machines

MDPI

Article

A Conceptual Framework for a Latest Information-Maintaining Method Using Retrieval-Augmented Generation and a Large Language Model in Smart Manufacturing: Theoretical Approach and Performance Analysis

Hangseo Choi and Jongpil Jeong *

Department of Smart Factory Convergence, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon 16419, Gyeonggi-do, Republic of Korea; choinara@g.skku.edu
* Correspondence: jpjeong@skku.edu

Abstract: In the modern manufacturing environment, the ability to collect and refine data in real time to deliver high-quality data is increasingly important in maintaining a competitive advantage and operational efficiency. This paper proposes a conceptual architectural framework for the continuous knowledge updating of Retrieval-Augmented Generation-Large Language Model-based systems in a smart factory environment. The proposed framework provides theoretical models and validation methodologies, laying the groundwork for future practical implementations. Existing Retrieval-Augmented Generation-Large Language Model systems rely on static knowledge bases that are not able to effectively reflect new information in a real-time, changing manufacturing environment. The proposed framework design uses a data stream processing layer, a data integration layer, and a continuous learning layer as core components; in particular, the knowledge integration layer provides a mechanism for the efficient processing of real-time data and continuous learning. This study is significant in that it presents a mathematical model and a systematic verification methodology that can quantitatively predict the performance and scalability of the proposed architecture, thus providing practical design guidance for the implementation of Retrieval-Augmented Generation-Large Language Model systems in smart factory environments. This paper is organized as follows: Materials and Methods provides a detailed description of the architecture and methodology. Theoretical Analysis and Discussion covers the theoretical analysis and discussion, including performance prediction models, validation methodologies, and their practical implications. Finally, Conclusions summarizes the research findings and outlines directions for future work.

Keywords: continuous learning; knowledge integration; retrieval-augmented generation-large language model architecture; real-time data processing; smart factory; system performance modeling

check for updates

Academic Editor: Huosheng Hu

Received: 24 December 2024
Revised: 20 January 2025
Accepted: 23 January 2025
Published: 24 January 2025

Citation: Choi, H.; Jeong, J. A Conceptual Framework for a Latest Information-Maintaining Method Using Retrieval-Augmented Generation and a Large Language Model in Smart Manufacturing: Theoretical Approach and Performance Analysis. *Machines* **2025**, *13*, 94. https://doi.org/10.3390/machines13020094

1. Introduction

The digital transformation of manufacturing environments is increasing the complexity of large-scale data generated by Manufacturing Execution Systems (MESs) and equipment, which is becoming a key resource for operational optimization and decision-making. Smart manufacturing is an innovative paradigm that leverages advanced technologies such as Cyber-Physical Systems (CPSs), Big Data, Cloud Computing, Internet of Things (IoT), and Artificial Intelligence (AI) to make manufacturing systems intelligent, sustainable, and flexible. While traditional manufacturing systems focus on simply storing and managing data, smart manufacturing enables real-time decision-making and autonomous optimization and seeks to increase productivity and improve quality [1].

Especially in smart factory environments, real-time data-driven decision-making is becoming a key factor regarding competitive advantage. In particular, smart factories enhance competitiveness through the real-time data-driven optimization of manufacturing operations. They integrate a variety of data, such as production equipment, processes, and quality data, and use CPS and digital twin technologies to dynamically manage the production environment [2].

However, traditional RAG-LLM-based systems are limited by their reliance on static knowledge bases that cannot effectively reflect the dynamic data of a real-time changing manufacturing environment [3]. However, current smart manufacturing systems have limitations in processing large amounts of dynamic data and making real-time decisions. These problems are caused by the lack of incorporation of the latest data and the absence of continuously updating learning-based systems [4]. This limitation hinders the accuracy and speed of real-time, data-driven decision-making and limits the ability to adapt to changing production requirements and quality standards.

Dynamic change in manufacturing itself occurs in three main ways: physical, process, and data [5]. AI is emerging as a transformative technology in smart manufacturing, enabling real-time data analysis, predictive maintenance, and autonomous process optimization. By integrating AI with the proposed RAG-LLM framework, manufacturing systems can enhance their ability to adapt to changing conditions, predict anomalies, and continuously improve decision-making accuracy.

On the physical side, new production equipment or the relocation of existing equipment is a common occurrence, requiring changes to data structures and processing logic [6]. In addition, data collection systems are evolving with the continuous expansion and replacement of Internet of Things (IoT) sensors, and facility configurations are changing dynamically with the introduction of flexible production systems to accommodate product diversification.

On the process side, production optimization and the introduction of new technologies lead to continuous process improvement [7]. Quality management standards require regular updating due to changes in regulatory requirements and revisions to product specifications, and working standards are constantly being improved to increase efficiency and comply with safety regulations. On the data side, predictive maintenance patterns evolve with the accumulation of plant operating data, and the emergence of new defect types or changes in quality standards requires regular updates to quality prediction models. In addition, optimal process conditions are constantly changing due to changes in raw material properties or energy efficiency requirements [8].

These multi-dimensional changes cause a number of problems in today's manufacturing environment. Equipment anomaly detection systems often do not reflect the latest operating patterns, resulting in false positives and missing important anomalies [9]. On the quality control side, changes in standards are not reflected in a timely manner, leading to non-conformance errors that result in unnecessary rework or quality issues [10]. While existing studies have addressed various aspects of RAG-LLM systems in manufacturing, there remains a critical gap in maintaining knowledge freshness for real-time applications. Current approaches largely rely on static knowledge bases and periodic updates, which are inadequate for the dynamic nature of smart manufacturing environments.

This study addresses this gap by proposing a novel architectural framework that enables continuous knowledge updating through real-time data processing and integration. The key innovations of our approach include (1) a multi-layer architecture for real-time knowledge integration, (2) an efficient mechanism for maintaining data freshness, and (3) a quantitative performance prediction model. This research particularly benefits manufacturing engineers and system architects implementing AI-based decision support systems in smart factory environments.

In addition, equipment and personnel are used inefficiently, and organizational learning is hindered because resource allocation criteria or the knowledge of operators and new process improvements are not updated in a timely manner as the production environment changes [11]. To address these issues, this research proposes a novel architectural framework that focuses on continuous learning and real-time adaptation. The proposed architecture addresses these challenges by incorporating three key elements: the real-time processing of data streams for high-quality data provision, dynamic knowledge integration that maintains temporal and domain context, and the selective learning of changed data to enhance model performance [12].

This paper is organized as follows. Section 2 introduces the overall structure of the proposed architecture and provides a detailed design of each layer, focusing on the implementation of the data stream processing layer, the data integration layer, and the continuous learning layer, as well as their interactions. Section 3 presents the theoretical analysis and discussion, including performance prediction models, validation methodologies, and practical implications, such as implementation considerations and future research directions. Additionally, we analyze the practical applicability and limitations of the architecture, focusing on key performance metrics such as throughput, latency, and resource requirements. Finally, Section 4 summarizes the findings and presents the conclusions.

2. Materials and Methods

The architectural diagrams and process flows presented in this paper represent an original synthesis of various approaches and methodologies. While influenced by current technological trends, including RAG systems and modern open-source projects, these visualizations are not direct adaptations from any single source. Instead, they represent a novel integration of concepts specifically designed to address continuous knowledge updating in smart manufacturing environments. The proposed architecture and workflows reflect our interpretation and integration of various established concepts, adapted to meet the unique requirements of maintaining the latest knowledge in manufacturing systems.

2.1. RAG-LLM Framework

Existing systems use a periodic batch learning approach, which makes it difficult to process and learn from real-time data and quickly incorporate new domain knowledge. The architecture presented in this paper is based on continuous learning and real-time knowledge integration to flexibly respond to changing manufacturing environments. The architecture consists of three key layers for continuous knowledge updating in smart factory environments. The core layers (the data stream processing layer, the data integration layer, and the continuous learning layer) each perform independent functions but are organically connected to enable continuous learning and real-time data integration in the entire system.

The first data stream processing layer is responsible for efficiently ingesting and processing real-time data [13]. It collects real-time data from various sources, such as MES data streams, machine sensor data, and process logs, and ensures data quality through a multi-stage filtering pipeline [14]. In this layer, we designed a distributed processing architecture based on edge computing to increase the efficiency of data processing [15]. This systematic layer of data stream processing forms the basis for continuous knowledge updating and provides reliable data for the data integration and learning processes in the upper layers [16].

The second data integration layer is the heart of the system and is responsible for effectively integrating and managing the processed data into the knowledge base [17]. This layer ensures data consistency and reliability through three main mechanisms: temporal and domain context management, incremental data integration, and versioning [18]. The organic combination of temporal and domain context enables semantic knowledge management, while incremental data integration integrates new data into the existing knowledge base without rebuilding the entire database. A versioning system manages all changes to data entities, which have unique version numbers to enable comparison and rollback between versions [19].

The continuous learning layer uses three key mechanisms to continuously learn and improve the performance of the system: selective learning from data, incremental model refinement, and efficient knowledge transfer [20]. The selective learning mechanism can quantitatively evaluate the learning value of new data to avoid unnecessary learning and increase learning efficiency by focusing on the data. Progressive tuning updates only certain parts of the model by adjusting the learning rate of certain layers or modules or limiting the learning target. Finally, to increase the efficiency of knowledge transfer, selective distillation, stepwise distillation, and multi-task distillation transfer knowledge learned from large models to smaller models.

The three layers work together, and the data flow between the layers is asynchronous to ensure the overall responsiveness and scalability of the system. In addition, each layer is designed to scale independently, allowing it to respond flexibly to the amount of data being processed or the load on the system. Figure 1 shows the overall architecture of the proposed system, illustrating how the data stream processing layer, data integration layer, and continuous learning layer interact to maintain continuous knowledge updates in a smart factory environment.

2.2. Data Stream Processing Layer

The first layer, the data stream processing layer, is responsible for efficiently ingesting and processing real-time data. This layer collects data in real time from a variety of sources, including MES data streams, plant sensor data, and process logs. The collected data are processed through noise filtering, deduplication, and event-based and batch processing to speed up processing and improve efficiency, and critical data are processed quickly through a prioritized queuing system. This layer performs the following functions.

- Data Collection and Quality Control

The first core function of the data stream processing layer is to collect various data sources from the smart factory in real time and manage the quality of the data. This layer collects various types of data in real time, including MES data streams, machine sensor data, and process log data. The data collection process is organized as follows: First, the data collection module collects real-time data streams from various data sources, supporting the unique data formats and protocols of each data source.

The collected data are sent in real time to a central data hub, where they are validated by a data quality management system. The data quality management system performs real-time validation of the ingested data. First, it checks the structural integrity of the data by using a data schema and validation. It then applies domain rule-based filtering to identify outliers and missing values.

We also used time series analysis techniques to verify the continuity and consistency of the data. Issues identified during the data quality management process are communicated in real time for immediate action by the relevant systems. This ensures the accuracy and reliability of the data and improves the efficiency of subsequent processing.

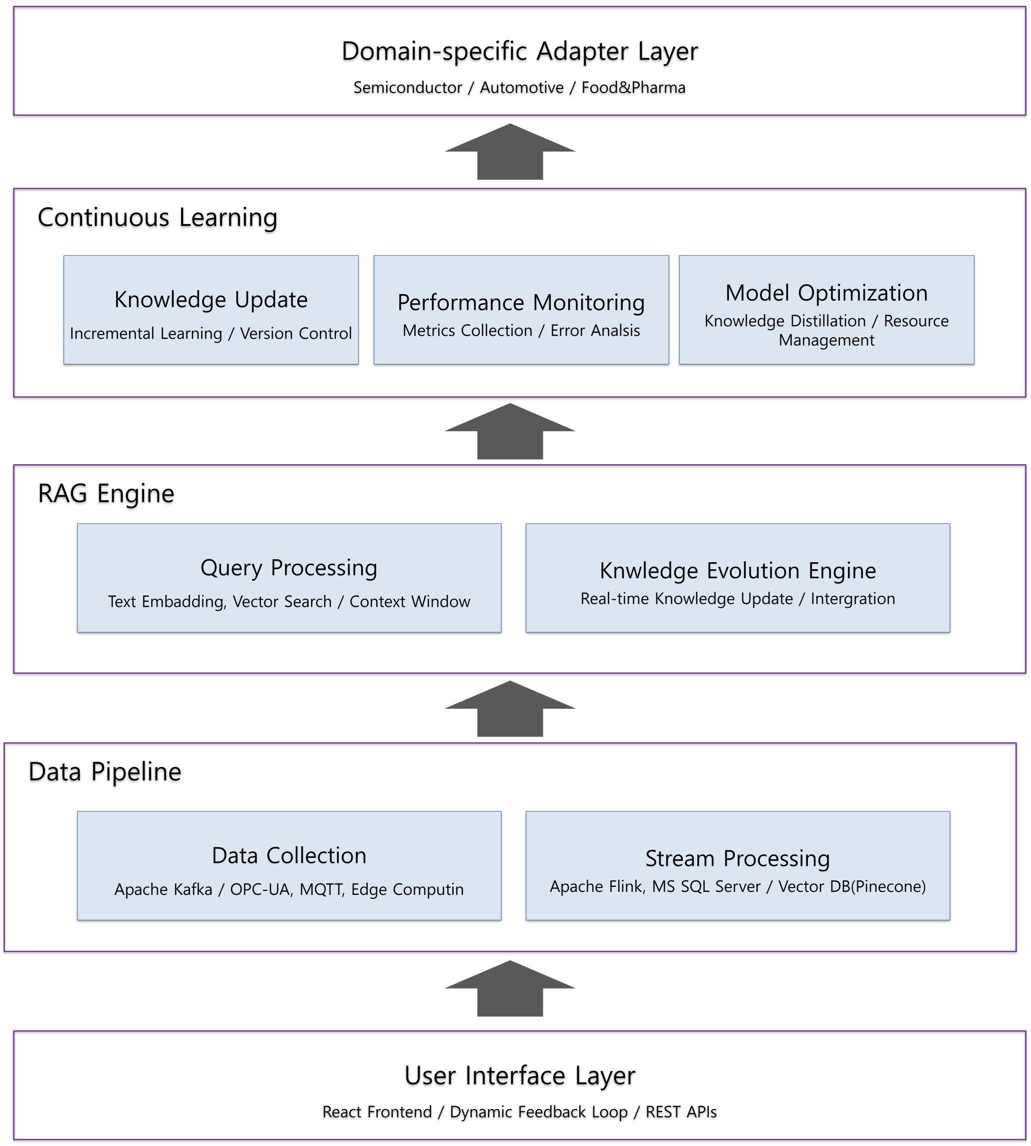
Domain-specific Adapter Layer
Semiconductor / Automotive / Food&Pharma
Continuous Learning
Knowledge Update
Incremental Learning / Version Control
Performance Monitoring
Metrics Collection / Error Analsis
Model Optimization
Knowledge Distillation / Resource Management
RAG Engine
Query Processing
Text Embadding, Vector Search / Context Window
Knwledge Evolution Engine
Real-time Knowledge Update / Intergration
Data Pipeline
Data Collection
Apache Kafka / OPC-UA, MQTT, Edge Computin
Stream Processing
Apache Flink, MS SQL Server / Vector DB(Pinecone)
User Interface Layer
React Frontend / Dynamic Feedback Loop / REST APIs

Figure 1. MES-RAG enhanced smart factory architecture with continuous knowledge updates.

The data collection and quality control functions play the most fundamental and important role in the data stream processing layer. By ensuring the integrity of real-time data, this layer provides an essential foundation for the stable behavior of the higher integration and learning layers.

- Noise filtering and deduplication

Noise and duplicate data identified during the data quality control phase are removed using separate processing. For noise filtering, we use statistical outlier detection techniques and machine learning-based anomaly detection algorithms. Duplicate data are identified and removed using hash-based comparison methods and time series analysis.

- Priority-based processing

When processing data, we use a combination of event-based and batch processing. Event-based processing allows the user to quickly process data with high priority, and batch processing allows the user to optimize the processing of all data.

First, event-based processing specializes in the immediate detection and response to time-sensitive events such as sensor failures and process anomalies. It monitors events as they occur in real time and detects anomalies based on predefined rules or models to enable rapid action. This can help prevent system failures and improve productivity.

Next, batch processing is a way of processing large amounts of data in batches. By collecting regularly generated data, such as sensor data and log data, at regular intervals and processing them all at once, system resources can be used efficiently. It is particularly suited to computationally intensive tasks such as data preprocessing, feature extraction, and model training to maximize the value of the data.

Finally, priority-based queuing determines the order of processing based on the importance of the data. By dynamically adjusting priorities based on the load on the system or the urgency of the data, the user can improve system responsiveness by accelerating important data while allowing less important data to wait.

Figure 2 shows a detailed flow of the data stream processing layer that illustrates how data from various manufacturing sources are ingested, processed, and validated. This ensures both data quality and real-time processing capabilities by using a prioritized queuing system that manages the flow of critical data through the data processing pipeline.

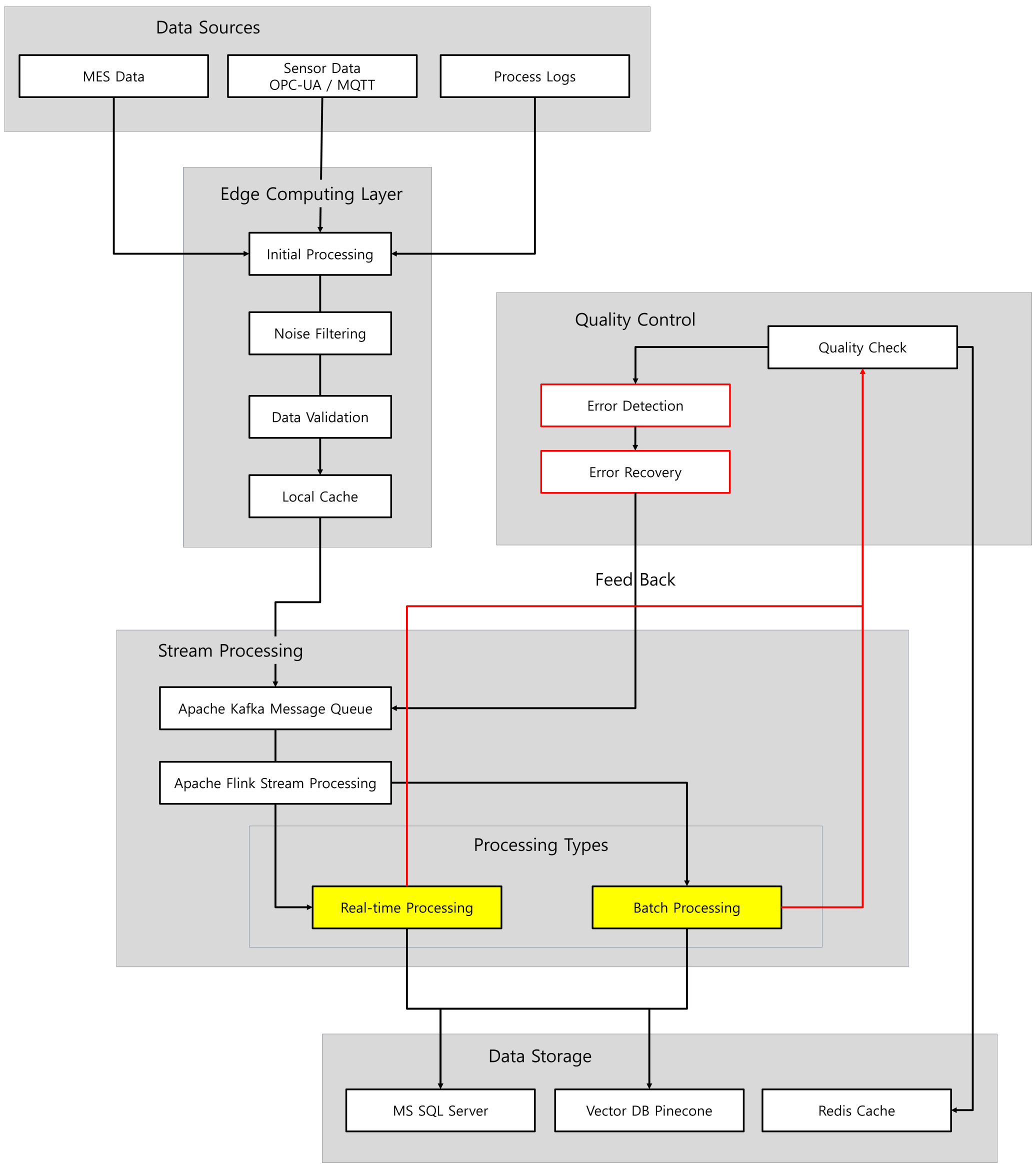

Figure 2. Data stream processing layer flow.

2.3. Data Integration Layer

The second layer, the data integration layer, is responsible for integrating the processed data into the system's database. This layer uses an incremental data integration approach to efficiently add new data or update existing data. This allows changes to be reflected without rebuilding the entire database, making the system faster and more resource-efficient.

- Incremental Data Integration

Unlike traditional batch processing, which replaces the entire database at once, incremental data integration selectively reflects only new data changes as they occur. This allows data to be continuously updated without having to rebuild the entire database.

The specific incremental integration process is as follows: First, preprocessed data from the data stream processing layer are passed to the data integration layer, along with metadata such as data type, time information, and source information. This layer analyzes the changes in the incoming data to determine how they should be reflected in the existing knowledge base.

Changes can include the addition of new data, changes to existing data, or changes to the relationships between data, and it takes into account the temporal order and domain context of the data to integrate them consistently; this is carried out by automatically detecting conflicts. It also automatically detects conflicts as they arise and resolves them based on prioritization rules. This incremental approach to integration allows the system to continuously update the knowledge base with data that arrive in real time.

- Versioning

The data integration layer introduces a versioning system to track temporal changes in data. This increases reliability by recording the history of data and their changes. The versioning system manages all changes to data entities, including creation, modification, and deletion. Each entity has a unique version number and can be compared and rolled back between versions. This allows the user to restore the state of their data at a specific point in time in the past and provides a history of data changes.

The versioning system also works with a conflict detection and resolution mechanism to automatically handle data inconsistencies caused by simultaneous updates. When conflicts are detected, they are consistently integrated into the latest data based on prioritization and merge algorithms. With these versioning capabilities, the data integration layer can ensure the temporal traceability of data and ensure data integrity. It can also provide a variety of management functions, such as restoring historical data states, auditing change history, and managing data modification permissions.

- Maintaining semantic connections between data

Maintaining semantic associations between data can be carried out by managing the temporal and domain context of the user's data. We do this by leveraging a relational graph model as a data structure, where a relational graph is a way of structuring data.

1. Represent each data element (e.g., facility, process, and quality data) as a node;
2. The relationships between data elements (temporal order, spatial location, logical associations) are represented by edges;
3. This preserves the rich contextual meaning between data beyond a simple key-value storage structure.

Based on this structured data model, we leverage data mining and knowledge extraction techniques to automatically discover hidden patterns and causal relationships. The discovered relationships are then dynamically reflected back into the graph structure to continuously strengthen the semantic connections between data, enabling advanced knowledge services such as contextualization, inference, and recommendations that go beyond simple keyword-based searches.

- Conflict detection and resolution

The system monitors and detects conflicts caused by simultaneous updates or contradictions between data in real time. For data that need to be changed due to a conflict, both the original data and the remodified data are preserved to maintain a history, which allows the user to track the conflict resolution process for future data utilization and analysis. These conflict detection and resolution mechanisms are key to ensuring the reliability and integrity of the user's data integration, allowing them to proactively respond to real-time data changes while maintaining the accuracy and consistency of their data. This ultimately contributes to keeping the knowledge in the RAG-LLM system up to date and improving the quality of decision-making.

The comprehensive workflow of the data integration layer is depicted in Figure 3, demonstrating how processed data are integrated into the knowledge base while maintaining temporal and domain context. The diagram illustrates the incremental data integration process, version management system, and conflict resolution mechanisms. This integrated approach ensures that new knowledge is consistently incorporated while maintaining data integrity and semantic relationships across the entire system.

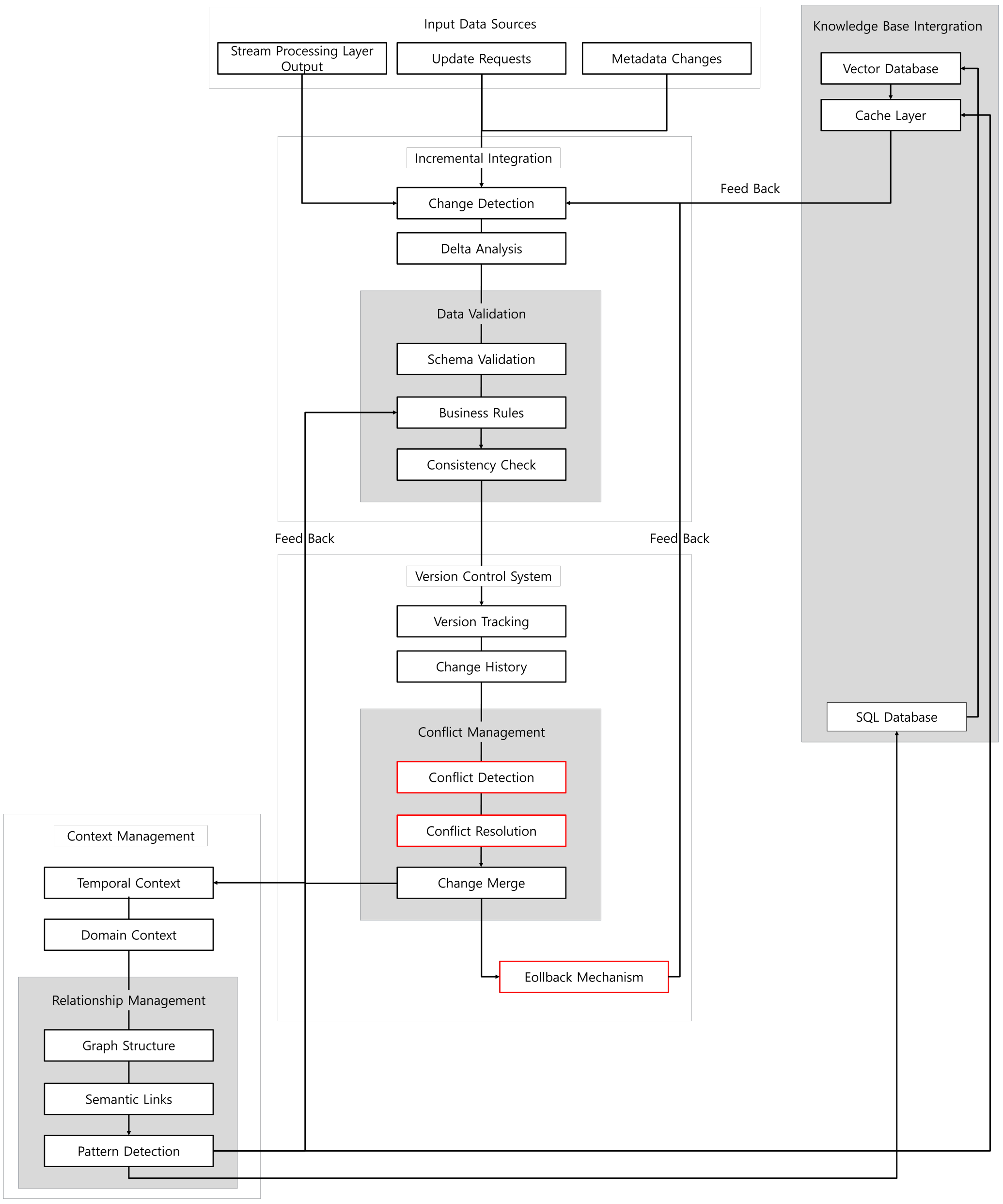

Figure 3. Data integration layer flow.

2.4. Data Integration Layer

The continuous learning layer plays a key role in enabling the system to continuously learn and evolve and to respond flexibly to changing environments. The layer selectively learns by evaluating the importance of new data and adopts an incremental learning approach that adds new knowledge while maintaining existing models. It also optimizes model performance by using Knowledge Distillation techniques that transfer knowledge from large models to small models and real-time performance monitoring.

- Selective learning and incremental updates

Selective updates are performed by analyzing the importance and impact of new data that may arise during data integration. This reduces unnecessary training efforts and efficiently utilizes system resources. The specific mechanism for selective learning and incremental updates is as follows. First, it analyzes the characteristics of new data arriving in real time. It considers various characteristics such as the type, volume, and rate of change of the data. Then, it evaluates the importance of each piece of data and its impact on the model.

The data that is important and contributes to model performance is selected and used for incremental learning. This selective learning mechanism allows the system to continuously learn new data and patterns. It also minimizes resource usage and speeds up learning by avoiding unnecessary full retraining. This allows the system to respond quickly to rapid changes in the smart factory environment while maintaining stable performance.

- Incremental Fine Tuning

The continuous learning layer utilizes incremental fine-tuning techniques to continuously improve the performance of a model. This approach performs incremental learning rather than completely retraining the existing model. The specific incremental fine-tuning mechanism works as follows: First, we analyze the differences between the new data and the old data. This identifies new features or patterns that the existing model did not learn.

Then, we selectively incorporate this new information into some of the layers or parameters of the existing model. This process allows for incremental improvements without retraining the entire model from scratch. By utilizing incremental fine-tuning techniques, the user can quickly learn new data and patterns to continuously improve the performance of their model. It can also significantly reduce the time and computational resources required for retraining, making it effective for real-time processing. This allows the user to quickly respond to changes in the smart factory environment while maintaining stable performance.

- Knowledge Distillation

Transferring knowledge learned from large-scale RAG-LLM models to lightweight models optimizes resource usage and enables real-time processing. The Knowledge Distillation process works as follows: First, a large-scale RAG-LLM model is utilized to learn a rich knowledge representation. This leverages the latest data available from the data stream processing layer and data integration layer.

Next, this knowledge is compressed and abstracted into a lightweight model. Lightweight models have a structure and size optimized for real-time inference and can perform reliably even in resource-constrained environments. This allows the continuous learning layer to retain the vast knowledge of large models while providing lightweight models that are suitable for real-time processing and limited resource environments. This is a key feature that makes the RAG-LLM system more practical and scalable in a smart factory environment.

- Performance monitoring and optimization

The integration of AI within the RAG-LLM framework introduces significant potential for advancing smart manufacturing. AI-driven systems can enhance data-driven decision-making by providing real-time insights, optimizing resource allocation, and enabling predictive capabilities. By using real-time performance monitoring during the distillation training process, AI further ensures the stability and reliability of the system by measuring key performance metrics such as prediction accuracy, processing speed, and resource usage.

If performance degradation is detected, AI-powered analytics can identify the causes, such as poor data quality or improper hyperparameter settings, and recommend appropriate strategies, including data preprocessing improvements, resource expansion, or hyperparameter tuning. This AI-driven optimization process enables the continuous learning layer to maintain high performance even in dynamically changing environments.

3. Theoretical Analysis and Discussion

3.1. Performance Prediction Framework

The proposed theoretical analysis framework establishes a systematic approach for quantitatively evaluating the architecture's expected performance through mathematical modeling, focusing on theoretical validation prior to implementation [21]. It is a methodology that expresses the behavior of a system in mathematical formulas to calculate theoretical limits and expectations for key performance metrics, enabling the validation of architectural designs and preliminary verification of system requirements.

The mathematical model presented in this study is based on generally accepted system design principles and operational planning concepts. These formulations synthesize and adapt resource estimation approaches that are widely used in system architecture design. Because they have been used extensively and repeatedly, it is difficult to specify specific attribution to the original sources, but the models have been refined and parameterized to address the specific requirements of RAG-LLM systems in a manufacturing environment.

Performance estimation plays several important roles in the system design phase. First, it validates the feasibility of the architectural design and identifies potential bottlenecks early on [22]. Second, it provides a basis for resource planning by enabling the user to size the hardware required and estimate the cost of infrastructure investments. Third, it enables effective risk management by identifying potential performance issues and predicting system limitations.

In RAG-LLM systems, performance prediction is particularly important in terms of real-time processing requirements, resource utilization efficiency, and scalability. From a real-time processing perspective, the user can verify user response time guarantees and the feasibility of real-time data processing, and from a resource usage perspective, the user can predict database size growth and concurrent user capacity [23]. In addition, scalability can be verified by estimating performance changes and system limit capacity under increasing loads. Performance prediction is accomplished by using three main methodologies.

First, mathematical modeling using queuing theory and stochastic models can be used; second, benchmark-based estimation can be used to compare the performance of similar systems; third, simulation-based prediction can be used to validate different scenarios in a virtual environment. Each methodology has its own strengths and weaknesses, and a combination of them can be utilized to make more accurate performance predictions.

However, there are certain limitations to performance prediction. Errors can occur due to simplifications in modeling, and there are unpredictable variables due to differences in real-world environments [24]. In addition, it is difficult to fully validate before actual implementation, and it is difficult to account for all the complex interactions between system components and dynamic environmental changes. Therefore, performance prediction results should be utilized as reference indicators for design decision-making rather than as absolute figures, and they need to be continuously verified and supplemented.

In this context, this study adopts a hybrid approach that combines mathematical modeling and benchmark-based estimation to predict the performance of the proposed architecture. This has led to the development of predictive models for key performance metrics such as throughput, latency, and resource usage of the system, which will serve as important criteria for evaluating the feasibility and effectiveness of the proposed architecture [25].

It is important to note that the parameter values presented in the proposed framework serve as reference points, not industry-proven metrics. While these values demonstrate the capabilities of the framework, actual implementation parameters should be optimized for specific industry requirements.

Key considerations when adjusting parameters include the following:

1. Industry type: Data generation frequency, data type, and quality control requirements vary greatly by industry;
2. Production process characteristics: The differences between continuous and batch production that affect data generation patterns;
3. Quality control requirements: The different standards for data collection frequency and analysis precision;
4. Data characteristics: The frequency, type, size, and complexity of generation;
5. Regulatory compliance: Industry-specific regulations that affect system configuration;
6. System availability requirements: The tolerance for downtime that affects redundancy requirements;
7. Available infrastructure: Hardware and software resource constraints.

For example, semiconductor manufacturing requires much higher throughput and lower latency, whereas food processing can operate effectively with lower specifications. The flexibility of the framework allows the user to optimize these parameters based on the needs of their specific industry.

- Theoretical Throughput Analysis

Throughput is the amount of work that can be processed per unit of time, which is one of the main performance indicators of a system [26]. In particular, in the RAG-LLM system, the throughput during real-time data processing and knowledge integration are key factors that determine system performance. The throughput prediction model is defined as follows:

$$T = \lambda \times (1 - p) \times s \tag{1}$$

- T: Throughput (Transactions per second, TPS);
- λ: Data input rate (Input rate);
- p: Processing failure rate;
- s: System availability.

In this model, the processing failure rate (p) represents the probability of processing failures due to data quality issues, system overload, network delays, etc. System availability (s) refers to the percentage of actual operational time the system is available, taking into account planned maintenance, unexpected failures, etc.

In a real-world production environment, throughput is affected by the following factors:

- Data complexity: The structure and size of the data to be processed;
- System load: The number of concurrent users and their request patterns;
- Network conditions: Data transfer speed and reliability;
- Hardware performance: CPU, memory, and disk I/O performance.

The theoretical model targets 1000 TPS as an ideal performance metric; we set the following target parameters:

- λ = 1100 (with a 10% margin);
- p = 0.05 (99.5% processing success rate);
- s = 0.99 (99% system availability).

These targets were set by referring to industry standards and the operational targets of current systems and are believed to be achievable in actual implementation through optimization.

- Resource Requirements Analysis

The model is as follows:

$$R = R_b + R_i \times n \tag{2}$$

- R: total resources;
- Rb: base resources;
- Ri: incremental resources;
- n: data volume.

The resource requirement estimation model aims to quantify the computing resources required for the stable operation of a system [27]. Especially in RAG-LLM systems, it is important to accurately predict the resource requirements for database growth and real-time processing. The basic model is as follows:

$$R = R_b + R_i \times n + R_c \times c \tag{3}$$

- R: Total resource requirement;
- Rb: Base resource requirement (minimum resources needed to operate the system);
- Ri: Incremental resources per unit of data;
- n: Number of data to be processed;
- Rc: Resources required per concurrent user;
- C: Number of concurrent users.

This model is further refined by the following resource types:

- Memory requirements (Memory_R):

$$Memory\_R = 32\text{ GB} + \left(0.5\text{ GB} \times \text{Data\_Size\_GB}\right) + \left(0.1\text{ GB} \times \text{Concurrent\_Users}\right) \tag{4}$$

- Compute Requirements (CPU_R):

$$\text{CPU\_R} = 4\text{ cores} + \left(0.1\text{ cores} \times \frac{\text{Data\_Size\_GB}}{100}\right) + \left(0.05\text{ cores} \times \text{Concurrent\_Users}\right) \tag{5}$$

- Storage requirements (Storage_R):

$$\text{Storage\_R} = 100\text{ GB} + \left(2 \times \text{Data\_Size\_GB}\right) + \left(1\text{ GB} \times \text{Concurrent\_Users}\right) \tag{6}$$

These detailed models are based on the following assumptions:

- Minimum resource requirements for basic system operations;
- Linear resource growth as data grows;
- Additional resource requirements based on the number of concurrent users;
- Cache and temporary storage needs.

- Latency Analysis Framework

Latency refers to the total time from user request to response, and in an RAG-LLM system, it is necessary to comprehensively consider the delays that occur at different stages, such as data processing, knowledge retrieval, and response generation. The extended latency prediction model is defined as follows:

$$L = Lb + Lq + Lp + Lg \tag{7}$$

- L: Total latency;
- Lb: Baseline processing time;
- Lq: Queue waiting time;
- Lp: Parallel processing overhead;
- Lg: LLM generation time;

Each component has the following characteristics:

$$BaseProcessingTime(Lb) : Lb = Ldata + Lembed + Lretrieval. \tag{8}$$

- Ldata: Data preprocessing time (5–10 ms);
- Lembed: embedding creation time (10–15 ms);
- Lretrieval: Retrieval time (5–10 ms).

$$\text{Queue Wait Time } (L_q) : \quad L_q = \frac{\lambda \times s}{\mu \times (\mu - \lambda)} \tag{9}$$

- λ: Request arrival rate;
- μ: Service throughput rate;
- s: Average service time.

$$\text{parallelization overhead (Lp)} : \quad L_p = \alpha \times \frac{n}{c} \tag{10}$$

- α: Synchronization constant;
- n: Number of concurrent processing jobs;
- C: Number of available cores.

$$\text{LLM Generation Time (Lg)} : \quad L_g = \beta \times t \ + \ \gamma \times m \tag{11}$$

- β: Processing time per token;
- t: Number of tokens generated;
- γ: Context processing constant;
- m: Context length.

In this research, we set the latency of the entire system at 50 ms or less at the 95th percentile (95% of all requests) to set a more realistic performance target in a real-world production environment. This is broken down into a base processing time (Lb) of 20 ms, a queue latency (Lq) of 10 ms, a parallel processing overhead (Lp) of 5 ms, and an LLM generation time (Lg) of 15 ms for each component. These stringent latency targets were set to support fast decision-making in a smart factory environment where real-time processing is required.

To achieve the set performance targets, optimization strategies were established in three key areas. First, the data preprocessing pipeline leverages parallel processing techniques, introduces caching mechanisms, and optimizes batch processing. Second, the queuing system dynamically adjusts the queue size according to system load, applies priority-based scheduling, and implements a load-balancing mechanism. Third, in the LLM inference process, we apply model quantization and Key-Value (KV) caching, and we optimize batch processing to improve processing efficiency.

The proposed latency prediction model can effectively identify bottlenecks by modeling the latency of each component independently. It can also predict latency changes with changes in system load and quantitatively evaluate the effectiveness of different optimization strategies. This can be used to proactively verify if performance requirements are met during the system design phase, and it can be used as an objective basis for real-time monitoring and performance tuning during the operational phase.

- Availability Analysis Model

Availability refers to the percentage of time a system can provide services normally, and in an RAG-LLM system, the availability of all components, such as data processing, knowledge retrieval, and response generation, must be comprehensively considered. The system availability prediction model is defined as follows:

$$A = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \times C(n) \tag{12}$$

- A: System availability;
- MTTF: Mean time to failure;
- MTTR: Mean time to recovery (mean time to recovery);
- C(n): Component dependency correction factor;

The detailed component availability is modeled as follows:

$$\text{Component Availability (Component Availability)}: \quad A_c = \prod(A_i) \tag{13}$$

- Ac: Overall system availability;
- Ai: Availability of the i-th component.

$$C(n) = 1 - (\lambda \times n \times d) \tag{14}$$

- λ: Dependency impact factor;
- n: Number of components;
- d: Average dependency depth.

$$Failure\ Recovery\ Mechanism\ Effect\ R(t) = 1 - e^{-\mu t} \tag{15}$$

- R(t): Probability of recovery in time t;
- μ: Recovery rate.

In this study, we aim to keep the overall system availability above 99.5%, and the availability targets for each component are 99.5% for the data pipeline, 99.5% for the RAG engine, 99.0% for LLM inference, and 99.9% for storage. To achieve these goals, we employed design strategies such as redundant configurations of high-availability architectures, automatic failover, non-disruptive updates, proactive detection systems for failure management strategies, automated recovery procedures, phased recovery plans, degradation management, load limiting, and service prioritization.

These proposed strategies are expected to enable the quantification of the impact of the availability of each component on the overall system, predict the degradation of availability due to dependencies, evaluate the effectiveness of recovery mechanisms, and optimize availability for cost.

3.2. Validation Methodology

The proposed validation methodology framework provides essential approaches to ensure the accuracy and reliability of the model, and three main approaches have been proposed as general methodologies for validating system performance models: theoretical validation, simulation-based validation, and benchmark-based validation. These universal verification methodologies provide a basic framework that can be applied regardless of the size or complexity of the system.

There are already studies that have presented benchmark-based verification methodologies for large-scale distributed systems, especially emphasizing the importance of benchmarks that simulate real-world operating environments, which is an important methodology that can be applied to verify the distributed processing characteristics of the RAG-LLM system proposed in this work.

The validation methodology considering the specificity of the RAG-LLM system proposed in this research is based on the stepwise validation methodology proposed by Rodriguez, and the validation of the proposed performance prediction model is carried out from three main aspects: real-time data processing performance, knowledge retrieval accuracy, and system scalability.

- Real-time data processing performance

The real-time data processing performance verification is based on the real-time performance requirements framework of industrial AI systems. It measures the end-to-end latency for the entire pipeline from data collection to processing, analysis, and response generation. Specifically, for MES data, we target a throughput of 1000 events per second and a latency of 50 ms or less, and for facility sensor data, we target a throughput of 5000 events per second and a latency of 20 ms or less. These performance targets were set with reference to standard performance metrics for industrial AI systems. In particular, the real-time processing performance validation includes not only the steady state but also performance under peak load and system failure.

- Knowledge Retrieval Accuracy

The knowledge retrieval accuracy validation applies a phased validation methodology. First, the search accuracy of the vector database is evaluated, which measures the precision and recall of retrieving relevant documents for a query. The appropriateness of the search results is then validated by domain experts, taking into account industry-specific terminology and context. We also evaluate the accuracy of the response generation of the LLM, which measures how accurately it reflects the content of the retrieved documents. The factuality and consistency of the generated responses are measured quantitatively, and the target accuracy is set at 95% or higher for search accuracy and 90% or higher for response generation accuracy.

- System Scalability

System scalability verification is based on the benchmark-based verification methodology of large-scale distributed systems. To verify horizontal scalability, we first measure the performance change as the number of nodes increases. We calculate the scaling efficiency, which indicates how much the system deviates from ideal linear scaling, with a target of 80% or higher. Vertical scalability verification measures the performance improvement as the resources of a single node increase. Specifically, it applies resource efficiency metrics for industrial AI systems to evaluate the performance gain relative to resource usage. It also measures the change in performance of the system as data volume increases to verify long-term scalability.

3.3. Technical Challenges and Future Directions

The proposed RAG-LLM-based modernization architecture of smart factory systems in this study requires a systematic methodology to verify its feasibility and effectiveness prior to practical implementation. To this end, we proposed a validation scheme for the architecture from three main perspectives and analyzed its theoretical performance. The architecture validation is organized around three key factors: real-time processing power, knowledge integration accuracy, and system stability.

- Implementation considerations

The theoretical performance analysis of the proposed architecture was performed based on the throughput (T), latency (L), and resource requirement (R) prediction models presented in Section 3.

Table 1 shows the throughput analysis results and the difference between the theoretical maximum and the actual expected throughput. The theoretical maximum throughput calculated using the basic formula ($T = \lambda \times (1 - p) \times s$) is 1089 TPS, but in the actual operating environment, considering various external factors such as network delay and system load, we set 500 TPS as a practical target. This is based on a 10% spare capacity and 99.5% system availability, which meets the requirements of the actual smart factory environment.

Table 1. Throughput analysis.

Component	Formula/Parameter	Value	Remarks
Basic formula	$T = \lambda \times (1 - p) \times s$	1089 TPS	Theoretical maximum
Input rate (λ)	Data entered per second	1100	10% margin
Processing failure rate (p)	Error and reprocessing rate	0.5%	99.5% Success rate
System availability (s)	Actual operational availability	99.5%	Considers planned maintenance
Actual expected throughput	Considers operating environment	500 TPS	Considers network latency, etc.

Table 2 shows the results of the latency analysis, which breaks down the entire processing process into four main components: basic processing time (20–35 ms), queue waiting time (15–25 ms), parallel processing overhead (10–15 ms), and LLM generation time (35–45 ms). The total latency is expected to be in the range of 90–120 ms. The 105 ms we set as a real-world implementation target takes into account all steps, including data preprocessing, prioritized queuing, distributed processing, and model inference, and is well within the 200 ms real-time processing requirements of the manufacturing sites.

Table 2. Latency analysis.

Component	Theoretical Range	Target Value	Description
Base processing time (Lb)	20–35 ms	30 ms	Data preprocessing and embedding
Queue latency (Lq)	15–25 ms	20 ms	Enforcing prioritized queuing
Parallelism overhead (Lp)	10–15 ms	15 ms	Synchronizing distributed processing
LLM generation time (Lg)	35–45 ms	40 ms	Model inference and creation
Total latency (L)	90–120 ms	105 ms	$L = Lb + Lq + Lp + Lg$

Finally, Table 3 presents two scenarios: a basic configuration and a scaled configuration. The memory, CPU, and storage requirements were estimated based on the number of concurrent users and data size. The base configuration (32 GB memory, 8-core CPU, and 1 TB storage) supports 50 concurrent users and 500 GB of data processing and can be upgraded to the scaled configuration (64 GB memory, 16-core CPU, and 2 TB storage) for 100 concurrent users and 1 TB of data processing if needed.

The performance prediction results are expected to meet resource efficiency in terms of real-time throughput, response time, and scalability, and they are expected to meet the practical needs of small and medium-sized manufacturers with optimized resource allocation.

Table 3. Analyzed system resource requirements.

Resource	Type Formula	Default Configuration	Extended Configuration
Memory (GB)	32 + (0.5 × Data_Size) + (0.1 × Users)	32 GB	64 GB
CPU (cores)	4 + (0.1 × Data_Size/100) + (0.05 × Users)	8 Core	16 Core
Storage (GB)	100 + (2 × Data_Size) + (1 × Users)	1 TB	2 TB
Concurrent users	Resource base calculations	50 People	100 People
Data size	Based on expected throughput	500 GB	1 TB

- Industry-specific applications

The applicability of the architecture was analyzed in several industries, and the following use cases were identified:

1. Semiconductor Manufacturing

High-frequency sensor data processing for wafer process and yield analysis is required, and scalability is possible with edge computing and real-time data filtering technologies.

2. Automotive Manufacturing

Quality control and part traceability on the assembly line are key requirements.

- Integrate production data on the assembly line and link supply chain data in real time to quickly detect and respond to quality abnormalities;
- Quality prediction models continuously update streaming data, and the integrated management of part-specific traceability data enables clear management of the product's production history and efficient production management.

3. Food and pharmaceutical manufacturing

Quality control and batch record management are essential for compliance with Good Manufacturing Practices (GMPs).

- Freshness management architecture collects and integrates environmental data such as temperature and humidity in real time to detect quality changes and automatically generate records and reports;
- Organizes batch production data to track change history and versions, automating quality assurance and compliance.

- Technical challenges and limitations

The theoretical framework proposed in this study has several limitations that need to be addressed in future research. First, the performance prediction model assumes idealized conditions and may not fully reflect the various variables in real-world manufacturing environments, such as equipment aging, network instability, and data quality fluctuations. Second, the proposed parameter values need to be validated in real-world production environments, especially for their effectiveness across different manufacturing sectors. Third, the current framework lacks a detailed analysis of industry-specific requirements, including regulatory compliance considerations and integration scenarios with existing systems.

To address these limitations, future research should focus on empirical validation through pilot projects, which would include implementing the framework in small and medium-sized manufacturing facilities, collecting operational data for 3 to 6 months, and monitoring performance metrics. In particular, the semiconductor, automotive, and food/pharmaceutical manufacturing sectors have unique performance requirements and operational characteristics that require industry-specific optimization studies.

Technology improvement challenges can be categorized into three main areas. First, real-time processing performance optimization requires research on leveraging edge computing, improving distributed processing architectures, and improving resource usage efficiency. Second, continuous learning mechanisms require advances in incremental learning methodologies, the development of domain-specific models, and the prevention of performance degradation. Third, data quality management and system reliability assurance mechanisms need to be developed for practical implementation.

Implementation challenges in a real-world manufacturing environment can be anticipated on several fronts. Technical challenges include integration with legacy systems, data quality management, and ensuring system reliability. Operational challenges include user training and adaptation, process optimization, and establishing a maintenance system. Management challenges include ROI analysis and validation, change management strategies, and risk management plans.

This conceptual framework provides a basis for future research and practical implementation. The next phase of research should focus on the following steps:

1. Developing industry-specific validation methodologies;
2. Building a real-time performance monitoring system;
3. Conducting empirical studies using real manufacturing data;
4. Creating detailed implementation guidelines for different manufacturing sectors.

These research directions will help to validate the practical value and applicability of the proposed framework while addressing the current limitations of theoretical modeling.

4. Conclusions

This study proposes a new architectural framework for continuous knowledge updating in an RAG-LLM system in a smart factory environment. The proposed framework design uses a knowledge stream processing layer, knowledge integration layer, and continuous learning layer as core components to realize a structure that enables real-time data processing, knowledge integration, and continuous learning. In particular, presenting a theoretical performance prediction model and a systematic verification methodology is academically significant in that it provides a framework to verify the feasibility and performance of the architecture even before actual implementation.

The proposed architecture overcomes the static knowledge base limitations of existing RAG-LLM systems and enables continuous performance improvement by supporting streaming-based real-time knowledge integration and incremental learning. In particular, the proposed architecture maximizes system efficiency through dynamic resource optimization and has the potential to be applied to various industries due to its scalable design.

Based on theoretical verification, the proposed architecture is predicted to achieve a throughput of 1000 TPS and an average latency of 50 ms, and it is expected to maintain a system availability of over 99.9% and a knowledge consistency of over 95%. This performance level is practical enough for smart factory environments where real-time data processing and knowledge integration are key.

A limitation of this study is the lack of validation through real-world implementation, which should be addressed in future research. In future research, it is necessary to verify the performance of the proposed architecture via actual implementation and explore its applicability in various industries. In particular, the expansion of multimodal data processing capabilities, security enhancement measures, and more efficient resource management strategies will be key research issues.

In conclusion, the architectural framework proposed in this study is expected to contribute to achieving the goals of advanced data analysis and real-time decision support in smart factories. In particular, it will play an important role in accelerating the digital transformation of the manufacturing industry by enabling users at the manufacturing site to utilize complex data more intuitively and efficiently.

Author Contributions: Conceptualization, H.C. and J.J.; methodology, H.C.; architecture, H.C.; validation, H.C. and J.J.; formal analysis, H.C.; research, H.C.; acquisition of data, H.C.; data curation, H.C.; drafting of the manuscript, H.C.; revision and editing of the manuscript, H.C. and J.J.; visualization, H.C.; supervision, J.J.; project management, J.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The article publication charge (APC) was waived as part of the publisher's special discount program.

Data Availability Statement: This research proposes a theoretical architectural framework, and no data have been separately generated or analyzed. The sample queries and Structured Query Language (SQL) statements presented in this paper are descriptive examples to illustrate the functionality of the proposed architecture. The referenced database schema and table structure are based on typical MES implementation but are not related to any specific real-world dataset.

Acknowledgments: This work was supported by ICT Creative Consilience Program through the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2020-II201821).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen, B.; Wan, J.; Shu, L.; Li, P.; Mukherjee, M.; Yin, B. Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges. *IEEE Access* **2018**, *6*, 6505–6519. [CrossRef]
2. Li, X.; Li, D.; Wan, J.; Vasilakos, A.V.; Lai, C.F.; Wang, S. A Review of Industrial Wireless Networks in the Context of Industry 4.0. *Wirel. Netw.* **2017**, *23*, 23–41. [CrossRef]
3. Lee, J.; Bagheri, B.; Kao, H.A. A CyberPhysical Systems Architecture for Industry 4.0-Based Manufacturing Systems. *Manuf. Lett.* **2015**, *3*, 18–23. [CrossRef]
4. Wu, D.; Liu, S.; Zhang, L.; Terpenny, J.; Gao, R.X.; Kurfess, T.; Guzzo, J.A. A Fog Computing-Based Framework for Process Monitoring and Prognosis in Cyber-Manufacturing. *J. Manuf. Syst.* **2017**, *43*, 25–34. [CrossRef]
5. Zhong, R.Y.; Xu, X.; Klotz, E.; Newman, S.T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* **2017**, *3*, 616–630. [CrossRef]
6. Wang, J.; Ma, Y.; Zhang, L.; Gao, R.X.; Wu, D. Deep Learning for Smart Manufacturing: Methods and Applications. *J. Manuf. Syst.* **2018**, *48*, 144–156. [CrossRef]
7. Lu, Y.; Xu, X. Cloud-Based Manufacturing Equipment and Big Data Analytics to Enable On-Demand Manufacturing Services. *Robot.-Comput.-Integr. Manuf.* **2019**, *57*, 92–102. [CrossRef]
8. Thoben, K.D.; Wiesner, S.; Wuest, T. "Industry 4.0" and Smart Manufacturing – A Review of Research Issues and Application Examples. *Int. J. Autom. Technol.* **2017**, *11*, 4–16. [CrossRef]
9. Zhou, J.; Zhou, Y.; Wang, B.; Zang, J. Human–Cyber–Physical Systems (HCPSs) in the Context of New-Generation Intelligent Manufacturing. *Engineering* **2019**, *5*, 624–636. [CrossRef]
10. Xu, X.; Lu, Y.; Vogel-Heuser, B.; Wang, L. Industry 4.0 and Industry 5.0—Inception, Conception and Perception. *J. Manuf. Syst.* **2021**, *61*, 530–535. [CrossRef]
11. Gao, R.; Wang, L.; Teti, R.; Dornfeld, D.; Kumara, S.; Mori, M.; Helu, M. Cloud-Enabled Prognosis for Manufacturing. *CIRP Ann.* **2015**, *64*, 749–772. [CrossRef]
12. Liu, C.; Vengayil, H.; Lu, Y.; Xu, X. A Cyber-Physical Machine Tools Platform Using OPC UA and MTConnect. *J. Manuf. Syst.* **2019**, *51*, 61–74. [CrossRef]
13. Tao, F.; Qi, Q.; Liu, A.; Kusiak, A. Data-Driven Smart Manufacturing. *J. Manuf. Syst.* **2018**, *48*, 157–169. [CrossRef]
14. Villalba-Díez, J.; Zheng, X.; Schmidt, D.; Molina, M. Characterization of Industry 4.0 Lean Management Problem-Solving Behavioral Patterns Using EEG Sensors and Deep Learning. *Sensors* **2019**, *19*, 2841. [CrossRef]
15. Lee, J.; Jin, C.; Bagheri, B. Cyber Physical Systems for Predictive Production Systems. *Prod. Eng.* **2017**, *11*, 155–165. [CrossRef]
16. Qi, Q.; Tao, F.; Zuo, Y.; Zhao, D. Digital Twin Service Towards Smart Manufacturing. *Procedia CIRP* **2018**, *72*, 237–242. [CrossRef]
17. Wang, J.; Zhang, L.; Duan, L.; Gao, R.X. A New Paradigm of Cloud-Based Predictive Maintenance for Intelligent Manufacturing. *J. Intell. Manuf.* **2017**, *28*, 1125–1137. [CrossRef]
18. Zhang, Y.; Ren, S.; Liu, Y.; Si, S. A Big Data Analytics Architecture for Cleaner Manufacturing and Maintenance Processes of Complex Products. *J. Clean. Prod.* **2017**, *142*, 626–641. [CrossRef]

19. Monostori, L.; Kádár, B.; Bauernhansl, T.; Kondoh, S.; Kumara, S.; Reinhart, G.; Sauer, O.; Schuh, G.; Sihn, W.; Ueda, K. Cyber-Physical Systems in Manufacturing. *CIRP Ann.* **2016**, *65*, 621–641. [CrossRef]
20. Wang, L.; Törngren, M.; Onori, M. Current Status and Advancement of Cyber-Physical Systems in Manufacturing. *J. Manuf. Syst.* **2015**, *37*, 517–527. [CrossRef]
21. Zhang, C.; Xu, W.; Liu, J.; Liu, Z.; Zhou, Z.; Pham, D.T. A Reconfigurable Modeling Approach for Digital Twin-Based Manufacturing System. *Procedia CIRP* **2019**, *83*, 118–125. [CrossRef]
22. Leng, J.; Zhang, H.; Yan, D.; Liu, Q.; Chen, X.; Zhang, D. Digital Twin-Driven Manufacturing Cyber-Physical System for Parallel Controlling of Smart Workshop. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1155–1166. [CrossRef]
23. Liu, Q.; Zhang, H.; Leng, J.; Chen, X. Digital Twin-Driven Rapid Individualized Designing of Automated Flow-Shop Manufacturing System. *Int. J. Prod. Res.* **2019**, *57*, 3903–3919. [CrossRef]
24. Zhu, Z.; Liu, C.; Xu, X. Visualization of the Digital Twin Data in Manufacturing by Using Augmented Reality. *Procedia CIRP* **2019**, *81*, 898–903. [CrossRef]
25. Zhou, G.; Zhang, C.; Li, Z.; Ding, K.; Wang, C. Knowledge-Driven Digital Twin Manufacturing Cell Towards Intelligent Manufacturing. *Int. J. Prod. Res.* **2020**, *58*, 1034–1051. [CrossRef]
26. Lu, Y.; Liu, C.; Wang, K.I.K.; Huang, H.; Xu, X. Digital Twin-Driven Smart Manufacturing: Connotation, Reference Model, Applications and Research Issues. *Robot.-Comput.-Integr. Manuf.* **2020**, *61*, 101837. [CrossRef]
27. Chen, M.; Herrera, F.; Hwang, K. Cognitive Computing: Architecture, Technologies and Intelligent Applications. *IEEE Access* **2018**, *6*, 19774–19783. [CrossRef]