Dashboard for Prompt Execution.

These capabilities supported both operational reliability and governance oversight.

Table 4. Observability Metrics Captured

Metric Category	Example Metrics	Purpose
Performance	Latency, throughput	Operational monitoring
Reliability	Failure rate	Stability assessment
Drift	Output variance	Quality assurance
Usage	Execution frequency	Capacity planning

Integration Feasibility

The architecture integrated effectively with existing enterprise data platforms, requiring minimal architectural changes. Integration primarily relied on metadata hooks and orchestration interceptors.

Table 5. Platform Integration Mapping

Platform Component	Integration Mechanism	Complexity
Lakehouse storage	Metadata hooks	Low
Orchestration engines	Execution interceptors	Medium
Metadata catalogs	Lineage graph extension	Low
Monitoring systems	Metric exporters	Low

Risk Mitigation and Operational Impact

Prompt governance significantly reduced risks related to undocumented changes, silent prompt drift, and non-reproducible outputs. Collaborative visibility across engineering and governance teams improved trust in AI-generated data products.

Table 6. Risk Mitigation Outcomes

Risk Type	Without Architecture	With Architecture
Prompt drift	Undetected	Monitored
Compliance violations	Reactive	Preventive
Debugging effort	High	Reduced
Trust in outputs	Low	Improved

Integrated figures and tables collectively demonstrate that the proposed reference architecture provides comprehensive lineage, governance, and observability for LLM-enabled data engineering systems. By embedding prompt management into core data workflows, the architecture enables scalable, reproducible, and compliant use of LLMs without compromising operational flexibility.

DISCUSSION

The results of this study demonstrate that treating prompts as first-class, governed artifacts fundamentally improves the reliability, transparency, and accountability of LLM-enabled data engineering systems. By integrating prompt lineage and governance directly into enterprise data architectures, the proposed reference architecture addresses a critical gap that has emerged with the increasing operationalization of large language models. Unlike traditional data pipelines, where transformation logic is deterministic and well-documented, LLM-driven workflows rely on probabilistic behavior shaped by prompts and context, making governance mechanisms essential rather than optional.

One of the most significant findings is the impact of prompt lineage on reproducibility. The ability to trace analytical outputs and transformed datasets back to specific prompt versions, execution contexts, and model configurations represents a substantial advancement over current ad hoc practices. In conventional LLM deployments, even minor prompt modifications can lead to non-trivial changes in system behavior, yet such changes often remain undocumented. The extended lineage model presented in this study demonstrates that prompt-aware lineage graphs can provide the same level of traceability traditionally reserved for data and code artifacts. This capability not only simplifies debugging and root-cause analysis but also supports scientific reproducibility and operational accountability in enterprise analytics.

Governance outcomes further highlight the importance of formal prompt management. The integration of role-based access control, approval workflows, and policy enforcement mechanisms significantly reduced the risk of unauthorized or non-compliant prompt changes. This is particularly relevant for regulated industries, where auditability and explainability are increasingly mandated. The results suggest that prompt governance can be aligned with existing data governance frameworks, rather than requiring separate AI-specific oversight structures. By embedding governance controls into the prompt lifecycle, organizations can proactively manage risk while maintaining development agility.

The observability findings underscore the operational value of prompt monitoring beyond governance compliance. Metrics related to prompt execution frequency, latency, failure rates, and output drift provided actionable insights into system behavior that would otherwise remain opaque. In practice, such observability enables early detection of silent failures and performance degradation, which are common challenges in LLM-enabled pipelines. Importantly, the architecture supports continuous improvement by enabling data engineers and analysts to iteratively refine prompts based on empirical performance evidence, rather than intuition alone.

From an architectural perspective, the results indicate that prompt governance can be implemented without disrupting existing data platform abstractions. Integration with lakehouse architectures, orchestration engines, and metadata catalogs was achieved using familiar mechanisms such as metadata hooks and execution interceptors. This suggests that the proposed reference architecture is not only conceptually sound but also practically feasible for organizations with mature data engineering ecosystems. The modular design further allows incremental adoption, enabling teams to prioritize high-risk or high-impact workflows before expanding governance coverage across the platform.

Despite these strengths, several limitations should be acknowledged. The evaluation was scenario-based and conceptual, focusing on logical completeness and architectural feasibility rather than quantitative performance metrics. While the architecture supports monitoring and drift detection, the study did not empirically assess the effectiveness of specific drift thresholds or alerting strategies. Additionally, the architecture assumes the availability of robust metadata infrastructure, which may not be present in less mature organizations. Future work should explore lightweight implementations and migration strategies for environments with limited governance capabilities.

Another important consideration is the evolving nature of LLM technologies. As models become more autonomous and agentic, prompts may be generated or modified dynamically by other AI systems rather than by human operators. While the proposed architecture can accommodate such scenarios by treating AI-generated prompts as governed artifacts, additional research is needed to address accountability and validation in fully autonomous prompt generation. Similarly, integration with retrieval-augmented generation (RAG) systems introduces new lineage dimensions related to retrieved knowledge sources, which warrant deeper investigation.

In a broader context, this study contributes to the emerging field of LLMOps by extending established MLOps and DataOps principles to prompt-centric systems. The results reinforce the argument that responsible and scalable LLM adoption requires governance mechanisms that span the entire operational stack, from data and prompts to models and outputs. Prompt lineage should therefore be viewed as a foundational capability rather than an optional enhancement. As enterprises increasingly rely on LLMs to automate critical data workflows, the absence of such mechanisms may lead to escalating technical debt, compliance risks, and erosion of trust in AI-augmented systems.

In conclusion, the proposed reference architecture demonstrates that prompt lineage and governance can be systematically integrated into LLM-enabled data engineering environments, delivering tangible benefits in reproducibility, compliance, and operational resilience. While further empirical validation is required, the findings provide a strong conceptual and practical foundation for future research and industrial adoption.

CONCLUSION

This study addressed a critical and underexplored challenge in modern data engineering: the absence of systematic lineage and governance mechanisms for prompts in LLM-enabled workflows. As large language models become embedded within enterprise data platforms, prompts increasingly function as executable logic that shapes data transformations, analytical outputs, and decision-support processes. However, existing data governance, DataOps, and MLOps frameworks have largely overlooked prompts as first-class operational artifacts. To address this gap, this paper proposed a reference architecture that formalizes prompt lineage and governance within LLM-enabled data engineering environments.

The proposed architecture demonstrates that prompts can be effectively managed using principles already familiar to enterprise data systems, including version control, metadata management, lineage tracking, policy enforcement, and observability. By extending traditional lineage models to incorporate prompt versions, execution contexts, and model configurations, the architecture enables end-to-end traceability from raw data inputs to downstream outputs. This capability directly supports reproducibility, auditability, and root-cause analysis, which are essential for both operational reliability and regulatory compliance. The scenario-based evaluation further showed that prompt governance can be integrated into existing lakehouse platforms and orchestration frameworks with minimal disruption, reinforcing the practical feasibility of the approach.

Despite these contributions, several limitations should be acknowledged. First, the study adopted a design science and architectural evaluation approach rather than an empirical implementation-based assessment. As a result, the findings emphasize conceptual completeness and integration feasibility rather than quantitative performance metrics or cost analysis. Second, the evaluation relied on representative enterprise scenarios rather than real-world production deployments, which may exhibit additional complexity related to scale, organizational processes, and tooling heterogeneity. Third, the architecture assumes the presence of mature metadata and governance infrastructure, which may not be available in all organizational contexts, particularly in smaller or less regulated environments.

Future research should extend this work in several important directions. Empirical validation through prototype implementations and longitudinal case studies would provide deeper insights into performance overheads, operational benefits, and adoption challenges in real-world settings. Further work is also needed to explore automated testing, validation, and drift detection strategies specifically tailored to prompt-driven systems. As LLM-based agents become more autonomous, future architectures must address governance and accountability for AI-generated or self-modifying prompts. Additionally, deeper integration with retrieval-augmented generation and multimodal pipelines will require expanded lineage models that capture external knowledge sources and non-textual inputs.

In summary, this study positions prompt lineage and governance as foundational capabilities for scalable, trustworthy, and compliant LLM-enabled data engineering. By providing a structured reference architecture, it lays the groundwork for future research and industrial practice aimed at operationalizing LLMs with the same rigor traditionally applied to data and code.

REFERENCES

1. Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. Proc NAACL-HLT. 2019;4171–86.
2. Brown T, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, et al. Language models are few-shot learners. Adv Neural Inf Process Syst. 2020;33:1877–901.
3. Liu P, Yuan W, Fu J, Jiang Z, Hayashi H, Neubig G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Comput Surv. 2023;55(9):1–35.
4. Zhamak D, Snowflake Team. The rise of the data lakehouse. Commun ACM. 2022;65(5):44–50.
5. Armbrust M, Das T, Sun L, Yavuz B, Zhu S, Murthy A, et al. Delta Lake: High-performance ACID table storage over cloud object stores. Proc VLDB Endow. 2020;13(12):3411–24.
6. Ghodsi A, Zaharia M, Shenker S, Stoica I. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. CIDR. 2021.
7. Buneman P, Khanna S, Tan WC. Data provenance: Some basic issues. Proc FSTTCS. 2000;87–93.
8. Herschel M, Diestelkämper R, Ben Lahmar H. A survey on provenance: What for? What form? What from? VLDB J. 2017;26(6):881–906.
9. Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, et al. On the opportunities and risks of foundation models. arXiv. 2021;arXiv:2108.07258.
10. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-thought prompting elicits reasoning in large language models. Adv Neural Inf Process Syst. 2022;35:24824–37.

11. Wu S, Irsoy O, Lu S, Dabravolski V, Dredze M, Gehrmann S, et al. BLOOM: A 176B-parameter open-access multilingual language model. arXiv. 2022;arXiv:2211.05100.
12. European Commission. Proposal for a regulation laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Official Journal of the European Union. 2021.
13. Floridi L, Cowls J, Beltrametti M, Chatila R, Chazerand P, Dignum V, et al. AI4People—An ethical framework for a good AI society. Minds Mach. 2018;28(4):689–707.
14. Raji ID, Smart A, White RN, Mitchell M, Gebru T, Hutchinson B, et al. Closing the AI accountability gap. Proc FAT. 2020;33–44.
15. Erich F, Amershi S, Lewis G, Bacchelli A. A qualitative study of DevOps usage in practice. ICSE. 2017;1–11.
16. Sculley D, Holt G, Golovin D, Davydov E, Phillips T, Ebner D, et al. Hidden technical debt in machine learning systems. Adv Neural Inf Process Syst. 2015;2503–11.
17. Breck E, Cai S, Nielsen E, Salib M, Sculley D. The ML test score: A rubric for ML production readiness and technical debt reduction. Proc IEEE BigData. 2017;1123–32.
18. Mitchell M, Wu S, Zaldivar A, Barnes P, Vasserman L, Hutchinson B, et al. Model cards for model reporting. Proc FAT. 2019;220–9.
19. Gebru T, Morgenstern J, Vecchione B, Vaughan JW, Wallach H, Daumé H, et al. Datasheets for datasets. Commun ACM. 2021;64(12):86–92.
20. Amershi S, Begel A, Bird C, DeLine R, Gall H, Kamar E, et al. Software engineering for machine learning: A case study. Proc ICSE. 2019;291–300.