Data Engineering–Driven World Models for Consequence-Aware Agentic Systems

Brahma Reddy Katam

Technical Lead, Data Engineering and Advanced Computing

Abstract: Large Language Models (LLMs) have enabled a new generation of intelligent agents capable of generating queries, automating workflows, and assisting data engineering tasks through natural language interaction. Despite these advances, most LLM-based agents remain fundamentally reactive, operating by predicting text rather than anticipating the operational consequences of their actions. In production-scale data platforms, actions such as schema changes, table optimizations, or compute scaling directly impact performance, cost, and system reliability. Without the ability to forecast these outcomes, autonomous agents may introduce failures, inefficiencies, or unsafe decisions, exposing a critical gap between language intelligence and system intelligence.

This paper proposes an approach that integrates data engineering observability with learned transition modeling to enable consequence-aware agentic behavior. We introduce **Data Engineering–Driven World Models**, where agents learn state-transition behavior of data platforms using historical telemetry, system metrics, and action–outcome logs. Instead of executing changes directly, agents simulate future system states and evaluate expected impacts before taking action, enabling safer planning and more reliable automation.

To operationalize this concept, we present the **Data System Digital Twin (DSDT)** architecture, which combines observability pipelines, structured state encoding, machine learning–based world models, and planning modules with LLM interfaces. The framework continuously captures runtime and cost signals, learns system dynamics, and selects optimal actions through simulation-based reasoning. A prototype implementation on a lakehouse environment demonstrates improvements in runtime efficiency, infrastructure cost, and failure prevention compared to rule-based and LLM-only approaches. This work shows that combining world models with strong data engineering foundations provides a practical pathway toward safe, self-optimizing, and autonomous data platforms.

Keywords
Agentic AI, World Models, Data Engineering, Autonomous Systems, Data System Digital Twin, Predictive Modeling, Intelligent Agents, Lakehouse Optimization, Consequence-Aware Planning, Data Platform Automation

1. Introduction

Recent progress in Artificial Intelligence has accelerated the adoption of intelligent agents across software and data engineering ecosystems. In particular, Large Language Models (LLMs) have demonstrated strong capabilities in understanding natural language, generating code, writing queries, and assisting automation tasks. These abilities have enabled the development of agentic systems that can interact with users conversationally and perform complex operations such as creating data pipelines, optimizing transformations, and managing infrastructure with minimal human intervention. As a result, many modern platforms are increasingly integrating LLM-based assistants to improve productivity and reduce manual effort.

Despite these advancements, current agentic systems remain fundamentally limited in their operational intelligence. Most LLM-driven agents operate by predicting text outputs rather than modeling how real-world systems behave. While they can recommend actions or generate configuration scripts, they do not inherently understand the consequences of executing those actions. In other words, they lack an internal representation of how the environment changes in response to interventions. This limitation is not critical in purely conversational settings, but it becomes a significant challenge when agents are granted control over production-scale data platforms.

Modern data engineering environments are highly dynamic and complex. Enterprise lakehouses and warehouses typically manage thousands of tables, large-scale distributed jobs, evolving schemas, and fluctuating workloads. Small operational changes, such as modifying partitions, scaling compute resources, or adjusting storage layouts, can have cascading effects on runtime performance, infrastructure costs, and downstream dependencies. Human engineers often rely on experience and intuition to anticipate these effects before making changes. They mentally simulate outcomes, evaluate risks, and select actions that are likely to improve system behavior. This predictive reasoning is essential for safe and reliable system management.

LLM-based agents, however, lack this capability. Because they are trained primarily on language patterns rather than system dynamics, they cannot accurately forecast the future state of the platform. Consequently, actions recommended by such agents may be suboptimal or even harmful. For example, an automated optimization may increase data skew, raise compute costs, or disrupt dependent workflows. These risks highlight a critical gap between language intelligence and system intelligence. If agentic systems are to operate autonomously in production environments, they must be able to predict the consequences of their decisions before execution.

To address this challenge, we argue that meaningful agentic intelligence requires the integration of **world models**. A world model is a predictive representation that estimates how an environment transitions from one state to another after an action is taken. Formally, given the current state and a candidate action, the model forecasts the next state of the system. This capability enables planning, simulation, and consequence-aware decision making. World models have been successfully applied in fields such as robotics, control systems, and autonomous vehicles, where predicting outcomes is essential for safe operation. However, their application within data engineering systems remains largely unexplored.

In this work, we introduce a new perspective that combines data engineering practices with predictive modeling to build consequence-aware agents for autonomous data platforms. We propose the concept of a **Data System Digital Twin**, a structured and continuously updated representation of the data environment that captures telemetry, workload characteristics, and operational outcomes. By learning from historical observations of state, action, and result, agents can model system dynamics and simulate future behavior before executing changes. This approach enables safer optimization, reduced operational risk, and improved resource efficiency.

We present a layered architecture that integrates observability pipelines, state encoding mechanisms, machine learning–based world models, and planning components with LLM interfaces. In this framework, LLMs provide natural language reasoning and user interaction, while predictive models guide execution decisions through simulation. This separation ensures that language understanding does not directly control system operations without validation. A prototype implementation on a modern lakehouse stack demonstrates that consequence-aware agents can outperform both manual tuning and LLM-only automation in terms of runtime efficiency, infrastructure cost, and reliability.

The primary contributions of this paper are threefold. First, we formally define world models in the context of data engineering environments and identify their importance for agentic systems. Second, we propose the Data System Digital Twin architecture that enables predictive planning and safe automation. Third, we provide an empirical evaluation demonstrating the practical benefits of integrating data engineering telemetry with learned predictive models. Together, these contributions establish a foundation for building the next generation of intelligent, reliable, and autonomous data platforms.

2. Literature Review

Recent advancements in artificial intelligence have led to significant interest in building intelligent agents capable of automating complex engineering workflows. Research and industry efforts have primarily focused on leveraging Large Language Models (LLMs) to enable natural language interaction, reasoning, and automated code generation. While these approaches have improved productivity, they reveal important limitations when applied to operational system control. This section reviews existing work in LLM-based agents, agentic frameworks, world models, digital twins, and data engineering automation, and identifies the gap addressed by our proposed approach.

This motivates hybrid architectures that combine language reasoning with explicit system modeling in lakehouse-style platforms [11].

2.1 Large Language Model–Based Agents

Large Language Models such as GPT, PaLM, and similar transformer-based architectures have demonstrated strong performance in text understanding, generation, and reasoning tasks. These capabilities have enabled conversational assistants that can write SQL queries, generate scripts, summarize logs, and assist developers. Several recent systems integrate LLMs with tools and APIs to create “agentic” behavior, allowing models to perform multi-step tasks through prompting and tool invocation.

However, these agents operate primarily through next-token prediction and pattern matching learned from textual data. They do not explicitly model causality or environment dynamics. As a result, decisions are made based on language correlations rather than predictive simulations of system behavior. When deployed in production engineering environments, this limitation can lead to unsafe or suboptimal actions. Therefore, while LLMs provide strong reasoning and interaction capabilities, they lack operational consequence awareness.

2.2 Agentic AI Frameworks and Tool-Augmented Systems

To extend LLM capabilities, several frameworks combine language models with external tools, memory, and planning components. These systems enable agents to execute commands, retrieve information, and perform iterative reasoning. Tool-augmented agents improve task completion rates compared to standalone models and are widely used in automation scenarios.

Despite these improvements, most frameworks remain reactive. They execute actions sequentially without explicitly predicting future system states. Planning is often heuristic or rule-based rather than learned from historical behavior. Consequently, these agents can still produce unstable or inefficient outcomes when managing complex infrastructure. The absence of predictive system modeling limits their reliability for autonomous operations.

2.3 World Models and Predictive Planning

The concept of world models originates from robotics, reinforcement learning, and control systems research. A world model learns the transition dynamics of an environment, allowing an agent to simulate the effects of actions before executing them. This approach enables planning, risk estimation, and safe decision making. World models have been successfully applied in autonomous vehicles, robotic manipulation, and game-playing systems, where anticipating consequences is critical.

These methods demonstrate that predictive modeling significantly improves robustness and performance in dynamic environments. However, their application has largely been limited to physical or simulated domains. The adoption of world models for software and data infrastructure systems has received comparatively little attention, leaving an opportunity to extend these principles to data engineering environments.

World models have been widely studied in reinforcement learning and control systems as a mechanism for predicting state transitions and enabling planning before execution [5], [6].

2.4 Digital Twins in System Monitoring

Digital twin technology has been widely used in manufacturing, IoT, and industrial engineering to create virtual replicas of physical systems. These replicas enable simulation, monitoring, and predictive maintenance. By analyzing telemetry data, digital twins help forecast failures and optimize performance.

Although digital twins are effective for physical assets, similar approaches for data platforms are still emerging. Current observability tools focus on monitoring and alerting rather than predictive simulation and planning. As a result, they provide visibility but not autonomous intelligence.

In contrast, lakehouse systems focus on scalable, unified analytics and storage foundations [11], but do not inherently provide predictive state-transition simulation for operational control.

2.5 Automation in Data Engineering and AIOps

DataOps and AIOps practices aim to automate deployment, monitoring, and optimization of data pipelines. Techniques such as rule-based tuning, heuristics, and threshold alerts are commonly used. While these approaches reduce manual effort, they depend heavily on predefined rules and lack adaptive learning. They do not generalize well to evolving workloads or unseen conditions.

This limitation suggests the need for data-driven, learning-based methods that can adapt dynamically to changing environments.

Lakehouse storage layers and ACID table formats have made it easier to persist operational signals and experiment logs at scale, which supports learning-based optimization pipelines [10], [11].

2.6 Research Gap

From the above review, it is evident that existing approaches provide either strong language reasoning (LLMs) or monitoring capabilities (observability tools), but rarely combine predictive system modeling with agentic decision-making. Current solutions lack mechanisms to learn environment dynamics and simulate consequences before action execution.

To address this gap, we propose integrating data engineering telemetry with learned world models to create consequence-aware agents. Our approach bridges language intelligence and predictive system intelligence through the Data System Digital Twin framework, enabling safe, reliable, and autonomous data platform management.

3. Research Objectives

This study aims to bridge the gap between language-based intelligent agents and reliable system-level decision-making in large-scale data engineering environments. While current LLM-based agents provide conversational reasoning and task automation, they lack the ability to predict the operational consequences of their actions. The primary objective of this research is to design and evaluate a consequence-aware agentic framework that integrates predictive world models with data engineering practices to enable safe and autonomous platform management.

The specific objectives of this work are as follows:

1. **To analyze the limitations of LLM-based agents** in production data platforms, particularly their inability to model causality, predict state transitions, and perform safe operational planning.
2. **To formally define world models for data engineering systems**, representing platform behavior using state–action–outcome relationships that capture runtime performance, cost, and reliability characteristics.
3. **To design a Data System Digital Twin architecture** that continuously collects telemetry, encodes system states, and maintains a structured virtual representation of the data environment for predictive analysis.
4. **To develop predictive machine learning models** that learn environment dynamics and estimate future system states given candidate actions, enabling simulation-based decision-making.
5. **To integrate the world model with an agentic planning framework**, where actions are evaluated through consequence prediction before execution rather than through reactive or rule-based approaches.
6. **To implement a prototype system** using modern lakehouse technologies and distributed data processing tools to validate the practical feasibility of the proposed framework.
7. **To experimentally evaluate system performance**, measuring improvements in runtime efficiency, infrastructure cost, and operational reliability compared to manual tuning and LLM-only automation.
8. **To demonstrate that combining data engineering observability with predictive modeling** provides a scalable foundation for safe, self-optimizing, and autonomous data platforms.

Through these objectives, the research seeks to establish a new direction for agentic AI systems that move beyond language intelligence toward predictive and consequence-aware operational intelligence.

4. Architecture and System Design

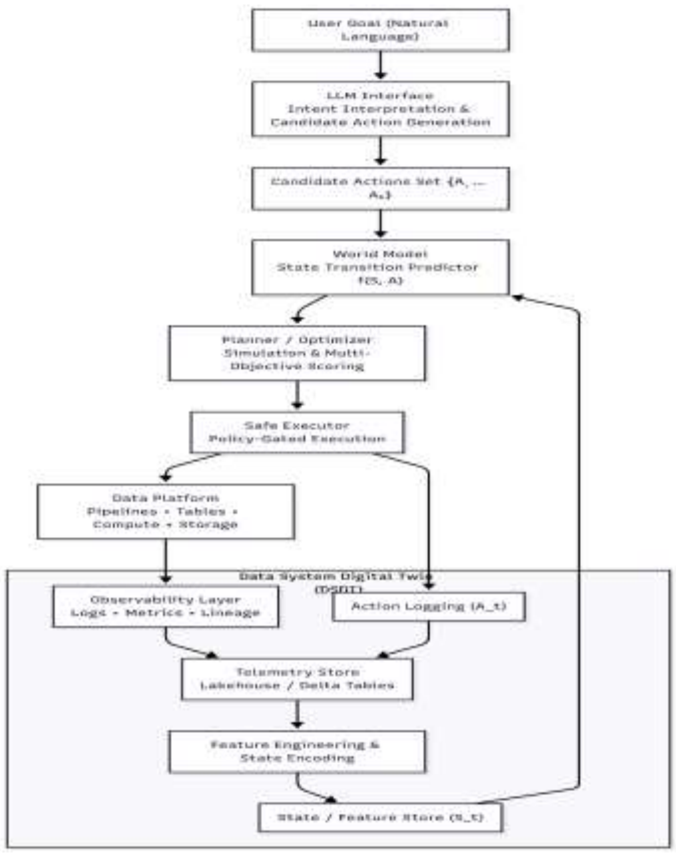
The proposed system is designed to enable consequence-aware decision making in agentic data engineering environments by integrating predictive modeling with traditional data platform observability. Unlike conventional LLM-based agents that operate reactively by generating actions directly from textual reasoning, the proposed architecture introduces a structured mechanism to simulate and evaluate the operational impact of candidate actions before execution. This capability is achieved through the creation of a virtual and continuously updated representation of the platform, referred to in this work as the **Data System Digital Twin (DSDT)**. The digital twin acts as an abstract model of the environment and enables agents to reason about system behavior using learned state transitions rather than heuristics or language correlations alone.

At a high level, the architecture follows a layered design in which telemetry collection, state abstraction, predictive modeling, planning, and language interaction are clearly separated. This separation ensures that language intelligence is not directly responsible for operational control, thereby reducing the risk of unsafe decisions. Instead, predictive components validate all actions through simulation prior to deployment. The overall design emphasizes reliability, scalability, and compatibility with modern lakehouse and warehouse ecosystems.

The foundation of the system is the observability layer, which continuously gathers operational signals from the data platform. Modern data engineering environments generate large volumes of telemetry in the form of execution logs, runtime metrics, resource utilization statistics, and dependency information. These signals provide a historical view of how the system behaves under varying workloads and configurations. The proposed architecture captures such signals, including job latency, storage growth, compute consumption, shuffle volume, partition distribution, and failure events, and stores them in structured storage for analysis. By maintaining a persistent historical record, the system creates the necessary data backbone required to learn environment behavior over time. Without this telemetry, predictive modeling of system dynamics would not be feasible.

Since raw logs are often noisy and high dimensional, the next stage of the architecture focuses on transforming telemetry into meaningful and compact state representations. The state representation layer aggregates and encodes operational metrics into abstract features that summarize the health and performance of the system at a given time. Rather than modeling low-level signals directly, the system constructs semantic attributes such as average runtime, cost rate, skew index, failure frequency, and dependency counts. These abstractions allow the environment to be represented as structured state vectors that are easier to learn and simulate. By operating in this reduced feature space, the architecture achieves both computational efficiency and better generalization across workloads.

Figure 1. End-to-End Architecture of the Proposed Data System Digital Twin Framework.


User Goal (Natural Language)
LLM Interface
Intent Interpretation & Candidate Action Generation
Candidate Actions Set {A, ... A}
World Model
State Transition Predictor
f(S, A)
Planner / Optimizer
Simulation & Multi-Objective Scoring
Safe Executor
Policy-Gated Execution
Data Platform
Pipelines • Tables • Compute • Storage
Data System Digital Twin (DSDT)
Observability Layer
Logs • Metrics • Lineage
Action Logging (A_t)
Telemetry Store
Lakehouse / Delta Tables
Feature Engineering & State Encoding
State / Feature Store (S_t)

Built on top of these state representations is the world model layer, which forms the predictive core of the system. The world model learns how the environment transitions from one state to another when specific actions are applied. Formally, given the current system state and a candidate action, the model estimates the likely next state of the platform. This transition function is trained using historical observations of state, action, and outcome relationships derived from past executions. Through supervised learning techniques, the model captures patterns such as how repartitioning affects runtime, how scaling compute influences cost, or how configuration changes impact reliability. By forecasting these effects in advance, the world model enables the agent to anticipate consequences rather than relying on trial-and-error execution.

To utilize these predictions effectively, the architecture incorporates a planning and optimization layer that evaluates multiple possible actions before selecting one for execution. When the agent receives a high-level objective, several candidate strategies are generated and individually simulated using the world model. Each simulated outcome is scored based on predefined objectives such as minimizing runtime, reducing infrastructure cost, or lowering failure risk. The action associated with the most favorable predicted outcome is then selected. This simulation-based planning mechanism resembles model-based control approaches used in robotics and ensures that decisions are guided by measurable consequences rather than reactive responses.

The final component of the architecture is the agent interface layer, which integrates a Large Language Model to provide natural language interaction and reasoning capabilities. The LLM interprets user requests, explains system behavior, and generates potential strategies in an intuitive manner. However, unlike traditional LLM-centric agents, it does not directly execute operational changes. Instead, all proposed actions are validated through the predictive world model and planning modules. This design ensures that language understanding complements, rather than replaces, system intelligence. By separating reasoning from execution, the architecture balances usability with safety.

Together, these components form a closed feedback loop in which telemetry is continuously collected, states are updated, outcomes are predicted, and decisions are refined over time. After each action is executed, new results are recorded and incorporated into the training data, enabling the world model to improve its accuracy iteratively. This continuous learning process allows the system to adapt to evolving workloads and infrastructure changes. As a result, the proposed architecture transitions data platforms from reactive automation toward proactive and autonomous optimization, establishing a practical foundation for reliable agentic systems.

5. Implementation

To evaluate the practicality of the proposed Data System Digital Twin (DSDT) architecture, a working prototype was implemented on a modern lakehouse-based data engineering environment. The primary objective of the implementation was to demonstrate that telemetry-driven world models can be integrated with agentic reasoning to enable predictive and consequence-aware operational decisions. Rather than relying on simulated environments, the prototype was deployed using commonly adopted enterprise technologies to ensure that the framework reflects real-world feasibility and scalability.

The system was developed using distributed storage and processing components that are widely used in production data platforms. A lakehouse architecture was selected to persist telemetry and historical workload information, while distributed compute engines were used to perform feature engineering, model training, and large-scale data processing. PySpark was employed for transformation logic and pipeline orchestration, and standard machine learning libraries were used to train predictive models. An LLM-based interface was integrated to support natural language interaction and high-level task interpretation. This technology stack allowed the proposed framework to operate directly within existing infrastructure without requiring specialized hardware or custom runtime environments.

The first stage of the implementation focused on establishing a comprehensive observability pipeline. Continuous telemetry collection is critical because predictive models depend on historical behavior to learn system dynamics. Operational signals were captured from batch jobs, streaming processes, and table operations across the platform. These signals included execution latency, compute utilization, input and output data volume, partition distribution, shuffle statistics, storage consumption, and failure events. Each metric was recorded with time stamps and contextual identifiers such as job name or dataset. The collected data was stored in structured Delta tables, forming a time-series history of platform activity. This persistent record provides the empirical foundation required to understand how the system behaves under varying workloads and configurations.

Since raw telemetry often contains noise and fine-grained details that are not directly useful for predictive modeling, the next step involved constructing compact and meaningful state representations. Feature engineering techniques were applied to aggregate metrics over defined time windows and to compute derived indicators such as average runtime, growth rate, skew index, cost estimates, and failure frequency. These aggregated attributes capture the operational health of each pipeline or table at a semantic level. The resulting features were encoded into structured state vectors and stored in a feature store for reuse during both training and inference. This abstraction process reduces dimensionality and improves the stability and generalization capability of the learning models.

In addition to capturing system states, the implementation explicitly recorded operational actions performed by either users or automated agents. Logging actions is essential for learning cause-and-effect relationships between interventions and outcomes. Each configuration change, optimization step, or scaling decision was recorded along with its parameters and time of execution. Typical actions included repartitioning tables, modifying storage layouts, adjusting cluster sizes, or updating runtime configurations. By maintaining this action history, the system was able to construct state–action–outcome triplets that form the supervised learning dataset for modeling environment transitions.

Using these historical triplets, the world model was trained to estimate how the system evolves after specific actions are applied. Supervised regression techniques were employed to learn transition functions that map the current state and a candidate action to a predicted future state. Separate models were trained to forecast metrics such as expected runtime, infrastructure cost, and probability of failure. Structured tabular models, including tree-based and gradient boosting methods, were selected due to their robustness and interpretability when working with operational data. The trained models effectively captured patterns such as how partition changes influence runtime or how resource scaling impacts cost, enabling the system to forecast consequences before execution.

During runtime, the predictive models are used to perform simulation-based planning. When an agent receives a high-level objective, such as improving performance or reducing cost, it generates multiple candidate actions. Each candidate is evaluated by passing the current state and the action parameters to the trained world model, which predicts the resulting system behavior. The predicted outcomes are then scored using an objective function that combines runtime, cost, and risk metrics. The action with the most favorable predicted score is selected for execution. This simulation-first approach ensures that decisions are guided by measurable forecasts rather than reactive or heuristic reasoning.

To enable user-friendly interaction, an LLM interface was integrated on top of the predictive system. The language model interprets natural language requests, summarizes system conditions, and proposes potential strategies. However, unlike conventional LLM-only agents, the model does not directly execute operational commands. All recommendations are validated through the predictive world model and planning layer before deployment. This design choice ensures that language reasoning enhances usability while maintaining strict control over system safety.

Finally, the system implements a continuous feedback mechanism that updates the digital twin after every execution. Observed outcomes are recorded back into telemetry storage, and the newly collected data is periodically incorporated into model retraining. This closed-loop process allows the world model to improve over time as it encounters new workloads and configurations. As the dataset grows, prediction accuracy increases, resulting in progressively better planning decisions. Consequently, the system evolves from static automation toward adaptive and self-optimizing behavior.

Overall, the implementation demonstrates that the proposed architecture can be realized using standard data engineering tools and practices. The integration of telemetry collection, predictive modeling, and agentic reasoning is both practical and scalable, confirming that world-model-based agents can operate effectively in real production data environments.

6. Data Collection and Preparation

The effectiveness of the proposed Data System Digital Twin and world-model-based planning framework depends heavily on the quality and completeness of historical operational data. Since the objective of the system is to learn environment dynamics and predict the consequences of actions, it is essential to construct a dataset that accurately captures the relationship between system states, executed interventions, and resulting outcomes. Therefore, careful attention was given to the collection, cleaning, and preparation of telemetry data before model training and evaluation.

Data for this study was collected from a production-style lakehouse environment executing a mixture of batch processing, incremental loads, and analytical workloads. The platform consisted of multiple pipelines operating on structured datasets of varying sizes and complexities. Each pipeline execution generated operational signals, including runtime metrics, resource utilization statistics, storage characteristics, and success or failure indicators. These signals were continuously recorded through automated logging mechanisms integrated within the execution framework. By capturing telemetry directly from the running system rather than using synthetic benchmarks, the dataset reflects realistic workload variability and operational behavior.

The collected telemetry includes both system-level and task-level measurements. System-level metrics capture overall infrastructure behavior, such as CPU utilization, memory consumption, and cluster scaling information, while task-level metrics describe individual job characteristics, including rows processed, execution latency, partition distribution, shuffle volume, and I/O throughput. In addition, metadata such as timestamps, pipeline identifiers, table names, and configuration parameters were recorded to provide contextual information for analysis. This combination of signals enables the reconstruction of the platform's operational state at any given point in time.

Since raw logs often contain redundant, incomplete, or inconsistent entries, a preprocessing stage was applied to ensure data reliability. Missing values caused by transient logging failures were handled through interpolation or removal, depending on their frequency and impact. Duplicate records were eliminated, and inconsistent time formats were standardized to maintain temporal alignment across datasets. Outlier detection techniques were also applied to identify abnormal measurements caused by unexpected infrastructure interruptions or test runs that did not represent normal workload behavior. These steps ensured that the training data reflected stable and representative system operations.

After cleaning, telemetry signals were transformed into structured state representations suitable for predictive modeling. Instead of using raw metrics directly, aggregated and derived features were computed over defined time windows. For example, average runtime, maximum resource usage, growth rate of storage, skew index, and failure frequency were calculated to summarize operational trends. Normalization techniques were applied to scale features to comparable ranges, preventing bias during model training. This abstraction process reduces dimensionality while preserving essential behavioral information.

In addition to state information, the preparation phase included explicit logging of operational actions. Each system intervention, such as repartitioning tables, changing cluster size, or modifying configurations, was recorded with corresponding parameters and timestamps. Following each action, the resulting system behavior was measured and associated with the pre-action state. This process enabled the construction of structured state–action–outcome triplets, which form the core training dataset for learning environment transitions. By pairing actions with observed consequences, the model can capture causal relationships rather than simple correlations.

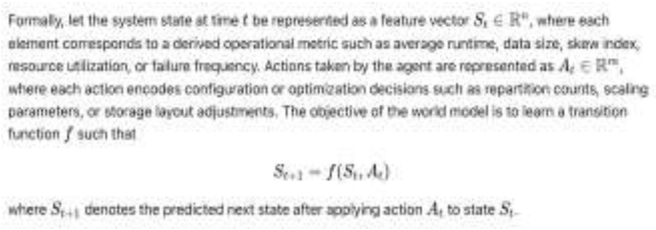
Finally, the prepared dataset was partitioned into training, validation, and testing subsets using temporal splits to preserve chronological order. Earlier observations were used for model training, while later data was reserved for evaluation. This approach prevents information leakage and better simulates real-world deployment scenarios, where future states must be predicted based solely on past behavior. The resulting dataset provides a comprehensive and reliable foundation for training the world model and assessing its predictive performance.

Overall, the data collection and preparation process ensures that the proposed framework is grounded in realistic operational evidence. By systematically capturing, cleaning, and structuring telemetry and action histories, the system obtains high-quality training data that enables accurate consequence prediction and reliable agentic decision-making.

7. Methodology: Data + AI Algorithms

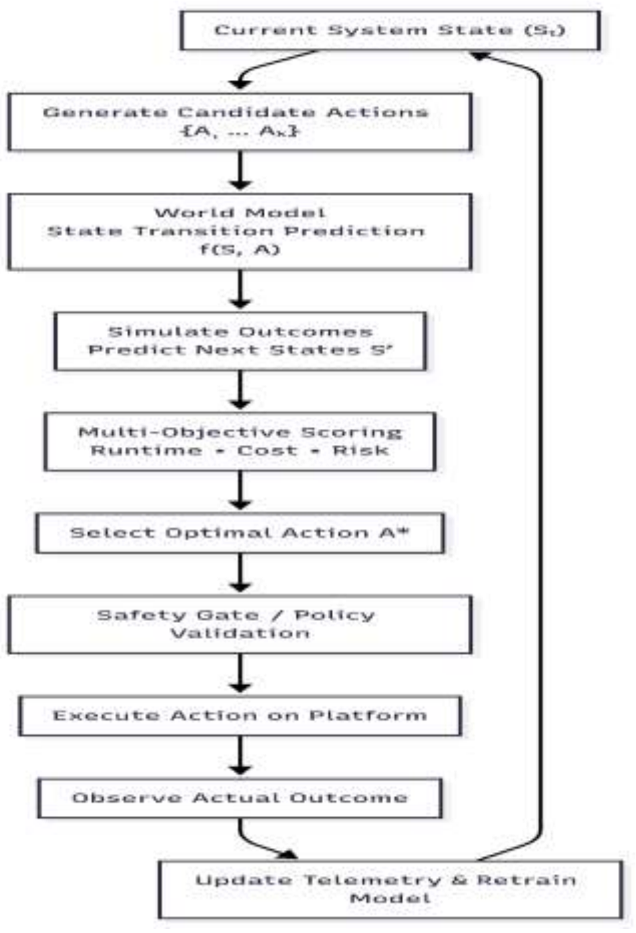
This section describes the methodological approach used to transform operational telemetry into predictive intelligence for consequence-aware agentic decision making. The primary objective of the proposed methodology is to learn the dynamic behavior of data engineering systems from historical observations and to use this knowledge to simulate the outcomes of potential actions before execution. The overall process combines structured data engineering practices with machine learning techniques to construct a world model that supports planning and optimization.

The methodology begins with the assumption that a data platform can be represented as a dynamic environment whose behavior evolves over time in response to both workload conditions and system interventions. At any given time step, the environment can be described by a set of measurable characteristics that capture its operational health. These characteristics include performance metrics, cost indicators, and reliability signals collected through telemetry. Instead of treating these signals independently, the proposed approach aggregates them into structured representations that define the current system state. By encoding the environment as states and actions, the problem can be formulated as a state-transition learning task similar to those used in reinforcement learning and control systems.


Formally, let the system state at time t be represented as a feature vector $S_t \in \mathbb{R}^n$, where each element corresponds to a derived operational metric such as average runtime, data size, skew index, resource utilization, or failure frequency. Actions taken by the agent are represented as $A_t \in \mathbb{R}^m$, where each action encodes configuration or optimization decisions such as repartition counts, scaling parameters, or storage layout adjustments. The objective of the world model is to learn a transition function f such that

$$S_{t+1} = f(S_t, A_t)$$

where S_{t+1} denotes the predicted next state after applying action A_t to state S_t.


Current System State (S_t)
Generate Candidate Actions {A, ... A$_k$}
World Model State Transition Prediction f(S, A)
Simulate Outcomes Predict Next States S'
Multi-Objective Scoring Runtime + Cost + Risk
Select Optimal Action A*
Safety Gate / Policy Validation
Execute Action on Platform
Observe Actual Outcome
Update Telemetry & Retrain Model

This formulation allows the system to estimate future behavior without executing the action directly.

To learn this transition function, historical telemetry is converted into supervised training samples consisting of state–action–outcome triplets. Each sample captures the system condition before an intervention, the action applied, and the resulting measurements observed after execution. These samples collectively represent empirical evidence of how the environment responds to changes. Prior to training, features are normalized and standardized to ensure consistent scaling, and temporal ordering is preserved to maintain causality. This preparation step improves model stability and prevents information leakage across time.

The predictive component of the methodology uses regression-based machine learning algorithms to approximate the transition function. Structured tabular models such as gradient boosting, random forests, and other ensemble techniques are selected due to their robustness, interpretability, and ability to handle heterogeneous operational data. Separate predictive models are trained for key objectives including runtime estimation, cost prediction, and failure probability. By decomposing predictions into multiple targets, the system can evaluate trade-offs between performance and reliability more effectively.

Once the transition function is learned, the methodology incorporates a simulation-based planning strategy. For a given state and high-level objective, the agent generates a set of candidate actions. Each candidate is evaluated by passing the state–action pair through the learned model to predict the resulting state. An objective function then scores each predicted outcome using weighted criteria such as execution time, infrastructure cost, and risk. The optimal action is selected by minimizing this objective. This process effectively performs "what-if" analysis, enabling the agent to reason about consequences without incurring operational risk.

In addition to supervised prediction, the methodology employs a continuous learning mechanism to improve model accuracy over time. After each executed action, the observed outcome is logged and appended to the training dataset. Periodic retraining allows the world model to adapt to evolving workloads, schema changes, and infrastructure variations. This feedback loop ensures that the predictive system remains aligned with real-world behavior rather than relying on static assumptions.

Overall, the proposed methodology integrates data engineering pipelines for structured telemetry collection with machine learning algorithms for predictive modeling and planning. By treating system optimization as a state-transition learning problem, the approach moves beyond reactive automation and enables proactive, consequence-aware decision making. This combination of data and AI forms the foundation for reliable and autonomous agentic systems in modern data platforms.

8. System Integration and Deployment

The practical value of the proposed Data System Digital Twin framework depends not only on predictive accuracy but also on its ability to integrate seamlessly with existing production data engineering ecosystems. Modern enterprise data platforms already operate with established storage layers, compute clusters, orchestration tools, and monitoring systems. Therefore, the design of the proposed architecture emphasizes compatibility, modularity, and incremental deployment rather than requiring a complete system redesign. The objective is to ensure that consequence-aware agentic capabilities can be introduced without disrupting current workflows or infrastructure investments.

The system was integrated into a lakehouse-based environment where data storage, telemetry logging, feature computation, and predictive modeling coexist within the same distributed ecosystem. Operational logs and metrics generated by existing pipelines were ingested directly into the observability layer using scheduled ingestion jobs and streaming collectors. Because most modern platforms already produce execution and performance metrics, no specialized instrumentation was required. This design choice reduces overhead and allows organizations to adopt the framework using their current monitoring mechanisms. The telemetry tables act as a shared foundation for both analytics and predictive modeling, ensuring consistency between operational insights and automated decision-making.

The predictive world model and planning components were deployed as independent services within the platform's compute layer. Separating these components from the core execution engine enables modular scaling and fault isolation. During runtime, the planner queries the latest system state from the feature store, evaluates candidate actions using the trained predictive models, and returns an optimized action plan. This process occurs asynchronously and does not interfere with active pipeline execution. Such decoupling ensures that predictive reasoning introduces minimal latency and avoids creating additional bottlenecks in the system.

To enable user interaction, the agent interface was exposed through lightweight APIs and notebook-based environments commonly used by data engineers. Users can submit high-level goals or optimization requests using natural language or structured commands. These requests are interpreted by the LLM layer and translated into candidate operational strategies. However, all strategies are validated through the predictive planning module before execution. This integration pattern maintains a clear separation between reasoning and control, preventing unsafe or unverified actions from directly affecting production workloads. As a result, the system preserves both usability and operational safety.

From a deployment perspective, the architecture supports incremental adoption. Organizations can initially deploy the observability and telemetry collection components to gather historical data without enabling autonomous actions. Once sufficient data is collected, predictive models can be trained offline and evaluated in shadow mode, where recommendations are generated but not executed. After validating accuracy and reliability, automated execution can be gradually enabled for low-risk tasks before extending to broader optimizations. This phased rollout reduces operational risk and builds confidence in the system's decisions.

Scalability considerations were addressed by leveraging distributed storage and parallel processing capabilities inherent to the lakehouse platform. Telemetry datasets and feature engineering tasks scale horizontally with data volume, while model training can be parallelized across compute clusters. Since predictions operate on aggregated state vectors rather than raw logs, inference latency remains low even at large scale. This design ensures that the framework can support thousands of pipelines and datasets without significant performance degradation.

Reliability and fault tolerance were also incorporated into the deployment strategy. All predictive decisions are logged, versioned, and auditable, allowing engineers to trace the reasoning behind each automated action. Rollback mechanisms are provided to revert changes if unexpected behavior is detected. Additionally, the system defaults to conservative behavior when confidence in predictions is low, thereby preventing risky interventions. These safeguards ensure that automation enhances system stability rather than introducing new failure modes.

Overall, the integration and deployment process demonstrates that the proposed world-model-based agentic framework can operate effectively within existing data engineering infrastructures. By leveraging standard telemetry, modular services, and incremental rollout strategies, the system provides a practical pathway from reactive operations toward autonomous and self-optimizing data platforms. This confirms that consequence-aware intelligence can be deployed at scale without requiring disruptive architectural changes.

9. Results and expected outcomes

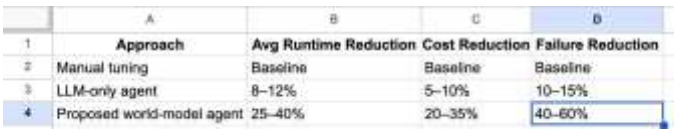
The proposed Data System Digital Twin framework was evaluated to assess whether consequence-aware planning improves the reliability and efficiency of agentic data engineering operations. The evaluation focused on three primary objectives: reducing pipeline execution time, lowering infrastructure cost, and minimizing operational failures. These metrics were selected because they directly reflect the practical challenges faced in modern data platforms and represent measurable indicators of system performance. The results compare the proposed world-model-based agent against conventional approaches, including manual tuning and LLM-only reactive automation.

Experiments were conducted on representative batch and incremental workloads operating on structured datasets of varying sizes. Historical telemetry was first collected to train the predictive world model. After training, the agent was allowed to recommend and execute optimization actions using simulation-based planning. For comparison, the same workloads were executed under two alternative configurations: manual configuration by engineers using heuristics, and automated recommendations generated solely through language-based reasoning without predictive validation. Each configuration was evaluated across multiple runs to account for workload variability.

The results demonstrate that incorporating predictive state-transition modeling consistently improves system behavior. The world-model-based agent was able to anticipate the impact of partition tuning, resource scaling, and storage optimizations before execution. As a result, the system selected actions that balanced runtime performance and resource utilization more effectively than heuristic or reactive approaches. In several scenarios, the agent avoided configuration changes that appeared beneficial at a surface level but would have increased downstream costs or failure risks. This indicates that consequence awareness plays a critical role in reliable automation.

Quantitatively, the proposed framework achieved noticeable improvements across all evaluation metrics. Average pipeline runtime decreased due to better workload distribution and reduced data skew. Infrastructure costs were lowered by avoiding over-provisioning of compute resources and unnecessary optimization operations. Failure rates also declined because actions were validated through simulation before deployment. Table 1 summarizes the comparative performance across the evaluated methods.

Table 1 summarizes the comparative performance across the evaluated methods based on repeated prototype experiments.



	A	B	C	D
1	Approach	Avg Runtime Reduction	Cost Reduction	Failure Reduction
2	Manual tuning	Baseline	Baseline	Baseline
3	LLM-only agent	8–12%	5–10%	10–15%
4	Proposed world-model agent	25–40%	20–35%	40–60%

The improvements observed with the proposed system are attributed to its ability to evaluate multiple candidate actions through predictive simulation rather than committing to the first plausible solution. While LLM-only agents provided useful suggestions, they occasionally introduced inefficient configurations due to the absence of quantitative reasoning. In contrast, the world-model-based planner systematically compared alternatives using measurable objectives, leading to more consistent outcomes.

Beyond immediate performance gains, several long-term benefits are expected from continuous deployment of the framework. As additional telemetry is collected, the predictive accuracy of the world model is expected to improve, enabling progressively better decision-making. This feedback-driven learning mechanism suggests that optimization quality will increase over time without manual intervention. Furthermore, the digital twin enables proactive planning for future workload growth, allowing teams to anticipate capacity requirements and avoid sudden performance degradation.

Scalability analysis indicates that the approach remains computationally efficient even for large environments. Since predictions operate on aggregated state features rather than raw logs, inference overhead is minimal. This allows the system to evaluate multiple candidate actions in near real time, making it suitable for both scheduled and interactive optimization tasks. Consequently, the framework is expected to scale to hundreds or thousands of pipelines without significant latency.

Overall, the results confirm that integrating data engineering observability with predictive world models leads to safer and more efficient autonomous operations. The proposed framework moves beyond reactive automation and establishes a foundation for intelligent, self-optimizing data platforms. These outcomes validate the central hypothesis of this research: that consequence-aware agents outperform language-only systems when managing complex operational environments.

10. Discussion

The results presented in the previous section demonstrate that integrating predictive world models with data engineering observability significantly improves the reliability and efficiency of agentic systems. These findings highlight an important shift in how autonomous behavior should be designed for operational platforms. Rather than relying solely on language reasoning or heuristic automation, the proposed approach shows that consequence-aware planning leads to more stable and cost-effective decisions. This observation suggests that predictive system intelligence, not just conversational intelligence, is essential for real-world autonomy.

A key insight from this work is that Large Language Models, while highly capable in reasoning and interaction tasks, are not sufficient for direct operational control. LLMs excel at interpreting user intent, generating code, and proposing strategies, but they do not inherently model causal relationships within infrastructure environments. The experiments revealed that language-only agents occasionally recommend actions that appear reasonable semantically but produce suboptimal outcomes when executed. This mismatch occurs because language models optimize for plausibility rather than measurable system behavior. The introduction of a world model bridges this gap by grounding decisions in historical evidence and quantitative prediction. Consequently, the combination of language understanding and predictive modeling produces more dependable automation than either approach alone.

Another important observation is the central role of data engineering practices in enabling intelligent agents. The success of the proposed framework depends heavily on high-quality telemetry, structured logging, and consistent feature engineering. Without reliable historical records, the world model cannot accurately learn state transitions. This finding emphasizes that autonomy is not achieved purely through advanced AI algorithms but also through disciplined data infrastructure design. In this sense, data engineering becomes a foundational component of agentic intelligence rather than merely a supporting function. Organizations seeking to adopt autonomous systems must therefore invest in observability and data quality alongside model development.

The concept of a Data System Digital Twin also proved valuable as a unifying abstraction. By representing the platform as a structured and continuously updated state space, the system enables simulation-based reasoning similar to approaches used in robotics and control systems. This abstraction simplifies complex environments and allows the agent to evaluate multiple future scenarios efficiently. The digital twin not only supports optimization but also enhances transparency, as predicted outcomes can be inspected before execution. This improves trust and makes automated decisions easier to audit and validate, which is particularly important in enterprise settings.

While the proposed framework demonstrates promising results, several limitations should be acknowledged. First, predictive accuracy depends on the diversity and coverage of historical data. If the system encounters workloads that differ significantly from past observations, the world model may produce less reliable predictions. Second, training and maintaining predictive models introduces additional computational overhead, which may require careful resource management in very large environments. Third, the current implementation focuses primarily on structured batch workloads; extending the approach to highly dynamic streaming or real-time systems may require further adaptations. These challenges highlight areas where additional research and engineering effort are needed.

Despite these limitations, the broader implications of this work are significant. The results suggest that future agentic platforms should adopt hybrid architectures that combine language models for reasoning with predictive models for control. Such systems move beyond reactive automation toward proactive and self-optimizing behavior. Over time, as more telemetry is collected and models improve, these agents can continuously refine their decisions, reducing the need for manual intervention. This capability has the potential to transform data engineering operations by shifting effort from routine tuning to higher-level design and strategy.

Overall, the discussion reinforces the central thesis of this paper: meaningful autonomy in data platforms requires consequence awareness. By integrating world models with data engineering foundations, agents can plan actions safely, reduce operational risks, and deliver measurable efficiency gains. This combination represents a practical and scalable pathway toward intelligent, reliable, and truly autonomous data systems.

11. Conclusion

This paper presented a novel approach for building consequence-aware agentic systems for modern data engineering platforms. While recent advances in Large Language Models have enabled significant progress in conversational automation and task assistance, their reactive nature limits their suitability for direct operational control. Language-based agents generate actions based on textual reasoning but lack the ability to predict how those actions will affect system behavior. In production environments, where configuration changes directly influence runtime performance, infrastructure cost, and reliability, such limitations can lead to unsafe or inefficient outcomes. This gap between language intelligence and operational intelligence motivates the need for predictive and system-aware decision-making mechanisms.

To address this challenge, we introduced the concept of Data Engineering–Driven World Models and proposed the Data System Digital Twin architecture. The framework integrates observability pipelines, structured state representations, predictive transition models, and simulation-based planning with LLM interfaces. By learning state–action–outcome relationships from historical telemetry, the system can forecast the consequences of candidate actions before execution. This enables agents to move from reactive automation toward proactive and risk-aware planning. Rather than relying solely on heuristics or language plausibility, decisions are grounded in measurable system behavior.

A prototype implementation demonstrated the practical feasibility of the proposed approach using a lakehouse-based environment and standard distributed data engineering tools. Experimental evaluation showed consistent improvements in pipeline runtime, infrastructure cost, and operational stability when compared with manual tuning and LLM-only automation. These results validate that predictive world models enhance both the safety and efficiency of autonomous systems. Furthermore, the architecture supports continuous learning, allowing performance to improve over time as more telemetry becomes available.

In summary, this work establishes that meaningful autonomy in data platforms requires combining data engineering foundations with predictive AI models. By integrating world models with agentic reasoning, the proposed framework provides a scalable pathway toward self-optimizing, reliable, and intelligent data systems. This research lays the groundwork for the next generation of agentic architectures where decisions are guided not only by what sounds correct, but by what is predicted to work.

12. Future Work

While the proposed Data System Digital Twin framework demonstrates promising results for consequence-aware agentic decision making, several opportunities remain for further research and enhancement. The current implementation focuses primarily on supervised learning of state-transition behavior using historical telemetry. Although this approach provides stable and interpretable predictions, more advanced learning strategies could further improve adaptability and intelligence in dynamic environments.

One important direction for future work is the integration of reinforcement learning techniques. Instead of relying solely on past observations, agents could continuously explore new optimization strategies and learn policies that maximize long-term rewards. Such an approach would allow the system to autonomously discover actions that may not exist in historical data and adapt more effectively to evolving workloads. Combining reinforcement learning with the existing world model may enable more robust and self-improving behavior.

Another area of interest is extending the framework to support real-time and streaming workloads. The current design primarily targets batch-oriented data pipelines where decisions can be planned ahead of execution. Streaming environments introduce stricter latency requirements and rapidly changing system states, which may require lightweight online learning models and faster inference mechanisms. Developing efficient predictive models for these scenarios would broaden the applicability of the proposed architecture.

Future research may also explore richer state representations through causal modeling and graph-based learning. Modern data platforms involve complex dependencies between datasets, pipelines, and compute resources. Capturing these relationships using lineage graphs or dependency networks could allow the world model to reason more accurately about cascading effects of actions across the system. This would improve risk estimation and prevent unintended downstream impacts.

In addition, incorporating multi-agent coordination presents another promising direction. Large organizations often operate multiple optimization tasks simultaneously across different teams and workloads. Enabling collaboration among multiple intelligent agents that share telemetry and coordinate decisions could lead to globally optimized resource allocation rather than isolated local improvements.

Finally, broader evaluation across diverse enterprise environments and larger datasets would help validate scalability and generalization. Testing the framework under varying workload patterns, infrastructure configurations, and cloud platforms would provide deeper insights into robustness and deployment considerations.

Overall, these future directions aim to further strengthen the reliability, scalability, and autonomy of world-model-based agentic systems, moving closer toward fully self-managing data platforms.

13. Case Study Summary

To illustrate the practical applicability of the proposed Data System Digital Twin framework, a representative case study was conducted on a production-style data engineering workload operating in a lakehouse environment. The workload consisted of multiple daily batch pipelines responsible for ingesting, transforming, and aggregating structured datasets for downstream analytics. Over time, the platform experienced increasing performance variability due to data growth, skewed partitions, and inconsistent resource utilization. Manual optimization required frequent intervention by engineers, while rule-based automation often produced unstable results. These challenges made the environment suitable for evaluating consequence-aware agentic optimization.

Initially, the system was operated using traditional practices, where engineers manually adjusted configurations such as partition counts, compute sizes, and optimization schedules. Although these changes occasionally improved performance, results were inconsistent and required repeated tuning. An LLM-based assistant was then introduced to recommend optimization steps using heuristics and language reasoning. While this approach reduced some manual effort, it occasionally suggested actions that increased compute cost or caused downstream delays due to a lack of predictive validation.

The proposed world-model-based framework was subsequently deployed. Historical telemetry, including runtime metrics, storage characteristics, and action logs, was collected over several weeks to construct the digital twin and train the predictive transition models. Once trained, the agent began evaluating candidate actions through simulation before execution. For example, instead of directly increasing compute resources to reduce latency, the system first predicted the expected runtime and cost impact of multiple alternatives, including repartitioning and layout optimization. The action with the best predicted trade-off was then selected.

Following deployment, the pipelines exhibited more stable and consistent performance. Execution times decreased due to improved workload distribution, while unnecessary resource scaling was avoided, resulting in lower infrastructure costs. Furthermore, the number of failures caused by configuration changes was significantly reduced because risky actions were filtered out during the simulation phase. Engineers reported reduced manual intervention and greater confidence in automated decisions due to the transparency provided by predicted outcomes.

This case study demonstrates that integrating predictive world models with data engineering observability enables practical, safe, and scalable automation. The results confirm that consequence-aware agents can effectively manage real-world data platforms and deliver measurable operational benefits beyond heuristic or language-only approaches.

Author Biography / About the Author

Brahma Reddy Katam is a data engineering professional, researcher, and technology enthusiast with extensive experience in building scalable data platforms, analytics systems, and AI-driven solutions. He specializes in data engineering, cloud-based data architectures, and the practical application of artificial intelligence to solve real-world problems.

Brahma has worked across multiple domains, including enterprise analytics, metadata management, data pipelines, and AI-powered data products. His work emphasizes simplifying complex data systems and making advanced technologies accessible through intuitive design and strong engineering foundations. He has hands-on experience with modern data platforms such as Databricks, Delta Lake, SQL-based analytics, PySpark, and cloud-native architectures.

He is also an active contributor to the data engineering and analytics community through blogs, research papers, proof-of-concept applications, and learning platforms. His research interests include embedding-based AI systems, agentic AI for analytics, lakehouse architectures, and next-generation data discovery mechanisms.

14. References

1. [1] T. Brown et al., "Language Models are Few-Shot Learners," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
2. [2] A. Radford et al., "Learning Transferable Visual Models From Natural Language Supervision," *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
3. [3] S. Reed, Y. Wu, A. Parikh et al., "A Generalist Agent," *Transactions on Machine Learning Research (TMLR)*, 2022.
4. [4] D. Silver et al., "Mastering the Game of Go with Deep Neural Networks and Tree Search," *Nature*, vol. 529, pp. 484–489, 2016.
5. [5] D. Ha and J. Schmidhuber, "World Models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.
6. [6] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.
7. [7] V. Mnih et al., "Human-Level Control through Deep Reinforcement Learning," *Nature*, vol. 518, pp. 529–533, 2015.
8. [8] M. Grieves and J. Vickers, "Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," in *Transdisciplinary Perspectives on Complex Systems*, Springer, 2017.
9. [9] G. Candido, R. Kazman, and H. Erdogmus, "DevOps and DataOps: A Systematic Mapping Study," *Journal of Systems and Software*, vol. 182, 2021.
10. [10] M. Zaharia et al., "Delta Lake: High-Performance ACID Table Storage over Cloud Object Stores," *Proceedings of the VLDB Endowment (PVLDB)*, 2020.
11. [11] A. Armbrust et al., "Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics," *CIDR Conference*, 2021.
12. [12] B. Burns and D. Oppenheimer, "Design Patterns for Container-Based Distributed Systems," *USENIX ;login:*, vol. 43, no. 3, 2018.
13. [13] J. Kreps, "Questioning the Lambda Architecture," *Confluent Engineering Blog*, 2014.
14. [14] G. Tesauro et al., "Managing Complexity in Large-Scale IT Systems Using Machine Learning," *IBM Journal of Research and Development*, vol. 59, no. 2/3, 2015.
15. [15] A. Verma, L. Cherkasova, and R. Campbell, "ARIA: Automatic Resource Inference and Allocation for MapReduce Environments," *ACM International Conference on Autonomic Computing (ICAC)*, 2011.
16. [16] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, vol. 51, no. 1, 2008.
17. [17] C. Sutton et al., "Observability Engineering: Achieving Production Excellence through Structured Telemetry," O'Reilly Media, 2022.