Leveraging Knowledge Graphs and LLM Reasoning to Identify Operational Bottlenecks for Warehouse Planning Assistance

Rishi Parekh
Quantiphi
Mumbai,India
rishi.parekh@quantiphi.com

Saisubramaniam Gopalakrishnan
Quantiphi
Bengaluru,India
saisubramaniam.gopalakrishnan@quantiphi.com

Zishan Ahmad
Quantiphi
Bengaluru,India
zishan.ahmad@quantiphi.com

Anirudh Deodhar*
Quantiphi
Mumbai,India
anirudh.deodhar@quantiphi.com

Abstract

Analyzing large, complex output datasets from Discrete Event Simulations (DES) of warehouse operations to identify bottlenecks and inefficiencies is a critical yet challenging task, often demanding significant manual effort or specialized analytical tools. We propose a novel framework that addresses this challenge through a powerful integration of Knowledge Graphs (KGs) and Large Language Model (LLM)-based agents. Our framework first transforms raw DES output data into a semantically rich KG, which uniquely captures the intricate relationships between simulation events and entities (e.g., suppliers, packages, workers, equipment), overcoming unstructured log limitations for robust analysis. On this KG, our novel LLM-based agent employs a sophisticated iterative reasoning mechanism that interprets complex natural language questions by generating insightful, interdependent sub-questions sequentially. Each sub-question is formulated one at a time, crucially conditioned on the evidence from answers to previous ones. For each individual sub-question, a multi-step process then generates precise Cypher queries for KG interaction, extracts relevant information, and performs crucial self-reflection to identify and correct potential errors. This adaptive, iterative, and self-correcting reasoning progressively pinpoints operational issues and diagnoses root causes, mimicking human investigative analysis. We evaluate our approach using an example warehouse DES setup, systematically introducing typical bottlenecks like equipment breakdowns and supplier arrival irregularities. For operational questions, our proposed pipeline using step-wise thinking demonstrates significantly higher pass rates compared to traditional baseline methods, achieving near-perfect performance in identifying key inefficiencies. Furthermore, for more complex investigative questions, we qualitatively showcase our framework's superior diagnostic capabilities through three case studies, highlighting its proficiency in uncovering subtle and interconnected inefficiencies often missed by traditional methods. This work attempts to bridge the gap between simulation modeling and advanced AI-driven data analysis informed by the broader advancements in KG+LLM, offering a more intuitive and potent method for extracting actionable insights from simulation outputs, thereby dramatically reducing time-to-insight and paving the way for automated, intelligent warehouse inefficiency evaluation and diagnosis for industrial data analysis.

ACM Reference Format:
Rishi Parekh, Saisubramaniam Gopalakrishnan, Zishan Ahmad, and Anirudh Deodhar. 2025. Leveraging Knowledge Graphs and LLM Reasoning to Identify Operational Bottlenecks for Warehouse Planning Assistance. In *Proceedings of the 1st Workshop on "AI for Supply Chain: Today and Future" @ 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25), August 3, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/XXXXXX.XXXXXX

1 Introduction

Modern warehouses are complex systems, defined by intricate interactions between resources (personnel, equipment), processes (receiving, storage, picking, packing, shipping), and physical layouts [7, 9, 21]. Discrete Event Simulation (DES) has emerged as a powerful technique for modeling these systems [5, 20], allowing stakeholders to evaluate performance, test design alternatives, and understand system dynamics before implementation. Despite its power, analysis of the voluminous and highly granular output data generated restricts the full exploitation of DES. Typical DES runs produce extensive event logs, time-series data on resource states, and detailed queue statistics, capturing micro-level system behavior. Transforming this raw data into actionable intelligence—such as precisely identifying performance bottlenecks, diagnosing root causes of delays, or pinpointing underutilized resources—presents a non-trivial analytical hurdle. Conventional approaches, frequently reliant on manual inspection of aggregate statistics [20] or development of custom scripts tailored to specific simulation outputs, are not only time-intensive and error-prone but, critically, often fail to uncover complex, emergent system behaviors and hidden inefficiencies arising from the interplay of numerous components.

The need for more sophisticated and efficient analysis methods is further amplified by the increasing integration of simulation with real-time operational data, particularly within the paradigm of Digital Twins (DTs) [2, 22, 32]. While DTs aim to provide a synchronized virtual counterpart to physical systems for monitoring and decision-making, the fundamental challenge of interpreting the state and behavior of the simulation or DT persists. Whether analyzing historical simulation runs or near real-time DT states, extracting clear insights remains a bottleneck in itself. AI is playing an increasingly important role in all fields; warehouse logistics is no exception [8, 16, 27, 41]. However, to effectively address the specific analytical bottleneck in DES and DT data interpretation for warehouses, there is a pressing need for advanced AI-driven approaches capable of

*Corresponding author

KDD '25, Toronto, ON, Canada.

ACM ISBN 979-8-4007-1454-2/25/08
https://doi.org/10.1145/XXXXXX.XXXXXX

unlocking the rich insights embedded within this complex data, thereby truly enhancing warehouse planning and operational control.

To address these limitations of conventional DES data analysis, our work proposes a novel framework integrating Knowledge Graphs (KGs) [12] and Large Language Model (LLM)s [31, 39] for bottleneck identification through natural language queries, leading towards intelligent warehouse planning assistance. The core idea of our work, supported by research on domain-specific and event KGs [1], is to first structure the complex relational data generated by DES using a KG. Representing simulation output as a graph allows the intricate dependencies and flows within the warehouse system to be explicitly captured and queried. While KGs are increasingly applied to analyze real-world industrial and supply chain data for enhanced visibility and risk management [19, 30], their application to the specific domain of simulation output data remains relatively unexplored. This work leverages KG technology to achieve deeper simulation understanding, thereby supporting strategic and operational warehouse planning.

Building upon the structured representation provided by the KG, our framework employs LLM-based agents [10, 38] to enable intuitive interaction with the simulation data [37], directly aiding warehouse planners. LLMs possess powerful natural language understanding and generation capabilities, allowing users—such as operations analysts or industrial engineers involved in warehouse design, optimization, and day-to-day planning, who may lack deep expertise in graph query languages—to pose questions in natural language. The LLM agent in our framework is not merely an intermediary but is designed with an iterative reasoning [24, 36] mechanism for the in-depth diagnostic analysis crucial for warehouse planning. When presented with a complex natural language query regarding warehouse performance or a planning scenario, the agent autonomously generates a sequence of insightful sub-questions. Each sub-question is formulated one at a time, strategically conditioned on the evidence and insights gathered from answers to previous sub-questions directed at the KG. For each such sub-question, the agent then generates precise NL-to-Graph Cypher queries for KG interaction [13, 26]; retrieves relevant information; and performs crucial self-reflection [14, 25] to validate its findings and correct potential errors in its analytical pathway. This translation to Cypher, instead of SQL, is a deliberate choice to leverage the KG's native structure; it allows for more expressive queries on complex operational patterns while avoiding the cumbersome joins typical of SQL on graph-like data, a distinction supported by recent text-to-query benchmarks [34]. This synthesis involves not just presenting raw data but interpreting patterns, identifying anomalies (like bottlenecks within specific warehouse zones or affecting critical operational sequences), and inferring potential root causes based on the relationships and event sequences captured within the KG. This approach moves significantly beyond simple data retrieval towards AI-driven analysis and explanation, vital for informed warehouse planning and decision-making.

This synergistic integration of Knowledge Graphs (KGs) and a reasoning LLM-agent transforms a warehouse Digital Twin (DT) from a predominantly passive simulation environment into an interactive, explainable knowledge base and an intelligent assistant for warehouse planners. Consequently, planners can interact with this enhanced DT using natural language to probe multifaceted operational scenarios, diagnose underlying causes of inefficiencies and complex performance deviations, understand the impact of variability (e.g., in supplier arrivals or equipment uptime), evaluate alternative operational strategies more deeply, and proactively identify potential bottlenecks in proposed warehouse layouts or future operational plans—all without the need to manually decipher voluminous simulation logs or write complex scripts. This heightened transparency and AI-driven decision support significantly improves decision-making agility and the strategic depth of warehouse planning. Our work thus provides a robust and novel methodology to bridge advanced AI with established simulation techniques, paving the way for more effective and intelligent warehouse operational management.

The main contributions of this paper are:

- **Novel Supply Chain Application for Bottleneck Identification from DES:** To the best of our knowledge, this work presents the first application combining Knowledge Graphs (KGs) and Large Language Model (LLM) agents specifically for analyzing output data from Discrete Event Simulations (DES) of warehouse operations to identify bottlenecks and inefficiencies.
- **Bridging Simulation and Generative AI:** We establish a methodological bridge between traditional DES analysis techniques and modern AI capabilities offered by KGs and LLMs, proposing a more powerful and intuitive paradigm for interpreting simulation data.
- **Framework Design:** We detail a comprehensive framework encompassing the ontological construction of a KG from DES output data, and the design of an LLM-agent equipped with a novel iterative reasoning mechanism (featuring sequential, conditioned sub-questioning, Cypher generation for KG interaction, and self-reflection) for effective operational performance analysis and bottleneck diagnosis.
- **Experimental Validation Plan:** We propose and evaluate experiments focused on warehouse simulation scenarios with datasets comprising both operational and investigative questions, designed to validate the effectiveness of our KG+LLM framework in identifying operational bottlenecks and enhancing analytical efficiency compared to established baseline methods.

2 Related Work

2.1 Discrete Event Simulation and Knowledge Graphs in Warehouse Operations

Discrete Event Simulation (DES) is extensively employed in logistics and warehousing to model and analyze diverse operational facets. The outputs from such DES models typically furnish key performance indicators (KPIs) like overall system throughput, queue lengths at different processing stages, waiting times for entities (e.g., orders, products), and the utilization rates of critical resources [5, 20]. In the contemporary Logistics 4.0 landscape, DES is also increasingly recognized as a fundamental component of Digital Twins (DTs). In this role, DES can function as the *cyber twin*, potentially updated with real-time operational data to mirror physical system states and behaviors [2, 22, 32]. A primary analytical objective when working with simulation outputs is the identification of performance bottlenecks. While traditional statistical indicators including average queue lengths, waiting times for entities, and resource utilization rates [5, 20] are valuable for initial assessments and identifying obvious areas of concern, they often provide only surface-level insights. Such methods may not adequately reveal

the underlying root causes of identified bottlenecks, particularly those that emerge from complex interactions between multiple system components or from dynamic and fluctuating operational conditions. Consequently, there is a clear and persistent need for analytical methods that can transcend simple statistical thresholds to offer deeper diagnostic capabilities, thereby enabling a more comprehensive understanding of system inefficiencies and their origins.

Artificial intelligence (AI) is increasingly being leveraged to optimize various facets of warehouse operations, including automation and decision support [8, 16, 27, 35]. Within this trend, Knowledge Graphs (KGs) have emerged as a powerful technology for representing and reasoning over complex, interconnected data in industrial domains [30]. KGs are increasingly utilized for diverse applications such as enhancing operational visibility across supply chains, mapping intricate supplier networks, tracking materials and products, managing operational and supply chain risks, optimizing inventory levels, ensuring product traceability, and monitoring sustainability initiatives [33]. For instance, KGs have been developed to improve robot operations in warehouses [18] and to create digital twin-enabled dynamic spatial-temporal knowledge graphs for optimizing resource allocation in production logistics [40].

While the application of KGs to real-world industrial data spanning supply chains, manufacturing processes, and asset management is rapidly advancing and demonstrating significant value, our review indicates a noticeable gap in the application of KG technology specifically to structure, analyze, and interpret the output data generated from simulations, such as DES. This type of data is critical for effective operational planning and strategic decision-making. Much of the existing work on KGs in industrial contexts focuses on modeling the physical system itself, its components, or real-time data streams from sensors and IoT devices. The opportunity to transform the rich, relational event data and temporal sequences produced by detailed simulation runs into semantically rich KGs for in-depth performance analysis and bottleneck diagnosis remains largely untapped. This represents a significant opportunity to leverage the structural and semantic strengths of KGs for achieving a deeper understanding of simulated systems, thereby enhancing simulation-driven planning and optimization.

2.2 Integrating LLMs and Knowledge Graphs for Industrial Data Analysis

The advent of Large Language Models (LLMs) [39] has introduced transformative capabilities in natural language understanding, generation, and reasoning. The integration of LLMs with KGs is increasingly recognized as a powerful combination [31], creating a synergy that aims to develop AI systems that are both deeply knowledgeable and intuitively conversational. In this paradigm, KGs serve to ground LLMs with factual, structured knowledge, which can help mitigate issues like hallucinations and improve the accuracy and reliability of LLM-generated responses [3]. Conversely, LLMs can make the rich information stored in KGs more accessible to a wider range of users by enabling natural language querying and interaction, abstracting away the need for specialized query languages [42].

Several patterns for integrating KGs and LLMs have emerged. One common approach is KG-enhanced LLMs, where KGs are leveraged either during the LLM's pre-training or, more frequently, at inference time; Retrieval-Augmented Generation (RAG) is a prominent technique in this category, using KGs or other external sources to inform and contextualize LLM generation [29]. Conversely, LLM-augmented KGs employ LLMs to assist in various stages of the KG lifecycle, including construction from unstructured text, knowledge base completion (like link prediction or entity resolution), enrichment of KG embeddings, or generating textual descriptions from graph data (KG-to-text) [31]. A third pattern involves synergized LLMs + KGs, characterized by a deeper, often bidirectional integration, frequently featuring LLM-based agents that can reason over, interact with, and manipulate KGs to perform complex, multi-step tasks [17, 24]; the proposed framework aligns with this synergistic approach.

2.3 KG-LLM Systems in Industrial Applications and the Identified Gap for Simulation Analysis

The synergistic combination of KGs and LLMs is beginning to find applications in various industrial scenarios. For example, researchers have explored integrating KGs and LLMs for enhanced querying in industrial environments [11], using LLMs and context-aware prompting to improve access to manufacturing knowledge [28], and developing LLM-based assistants for warehouse operations as part of broader AI-driven logistics optimization efforts [15]. Furthermore, KG-enhanced LLMs have been proposed for domain-specific question answering systems in technical fields [23].

However, despite these advancements, the application of KG-LLM systems to the unique challenges of analyzing DES output data for operational insights, particularly for complex tasks like iterative bottleneck diagnosis and root cause analysis, remains largely unexplored. While frameworks like SparqLLM [4] have investigated the use of RAG and query templates to improve the reliability of LLM interactions with KGs in industrial settings, the specific application of such LLM+KG agent systems to analyze the unique structure and temporal nature of DES *output data* for operational insights (like bottleneck diagnosis) is largely unexplored. There is a critical need to explore how effectively LLM-based agents, equipped with reasoning capabilities, can:

- Transform complex natural language questions about DES output simulated warehouse performance into precise, executable queries over a KG.
- Iteratively refine their understanding and analytical path based on the evidence retrieved from the KG.
- Synthesize information from disparate parts of the KG to diagnose operational issues and explain their findings in an intelligible manner.

The reliability of LLM-driven query generation, the efficacy of iterative reasoning over simulation-specific KGs, and the overall ability of such integrated systems to provide actionable, explainable insights for DES-based warehouse planning and analysis constitute open research areas that this work aims to address. Our framework specifically focuses on bridging this gap by proposing a novel LLM-based agent that employs an iterative, self-correcting reasoning process over KGs derived from DES outputs to automate and enhance the identification and diagnosis of warehouse inefficiencies.

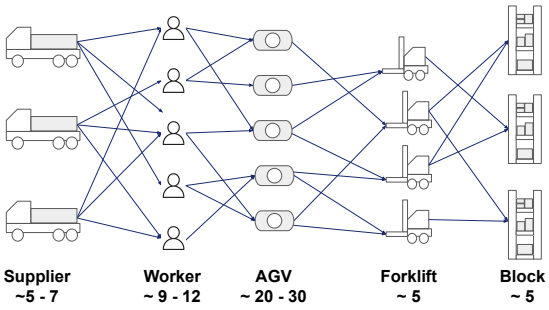

Figure 1: Workflow diagram of the warehouse unloading process, detailing the sequence from supplier deliveries and worker handling, through stages involving Automated Guided Vehicles (AGVs) and forklifts, to packages formed for storage. Numerical figures are for representational purposes only, the exact numbers are provided in text.

3 Discrete Event Simulation – Scenario Design of Warehouse Discharge

This study is based on the data generated by an in-house discrete-event simulation (DES) model that includes operations of a warehouse facility engaged in the unloading, internal transport, and storage of incoming packages. The simulation is designed to replicate real-world warehouse logistics, capturing the interactions between key resources such as suppliers (trucks), workers, automated guided vehicles (AGVs), forklifts, and storage infrastructure. Key highlights are described here, more information can be found in appendix.

3.1 Scenario Configuration

The following sections outline the detailed scenario configuration used in the simulation model. These parameters and design decisions collectively contribute to the fidelity of the simulation to replicate real-world warehouse operations:

3.1.1 Equipment and Resource Specifications:

- **Suppliers:** These are external trucks that bring in the packages to the warehouse. Each supplier holds a total of N packages where N is sampled from a distribution ranging from 30 to 35. They move at a speed of 20 km/hr. A total of five suppliers are expected to arrive in simulation.
- **Workers:** These are 12 employees working at the warehouse. They can move one package at a time at a speed of 2 km/hr.
- **Automated Guided Vehicles (AGVs):** They are automated transporters that carry packages from the worker to the forklift. They can be programmed to traverse prespecified paths. There are 20 AGVs present in the warehouse. They move with a speed of 3.5 km/hr. The Time taken for the AGV to transport a package depends on the distance it travels which is 140 meters on an average.
- **Forklifts:** They are man operated machines that can move packages both horizontally and vertically. They pick up packages from AGVs and place them on a given shelf in the storage block. A forklift can move at speed of 5 km/hr.

3.1.2 Process Flow: The unloading and storage process follows a structured flow:

(1) **Supplier Arrival**: On arrival, a supplier truck heads to the parking area and waits to be assigned to an unloading dock. Once an unloading dock becomes available, the supplier moves towards it and starts unloading.
(2) **Unloading Operation**: Each supplier is assigned a team of four workers. Upon reaching an available unload spot, workers begin transferring packages from the supplier to a pre-defined waiting point.
(3) **Worker to AGV Handoff**: Workers wait for the arrival of an AGV at the waiting point. Upon arrival, the package is loaded onto the AGV and the worker returns to the supplier to repeat the process. The worker handling time is determined by the distance between the supplier and the waiting point, adjusted for the walking speed.
(4) **AGV Transit**: The AGV transports the package to the appropriate block-specific pickup point, with travel time determined by distance and AGV speed.
(5) **Forklift Transfer**: A dedicated forklift on the block collects the package from the AGV and stores it in one of the available bays. The forklift operation time includes the travel time and a stochastic storage time drawn from a distribution ranging between 60 to 90 seconds. The storage space has a cluster of 15 bays with each bay having 3 shelves. Each storage block can store 45 packages. There are five such blocks in the warehouse, making the total capacity 225.

3.1.3 Operational Assumptions:

- Suppliers arrive at regular intervals of 30 minutes, subject to yard capacity constraints (maximum of 3 unloading simultaneously).
- Each worker team of four operates exclusively with its assigned supplier and unload dock during the unloading process.
- AGVs are dynamically dispatched to waiting points and are assigned to packages in a First-In-First-Out (FIFO) manner.
- Forklifts serve incoming packages at the pickup point using a FIFO approach and are restricted to their respective blocks.
- Package handling times for workers and AGVs are determined by distance and speed; storage time for forklifts includes a stochastic component.
- Storage allocation for a block is stochastic.

3.1.4 Data Extracted from Simulation. Following data is captured from each simulation run

- **Process and equipment specific:** The process specific data, including the equipment ID, arrival time, process initiation time, waiting time, process completion time. This includes suppliers, workers, AGVs and forklifts.
- **Package specific:** For each individual package, we capture a unique package ID and log key timestamps throughout the material handling process. These include the time the package is picked up from the supplier, the waiting time at the transfer point, the time the package is loaded onto and departs with an AGV, the time of arrival at the storage block, and the final timestamp when the package is placed into storage.

3.2 Generation of Evaluation Questions

Based on data generated from the normal operating scenario, two types of questions were formulated for the evaluation of analytical capabilities:

3.2.1 Operational Questions: A set of 25 distinct operational questions (see Table 6) was created to assess the proficiency in retrieving specific factual information and performing straightforward analyses using the simulation output. These questions were designed to cover various aspects of the simulated operation, with an approximately uniform distribution across key entities and stages such as supplier interactions, worker activities, AGV and forklift utilization, and package flow.

3.2.2 Investigative Scenarios and Questions for Bottleneck Identification: To specifically evaluate the capabilities in identifying operational bottlenecks, three distinct investigative scenarios were simulated. Each scenario introduced a specific type of inefficiency into the baseline model, mirroring potential real-world disruptions:

- **Scenario 1: Delay in Stage Transfer:** For a particular supplier, a specific process inefficiency was simulated, primarily introducing intermittent delays within the AGV-to-Forklift (AGV-FL) transfer stage, leading to significantly prolonged overall discharge times for their packages.
- **Scenario 2: Supplier-Specific Processing Delay:** For a particular supplier, targeted inefficiencies were simulated, introducing increased handling and suboptimal task allocation within the unloading and package processing stages, leading to significantly prolonged processing times.
- **Scenario 3: Degraded Forklift Performance:** One specific forklift was modeled to operate with reduced efficiency throughout its designated shift, leading to localized congestion and delays in tasks reliant on that particular forklift.

For each of these three systematically perturbed scenarios, a unique investigative question was formulated. The objective of each such question was to task the framework with identifying the primary operational bottleneck or pinpointing the most significant performance degradation resulting from the deliberately introduced inefficiency.

4 Methodology

This section briefly outlines the core technical components of the proposed framework.

4.1 KG Schema Design for DES Data

We utilize a custom KG schema tailored to represent the resources (supplier, worker, AGV, forklift, storage) as nodes and movement of each package between resources as edges of the KG. The operational data including timestamps is added as features of these edges and nodes. The KG is constructed from the output logs generated by the DES model through an automated pipeline. See appendix A.2 for more details.

4.2 LLM Reasoning Agent

The LLM reasoning agent utilizes a dual-path architecture, initiated by query classification, for the complex analysis of operational and investigative questions on the simulation output-derived KG. For operational queries, a QA Chain features a

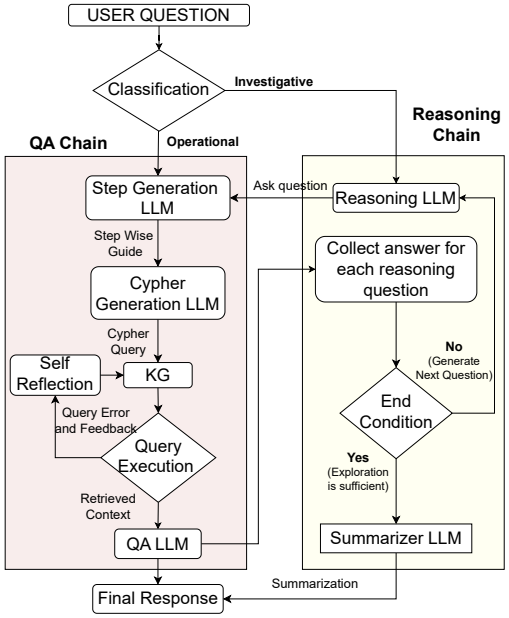

Figure 2: System architecture of the LLM Reasoning Agent comprising two components: the QA Chain and the Reasoning Chain. Queries are first classified as normal or bottleneck. Normal queries follow step-wise guidance for Cypher generation, execution, and self-reflection. Bottleneck queries invoke iterative reasoning, where the agent decomposes the problem into sub-questions, gathers intermediate evidence, and dynamically refines its analysis toward a final answer.

Step Generation module that translates the natural language query into structured steps, often breaking down complex questions into simpler sub-queries, each aimed at extracting relevant information via a single, targeted Cypher query. A Cypher Generation module then formulates these formal queries, which are programmatically executed against the KG by a Query Execution and Correction Module that incorporates an error-handling loop. The Answer Synthesis (QA) module subsequently receives and processes the query results (e.g., list of entities, subgraphs, aggregated values), moving beyond mere data presentation to interpret initial patterns and synthesize coherent answers.

For complex investigative (bottleneck) queries, an Iterative Reasoning Chain is activated: a Reasoning module decomposes the main problem and sequentially generates sub-questions one at a time. Each sub-question then leverages the entire QA Chain—enabling iterative Cypher generation, execution, and focused evidence collection—which allows the agent's overall analytical path to be dynamically refined based on intermediate findings. Upon meeting a sufficiency condition, a Summarizer module performs the final answer synthesis. This stage involves in-depth interpretation of the aggregated graph data to identify patterns indicative of performance issues like bottlenecks (e.g., identifying workstations with consistently high incoming flow but low outgoing flow, correlated with long queue times) and potentially suggesting causal factors based on traversing relationships in the KG (e.g., linking a bottleneck to upstream resource unavailability or specific event sequences thereby delivering a comprehensive diagnostic summary). This architecture

Table 1: Performance on Operational QA by Method and Stage (Pass@k Scores).
Direct QA: Single-pass Cypher query generation followed by answer synthesis. SR: Self-Reflection. Step-wise Guide: Question decomposition for structured step generation; each step involves (Cypher query + Answer Generation + Self-Reflection). P@k indicates Pass@k scores.

Method	Supplier		Worker		AGV		Forklift		Package		Average	
	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4
Direct QA [a]	0.50	0.60	0.40	0.40	0.25	0.60	0.41	0.56	0.41	0.56	0.41	0.56
Direct QA + SR [b]	**0.95**	**1.00**	0.55	0.80	0.60	0.60	0.73	0.80	0.73	0.80	0.73	0.80
Step-wise Guide [c]	0.90	**1.00**	**0.74**	**1.00**	**0.83**	**1.00**	**0.75**	**1.00**	**0.90**	**1.00**	**0.82**	**1.00**

synergizes robust, self-correcting query execution with adaptive, iterative reasoning for advanced diagnostics, where each distinct processing module is realized through an independent Large Language Model call, leveraging either a general-purpose foundation model or a smaller, task-specific fine-tuned model.

5 Results and Discussion

We evaluated our LLM agent framework using OpenAI's GPT-4o (via Langchain QA chains) interacting with a Neo4j knowledge graph through LLM-generated Cypher queries. Interactions were configured with temperature 0.0, top_p 0.95, and a 4096-token limit. While a zero temperature aims for determinism, minor variability can arise from top_p sampling or multi-step reasoning dynamics. For operational queries, we employed a QA chain guided by a step-wise approach that decomposed input questions into structured steps—each involving Cypher generation, KG querying, and self-reflection. Performance was measured using the pass@k metric[6] to assess answer accuracy across 4 attempts. This was benchmarked against two baselines: (i) single-pass Cypher generation with answer synthesis, and (ii) an enhanced version adding post-answer self-reflection. For investigative bottleneck scenarios, our iterative Reasoning chain—refining each step based on accumulated evidence was compared qualitatively against the enhanced baseline and a human expert. Future work will explore additional specialized reasoning models.

5.1 Performance on Operational QA

The experimental results for operational question answering presented in Table 1 highlight the significant advantages of our proposed Guided Iterative Steps approach. While incorporating a self-reflection (SR) mechanism into a direct question-answering pipeline (Baseline: Direct QA + SR) does offer a substantial improvement over a simple, single-pass baseline: Direct QA, our proposed method consistently outperforms both baselines, particularly in achieving comprehensive correctness as indicated by the maximum Pass@4 scores across all operational stages. Qualitative success and failure cases across the approaches are provided in Appendix A.4.

Our proposed technique takes a different approach than the conventional reliance on a single, monolithic Cypher query for operational questions, an approach prone to brittleness due to its complexity. Instead, it introduces a layer based on question decomposition and structured step-wise guidance generation. Rather than attempting to retrieve and synthesize information in one pass, our agent breaks down each query into a sequence of focused analytical steps. Each step involves targeted Cypher query formulation, execution, and an immediate self-reflection phase to assess and refine the output before proceeding. This step-level interaction enables localized error detection and correction, improving both precision and robustness. This approach contrasts sharply with the baselines, where reflection—if present—occurs only after a full KG interaction, limiting its corrective potential. By embedding reflection within each step, our agent incrementally builds understanding, guided by intermediate results and error signals. This not only reduces query complexity but also strengthens the overall reasoning process. The resulting architecture supports more reliable QA and serves as a foundation for the iterative, evidence-driven reasoning required in investigative tasks. The ability to decompose, validate, and refine sub-questions at each stage enables more accurate KG interactions and enhances the agent's capability in both direct question answering and complex diagnostic scenarios.

5.2 Qualitative Analysis on Investigative QA

We present three case studies to evaluate the agent's effectiveness in handling investigative QA. Due to space constraints, only two are discussed in detail, and the third scenario is provided in Appendix A.3.

5.2.1 Scenario 1: The first study examined why a supplier, *CamelCargo*'s discharge was significantly delayed. The human expert identified a critical symptom: a 38-minute delay at the AGV stage for the final package. When the same question was posed to the baseline method, it broadly attributed delays to varying times at each stage, mentioning the AGV and forklift but failing to isolate or quantify the main bottleneck. In contrast, our LLM-agent (process detailed in Table 2) validated the expert's observation with precise, data-driven analysis. It first confirmed the overall delay (6,848s vs. a 4,934s average), then, through sub-questioning, identified the "AGV to FL" transfer as the key issue—highlighting extreme variability and delays. Further analysis of related factors (e.g., normal AGV wait times, localized forklift delays, typical utilization rates) led to a robust conclusion: the bottleneck stemmed from inefficiencies in the AGV-to-FL process. This demonstrates the agent's strength in not only aligning with expert intuition but also delivering a more precise and comprehensive diagnosis than the baseline.

5.2.2 Scenario 2: The second study addressed a slowdown in warehouse discharge operations observed between 10:00 AM and 12:30 PM. The human expert, analyzing detailed data, noted that AGV operational times seemed longer for most packages between 10:30 AM and 11:11 AM, but was not conclusive whether this was solely due to AGVs or potentially linked to specific workers or forklifts, though other resource timings appeared normal. The baseline approach provided a very general explanation for the overall slowdown in the 10:00 AM - 12:30 PM window, attributing it to the average operational durations of

Table 2: Case Study 1: Why did CamelCargo discharge take longer than usual?

Note: The LLM Agent formulates its own sequence of questions based on its reasoning framework and evolving evidence from the KG. Human expert iterative questions for this investigation: (1) What was the difference in discharge time of CamelCargo? (2) What was the average discharge time of other suppliers? (3) How many packages needed to be unloaded by CamelCargo? (4) What was the average time taken per package in terms of worker operation time, AGV arrival time, AGV operation time, and forklift operation time? (5) Calculate derived metrics such as waiting time (AGV end time - forklift start time) (6) Did any package deviate from the average time? If yes, in which category? The LLM generations have been condensed to fit inside the table. Cypher queries shown are highly condensed conceptual representations for brevity.

LLM Agent - Iterative Question	LLM Agent - Plan (Condensed)	KG ResultSet (Condensed)	Cypher Query (Condensed)
What is the total unload time for CamelCargo compared to the global average unload time for all suppliers in the Discharge Flow?	Calculate CamelCargo's unload duration (discharge_start to discharge_end) and compare with global average for all suppliers.	CamelCargo Unload Time: 6848s. Global Avg. Unload Time: 4933.8s.	`MATCH (s:SUPPLIER {supplier_id:'CamelCargo'}) RETURN duration.between(s.discharge_start, s.discharge_end); MATCH (s:SUPPLIER) RETURN avg(duration.between(s.discharge_start, s.discharge_end))`
What is the breakdown of time taken for each stage of the unload process for CamelCargo in the Discharge Flow, and how does it compare to the global average for each stage?	Calculate durations for Worker-to-AGV, AGV-to-FL, FL-to-Storage stages for CamelCargo's packages; compare with global averages for these stages.	Worker-AGV: 58s (matches global avg). AGV-FL (CamelCargo): Highly variable (e.g., one 2300s), many > global avg 422.6s. FL-Storage: Near global avg 116.4s.	`MATCH (s:SUPPLIER {id:'CC'})-[*]->(st:STORAGE) RETURN duration(worker_stage) AS t_ws, duration(agv_stage) AS t_as, ...; MATCH path RETURNavg(t_ws),avg(t_as)...`
What are the waiting times for AGVs and forklifts during the CamelCargo unload process in the Discharge Flow, and how do they compare to the global average waiting times for these equipments?	Calculate AGV waiting (e.g., worker pick-up end - AGV arrival) and Forklift waiting (e.g., AGV journey end - FL placement start) for CamelCargo; compare to global averages.	Avg. AGV Waiting (CamelCargo): Neutral (12s). Avg. FL Waiting (CamelCargo): Positive (60.6s, indicating delays).	`MATCH (s:SUPPLIER {id:'CC'})-...->(a:AGV) RETURN duration.between(wta.agv_arrival, wta.worker_end) AS agv_w;...avg(agv_w)...`
What is the utilization rate of forklifts during the CamelCargo unload process in the Discharge Flow, and how does it compare to the global average utilization rate for forklifts?	Determine FL utilization (total processing time / total active time) for each forklift with CamelCargo packages; compare to global average FL utilization.	Individual FL utilization rates for CamelCargo (e.g., FL_00: 41.2%) generally matched their respective global average utilization rates.	`MATCH (a:AGV)-[]->(fl:FL)-[]->... RETURN fl.id, SUM(dur_proc) / dur_active AS util; ...avg(util)`
What is the utilization rate of AGVs during the CamelCargo unload process in the Discharge Flow, and how does it compare to the global average utilization rate for AGVs?	Determine AGV utilization (total processing time / total active time) for each AGV handling CamelCargo packages; compare to global average AGV utilization.	AGV utilization rates varied (e.g., AGV_00: 38.1%, AGV_10: 55.1%); some higher/lower than an estimated global average of 41.4%.	`MATCH (w:WORKER)-[]->(a:AGV)-[]->... RETURN a.id, SUM(dur_proc) / dur_active AS util`

Agent Summary: The **'AGV to FL' stage** for CamelCargo exhibited significant delays (specific instances up to 2300s versus 423s average for this stage) and high variability, which was the main contributor to its extended total unload time (6848s vs. 4934s global average). Forklift waiting times also indicated related delays. Other stages and overall equipment utilization rates were found to be less critical, pointing towards process inefficiencies primarily within the AGV-FL transfer.

workers (58s), AGVs (474s), and forklifts (118s) without identifying any specific entity or cause for deviation. In contrast, our iterative LLM agent (Table 3) systematically diagnosed the issue within the given timeframe. It first identified that 'AuroraFarms' had a significantly longer total unload time (8,896s) compared to other suppliers and the period's global average (6,904s). Subsequent investigation revealed that AuroraFarms, along with 'BlackSheepDist', also exhibited higher average package processing durations. Crucially, the agent pinpointed inefficient worker and AGV utilization linked to AuroraFarms (e.g., some worker utilization as low as 2.6% and high AGV utilization peaks suggesting bottlenecks) as key contributing factors, while ruling out initial supplier waiting times. This allowed the agent to determine that the slowdown within the specified timeframe was primarily driven by inefficiencies related to a specific supplier, AuroraFarms, particularly concerning their package processing throughput and associated resource utilization, a far more precise and actionable insight than either the human expert's localized AGV observation or the baseline's generic summary.

5.3 Discussion on relevance for warehouse planning

Our framework demonstrates significant relevance as a planning assistant across multiple horizons of warehouse operations. Its high pass@k scores on diverse operational queries (Table 1) enables planners to obtain precise, real-time visibility into supplier interactions, resource utilization, and package flow—supporting both reliable daily control and agile tactical adjustments. More importantly, the investigative case studies highlight the framework's ability to move beyond surface-level reporting toward meaningful diagnostic insight. By systematically querying a simulation-derived KG using an LLM-driven reasoning process, the agent effectively isolates root causes of performance issues, revealing subtle bottlenecks and interdependencies often missed by traditional analytics. This fusion of DES with GenAI methods offers a more powerful and interpretable warehouse digital twin. As a result, planners are better equipped to make targeted, data-driven interventions—whether through process redesign, resource reallocation, or supplier strategy refinement—ultimately enabling more adaptive, efficient, and informed warehouse planning.

6 Implications and Limitations

This work marks a significant step towards automating the intricate analysis of DES outputs, offering warehouse planners a potent tool for rapid diagnostic insights via natural language. However, the current study has certain limitations. While our proposed pipeline facilitates KG construction, the initial design of a comprehensive KG schema tailored to specific DES model

Table 3: Case Study 2: Why was the discharge slow from 10 to 12.30?

Note: Human expert iterative questions for this investigation: (1) What was the difference in discharge time of suppliers between 10:00 AM and 12:30 PM? (2) What was the average discharge time of other suppliers during this period? (3) What was the average time taken per package across key operations (worker, AGV, forklift) within this timeframe? (4) Calculate derived metrics such as waiting times for equipment or packages. (5) Did any package deviate significantly from average processing times? (6) Do these deviations fall within the 10:00 AM - 12:30 PM window, and which operational categories are primarily responsible? The LLM generations have been condensed to fit inside the table. Cypher queries are highly condensed conceptual representations for brevity.

LLM Agent - Iterative Question	LLM Agent - Plan (Condensed)	KG ResultSet (Condensed)	Cypher Query (Condensed)
What was the total unload time for each supplier from 10:00 AM to 12:30 PM, and the global average?	Calculate unload duration (discharge_end - discharge_start) for each supplier active in the specified timeframe; compute global average.	AuroraFarms: 8896s; BlackSheepDist: 6713s; CamelCargo: 5104s. Global Avg. (in timeframe): 6904.3s.	`MATCH (s:SUPPLIER) WHERE s.discharge_start/end IN [timeframe] RETURN s.id, duration(...);...avg(duration)`
What were the package waiting times for each supplier from 10:00 AM to 12:30 PM, and the global average?	Calculate average total processing duration for packages per supplier active in timeframe (summing key stage durations); compute global average.	Avg. Pkg. Processing Duration - AuroraFarms: 760.1s; BlackSheepDist: 746.9s; CamelCargo: 564.0s. Global Avg: 689.9s.	`MATCH (s:SUPPLIER)-[*]->(st:STORAGE) WHERE s.discharge IN [timeframe] RETURN s.id, avg(total_pkg_proc_time);...avg(...)`
What was the utilization rate of workers and AGVs for each supplier from 10:00 AM to 12:30 PM, and the global average?	Determine worker and AGV utilization (processing time / active time) per supplier in timeframe; compute global averages.	AuroraFarms: Varied worker util. (some low, e.g., 2.6%), varied AGV util. (some high, e.g., 86%). Other suppliers also varied.	`MATCH (s)-[]->(w:WORKER), (s)-[]->(a:AGV) WHERE s.discharge IN [timeframe] RETURN s.id, util(w), util(a); ...avg(...)`
What was the supplier waiting time for each supplier from 10:00 AM to 12:30 PM, and the global average?	Calculate supplier waiting (discharge_start - arrival_time) for suppliers arriving in timeframe; compute global average.	Main suppliers (Aurora, BlackSheep, CamelCargo): 0s waiting. DeltaDrops: 52s. Global Avg: 15.4s.	`MATCH (s:SUPPLIER) WHERE s.arrival_time IN [timeframe] RETURN s.id, duration.between(s.arrival, s.discharge_start);...avg(...)`
What was the breakdown of time taken for each stage of the package unload process for each supplier from 10:00 AM to 12:30 PM, and the global average?	Further dissect stage durations (Worker-AGV, AGV-FL, FL-Storage) for each supplier in timeframe; compare to global stage averages to identify specific inefficiencies per supplier.	(Detailed stage breakdown per supplier, e.g., AuroraFarms' AGV-FL stage contributing significantly to its high package processing time.)	`MATCH (s:SUPPLIER)-[stages...]-> (st:STORAGE) WHERE s.discharge IN [timeframe] RETURN s.id,avg(dur_stage1), avg(dur_stage2)...`

Agent Summary: Operations related to supplier **'AuroraFarms'** were the main driver. This was evidenced by AuroraFarms significantly longer total unload time (8896s vs. ~6904s global average for the period) and higher average package processing/waiting durations (~760s vs. ~690s global average). Contributing factors included inefficient worker (some instances as low as ~2.6%) and variable AGV utilization (some instances as high as ~86%) associated with AuroraFarms' packages during this timeframe. Initial supplier waiting times for key suppliers were not a factor.

outputs requires careful upfront domain expertise and engineering effort. Furthermore, although the LLM agent with its self-correction mechanisms performed robustly, the absolute reliability of LLM-generated Cypher queries and the nuanced accuracy of its synthesized explanations warrant ongoing evaluation, particularly when faced with highly novel or ambiguous operational scenarios not extensively represented within the KG's current scope (derived from the simulated data). The generalizability of the specific KG schema and agent fine-tuning has primarily been validated within the described warehouse unloading context, and its seamless applicability to vastly different DES models or a broader array of warehouse processes is yet to be exhaustively demonstrated.

7 Conclusion and Future Work

Extracting actionable insights from the complex and voluminous data generated by Discrete Event Simulations poses a significant challenge to timely and effective decision-making in warehouse operations. To address this, we proposed a novel framework that integrates Knowledge Graphs with a reasoning-capable LLM agent, offering a more intuitive and powerful means of interacting with simulation data. The architecture combines a QA chain with step-wise guidance and an iterative reasoning chain equipped with sub-questioning, Cypher query generation, and self-reflection. This enables both high-accuracy responses to operational queries and deeper, evidence-driven investigations into system inefficiencies. Experimental evaluations demonstrate this framework's proficiency in accurately answering operational questions and, more significantly, its robust capability in performing iterative, evidence-driven investigations to identify operational bottlenecks within simulated scenarios, surpassing traditional baseline methods.

Looking ahead, several exciting avenues for future research emerge. Firstly, we plan to explore the integration and performance of other advanced reasoning-focused Large Language Model architectures or emerging state-of-the-art alternatives to potentially enhance the agent's diagnostic depth and efficiency. Secondly, to further validate and demonstrate the framework's robustness, we will focus on expanding its application to a wider array of warehouse operations beyond unloading—such as slotting design, order picking, loading, and inventory management—which will inherently involve generating more diverse and complex simulated scenarios tailored to these new contexts. This expansion will also necessitate developing rigorous benchmarking methodologies for its investigative question-answering capabilities to formally quantify performance in bottleneck identification tasks across these varied settings. Such extensions will allow for a thorough assessment of the framework's adaptability and utility across a broader spectrum of logistics challenges, including the potential for analyzing larger-scale supply chain simulations.

Acknowledgments

This work is supported by Phi Labs, Quantiphi Inc. We would like to thank Dr. Dagnachew Birru and Mr. Asif Hasan for their continued support.

References

[1] Bilal Abu-Salih. 2021. Domain-specific knowledge graphs: A survey. *Journal of Network and Computer Applications* 185 (2021), 103076.

[2] K. Agalianos, S.T. Ponis, E. Aretoulaki, G. Plakas, and O. Efthymiou. 2020. Discrete Event Simulation and Digital Twins: Review and Challenges for Logistics. *Procedia Manufacturing* 51 (2020), 1636–1641. doi:10.1016/j.promfg.2020.10.228 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).

[3] Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2023. Can knowledge graphs reduce hallucinations in llms?: A survey. *arXiv preprint arXiv:2311.07914* (2023).

[4] Marco Arazzi, Davide Ligari, Serena Nicolazzo, and Antonino Nocera. 2025. Augmented Knowledge Graph Querying leveraging LLMs. *arXiv preprint arXiv:2502.01298* (2025).

[5] Jerry Banks. 2005. *Discrete event system simulation.* Pearson Education.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[7] René De Koster, Tho Le-Duc, and Kees Jan Roodbergen. 2007. Design and control of warehouse order picking: A literature review. *European journal of operational research* 182, 2 (2007), 481–501.

[8] Adnane Drissi Elbouzidi, Abdessamad Ait El Cadi, Robert Pellerin, Samir Lamouri, Estefania Tobon Valencia, and Marie-Jane Bélanger. 2023. The Role of AI in Warehouse Digital Twins: Literature Review. *Applied Sciences* 13, 11 (2023). https://www.mdpi.com/2076-3417/13/11/6746

[9] Jinxiang Gu, Marc Goetschalckx, and Leon F McGinnis. 2007. Research on warehouse operation: A comprehensive review. *European journal of operational research* 177, 1 (2007), 1–21.

[10] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. 2024. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680* (2024).

[11] Domen Hočevar and Klemen Kenda. 2024. *Integrating Knowledge Graphs and Large Language Models for Querying in an Industrial Environment.* Ph. D. Dissertation. Bachelor's Thesis. University of Ljubljana.

[12] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, et al. 2021. Knowledge graphs. *ACM Computing Surveys (Csur)* 54, 4 (2021), 1–37.

[13] Markus Hornsteiner, Michael Kreussel, Christoph Steindl, Fabian Ebner, Philip Empl, and Stefan Schönig. 2024. Real-Time Text-to-Cypher Query Generation with Large Language Models for Graph Databases. *Future Internet* 16, 12 (2024), 438.

[14] Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022. Large language models can self-improve. *arXiv preprint arXiv:2210.11610* (2022).

[15] Saverio Ieva, Ivano Bilenchi, Filippo Gramegna, Agnese Pinto, Floriano Scioscia, Michele Ruta, and Giuseppe Loseto. 2025. Enhancing Last-Mile Logistics: AI-Driven Fleet Optimization, Mixed Reality, and Large Language Model Assistants for Warehouse Operations. *Sensors* 25, 9 (2025), 2696.

[16] Dmitry Ivanov, Alexandre Dolgui, and Boris Sokolov. 2019. The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. *International journal of production research* 57, 3 (2019), 829–846.

[17] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. 2024. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. *arXiv preprint arXiv:2402.11163* (2024).

[18] Ajay Kattepur and Balamuralidhar P. 2019. Roboplanner: autonomous robotic action planning via knowledge graph queries. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 953–956.

[19] Edward Elson Kosasih, Fabrizio Margaroli, Simone Gelli, Ajmal Aziz, Nick Wildgoose, and Alexandra Brintrup. 2024. Towards knowledge graph reasoning for supply chain risk management using graph neural networks. *International Journal of Production Research* 62, 15 (2024), 5596–5612.

[20] Averill M Law, W David Kelton, and W David Kelton. 2000. *Simulation modeling and analysis.* Vol. 3. Mcgraw-hill New York.

[21] Carman KM Lee, Yaqiong Lv, Kam KH Ng, William Ho, and King Lun Choy. 2018. Design and application of Internet of things-based warehouse management system for smart logistics. *International Journal of Production Research* 56, 8 (2018), 2753–2768.

[22] Jiewu Leng, Hao Zhang, Douxi Yan, Qiang Liu, Xin Chen, and Ding Zhang. 2019. Digital twin-driven manufacturing cyber-physical system for parallel controlling of smart workshop. *Journal of ambient intelligence and humanized computing* 10 (2019), 1155–1166.

[23] Donghe Li, Zuchen Li, Ye Yang, Li Sun, Dou An, and Qingyu Yang. 2024. Knowledge graph-enhanced large language model for domain-specific question answering systems. *Authorea Preprints* (2024).

[24] Linhao Luo, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Reasoning on graphs: Faithful and interpretable large language model reasoning. *arXiv preprint arXiv:2310.01061* (2023).

[25] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems* 36 (2023), 46534–46594.

[26] Ioanna Mandilara, Christina Maria Androna, Eleni Fotopoulou, Anastasios Zafeiropoulos, and Symeon Papavassiliou. 2025. Decoding the Mystery: How can LLMs Turn Text into Cypher in Complex Knowledge Graphs? *IEEE Access* (2025).

[27] Hokey Min. 2010. Artificial intelligence in supply chain management: theory and applications. *International Journal of Logistics: Research and Applications* 13, 1 (2010), 13–39.

[28] Sebastian Monkaa, Irlan Grangel-Gonzáleza, Stefan Schmida, Lavdim Halilaja, Marc Rickartb, Oliver Rudolphb, and Rui Diasb. [n. d.]. Enhancing Manufacturing Knowledge Access with LLMs and Context-aware Prompting. ([n. d.]).

[29] I Muneeswaran, Advaith Shankar, V Varun, Saisubramaniam Gopalakrishnan, and Vishal Vaddina. 2024. Mitigating Factual Inconsistency and Hallucination in Large Language Models.. In *WSDM*. 1169–1170.

[30] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale Knowledge Graphs: Lessons and Challenges: Five diverse technology companies show how it's done. *Queue* 17, 2 (2019), 48–75.

[31] Jeff Z Pan, Simon Razniewski, Jan-Christoph Kalo, Sneha Singhania, Jiaoyan Chen, Stefan Dietze, Hajira Jabeen, Janna Omeliyanenko, Wen Zhang, Matteo Lissandrini, et al. 2023. Large language models and knowledge graphs: Opportunities and challenges. *arXiv preprint arXiv:2308.06374* (2023).

[32] Adil Rasheed, Omer San, and Trond Kvamsdal. 2020. Digital twin: Values, challenges and enablers from a modeling perspective. *IEEE access* 8 (2020), 21980–22012.

[33] Chaouki Saidi, Nadia Hamani, Mounir Benaissa, Benjamin Rolf, Tobias Reggelin, and Sebastian Lang. 2025. Modeling reconfigurable supply chains using knowledge graphs: towards Supply Chain 5.0. *Production Engineering* (2025), 1–24.

[34] Sithursan Sivasubramaniam, Cedric E Osei-Akoto, Yi Zhang, Kurt Stockinger, and Jonathan Fürst. 2024. Sm3-text-to-query: Synthetic multi-model medical text-to-query benchmark. *Advances in Neural Information Processing Systems* 37 (2024), 88627–88663.

[35] Enoch Oluwademilade Sodiya, Uchenna Joseph Umoga, Olukunle Oladipupo Amoo, and Akoh Atadoga. 2024. Ai-driven warehouse automation: a comprehensive review of systems. GSC Advanced Research and Reviews, 18 (2), 272-282.

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837. https://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html

[37] Yuchen Xia, Daniel Dittler, Nasser Jazdi, Haonan Chen, and Michael Weyrich. 2024. LLM experiments with simulation: Large Language Model Multi-Agent System for Simulation Model Parametrization in Digital Twins. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 1–4.

[38] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629* (2022). doi:10.48550/arXiv.2210.03629

[39] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* 1, 2 (2023).

[40] Zhiheng Zhao, Mengdi Zhang, Jian Chen, Ting Qu, and George Q Huang. 2022. Digital twin-enabled dynamic spatial-temporal knowledge graph for production logistics resource allocation. *Computers & Industrial Engineering* 171 (2022), 108454.

[41] Ray Y Zhong, Xun Xu, Eberhard Klotz, and Stephen T Newman. 2017. Intelligent manufacturing in the context of industry 4.0: a review. *Engineering* 3, 5 (2017), 616–630.

[42] Yunqi Zou, Yongli Wang, and Dongmei Liu. 2024. Q2Cypher: Converting Natural Language Questions to Cypher with Fine-Tuned Large Language Models. In *2024 5th International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*. IEEE, 783–788.

A Appendix

A.1 Warehouse Resources

The Table 4 highlights the various resource types and their respective ids that are modeled in the simulation.

Table 4: The resource ids for the different resource present in the simulated scenario

Resource Type	Resource IDs
Supplier	AuroraFarms, BlackSheepDist, CamelCargo, DeltaDrops, EvergreenEdge
Worker	BW_02, BW_00, BW_01, BW_03, BW_09, BW_10, BW_11, BW_08, BW_05, BW_07, BW_06, BW_04
AGV	AGV_10, AGV_12, AGV_11, AGV_00, AGV_01, AGV_02, AGV_03, AGV_13, AGV_14, AGV_04, AGV_15, AGV_05, AGV_16, AGV_08, AGV_17, AGV_09, AGV_07, AGV_18, AGV_19, AGV_06
Fork Lift	FL_00, FL_01, FL_04, FL_02, FL_03
Storage Block	A, B, C, D

A.2 KG Schema:

Node Properties:

- **SUPPLIER:**
  `supplier_id: STRING, arrival_time: DATETIME, discharge_start:DATETIME, discharge_end: DATETIME`
- **WORKER:**
  `worker_id: STRING`
- **AGV:**
  `agv_id: STRING`
- **FL (Forklift):**
  `forklift_id: STRING`
- **STORAGE:**
  `block_id: STRING`

Relationship Properties:

- **SUPPLIER_TO_WORKER**
  `package_id:STRING, worker_pick_up_start:DATETIME`
- **WORKER_TO_AGV**
  `package_id: STRING, agv_arrival:DATETIME, agv_journey_start:DATETIME, worker_pick_up_end: DATETIME`
- **AGV_TO_FL**
  `package_id: STRING, agv_journey_end: DATETIME, fl_placement_start: DATETIME`
- **FL_TO_STORAGE**
  `package_id: STRING, fl_placement_end: DATETIME`

A.3 Investigative Case Studies - Scenario 3

A.3.1 Scenario 3: Forklift Waiting Times and Connection to Discharge Flow. In the third investigative case study, which implicitly probed for forklift-induced delays, the human expert insightfully identified Forklift FL_00 (and an associated block) as being linked to above-average AGV differences or wait times, though with some initial uncertainty about whether AGVs, workers, or forklifts were solely responsible. The baseline Direct QA + Self-Reflection method also flagged FL_00 for having the highest average waiting time (333s) but offered limited further diagnosis beyond noting variations among forklifts. Our method (Table 5), through its sequence of targeted inquiries, not only strongly corroborated FL_00's problematic status but also provided a more comprehensive diagnostic picture. The agent confirmed FL_00's exceptionally high average waiting time and, crucially, also identified its significantly longer package movement time from AGV to storage (152s vs. 123s global average). While also exploring AGV performance variability, the agent's overall diagnosis clearly centered on FL_00's dual inefficiency (high waiting and long task execution) as a key contributor to delays, thus offering a more multi-faceted and data-rich explanation than the baseline and a more conclusive identification of forklift-specific issues compared to the human expert's initial broader considerations.

A.4 Operational QA - Success and Failure Cases

We present selected query instances from Operational QA to qualitatively compare our proposed method against baseline approaches. These examples highlight specific successes (marked in green) and failures (marked in red), offering deeper insight into the practical performance of each technique.

Case 1: Baseline (Direct QA) Partial Failure, Other Methods Pass

Question: Which supplier had the shortest total discharge time and how many packages were moved?

Direct QA: The supplier "EvergreenEdge" had the shortest total discharge time, which was 3562 seconds. A total of 1 package was moved.

Direct QA + SR: The supplier with the shortest total discharge time is EvergreenEdge, and they moved 33 packages.

Step-wise Guide: The supplier with the shortest total discharge time is EvergreenEdge, with a discharge time of 3562 seconds. They moved a total of 33 packages.

In this instance, the query required retrieving two distinct pieces of information (the supplier with the shortest discharge time and their corresponding package count) and ensuring their correct association. The 'Direct QA' baseline successfully identified the correct supplier and their discharge time but faltered in accurately retrieving the associated package count, incorrectly stating *1 package*. This likely indicates an inability of the single-pass query generation to correctly formulate or execute a Cypher query complex enough to link these two facts or perform the necessary aggregation accurately. The introduction of self-reflection in the 'Direct QA + SR' baseline enabled it to overcome this error, suggesting the reflection process could validate and correct the factual components of the synthesized answer. Our 'Step-wise Guide' approach also provided the correct and complete answer. Its structured approach, which can decompose the query into logical steps (e.g., one for identifying the supplier with the shortest time, followed by a step to retrieve the package count specifically for that identified supplier) and validate each part, inherently leads to higher accuracy for such multi-fact retrieval questions.

Case 2: Both Baselines Fail, Step-wise Guide Succeeds

Question: What is the average travel time for an AGV to move a package from the dock to its assigned storage area?

Table 5: Case Study 3: What do the differences in forklift waiting times reveal about the discharge flow?

Note: The LLM Agent formulates its own sequence of questions. Potential human expert iterative questions for a forklift delay investigation might include: (1) KPIs per forklift (tasks, durations, idle times)? (2) Which forklifts show consistently longer task/idle times? (3) Operational time breakdown for suspect forklifts? (4) Queue/wait times for packages per forklift? (5) Are delays linked to specific zones/tasks/times for suspect forklifts? (6) How does a suspect forklift's throughput compare? (7) Any error codes/issues reported? Cypher queries are highly condensed conceptual representations.

LLM Agent - Iterative Question	LLM Agent - Plan (Condensed)	KG ResultSet (Condensed)	Cypher Query (Condensed)
What is the average forklift waiting time and utilization across all blocks during the discharge flow?	Calculate FL waiting (AGV journey_end - FL placement_start)	FL utilization (active_time / total_span) per FL. & FL_00: AvgWait 332.9s, Util 0; FL_01: AvgWait 48.9s, Util 0; FL_04: AvgWait 36.3s, Util 0;	`MATCH (a:AGV)-[]->(fl:FL)-[]->(st:S) RETURN fl.id, avg(wait_time), SUM(active_FL)/total_FL_span AS util`
What are the average package waiting times at the waiting point for each AGV during the discharge flow, and how do these compare to the global average?	Calculate package waiting time for AGV arrival (AGV_arrival - worker_pickup_end) per AGV; compare to global average.	Highly variable: e.g., AGV_10: 3.6m; AGV_08: 10.3m. Global Avg: 5.5m. (Indicates some AGVs contribute to or experience more package wait).	`MATCH (w)-[wta:WTA]->(a:AGV) RETURN a.id, avg(dur(wta.worker_end, wta.agv_arrival)).min UNION ... global_avg`
What is the average time taken by each forklift to move a package from the AGV to the storage bay, and how does this compare to the global average?	Calculate FL movement time (FL_placement_end - FL_placement_start) per FL; compare to global average and show difference.	FL_00: AvgMove 152.2s (28.9s > global avg 123.3s). Others (FL_01, FL_02, FL_03, FL_04) were below global average.	`MATCH (a)-[]->(fl:FL)-[]->(st:S) RETURN fl.id, avg(FL_move) AS t1; ... global_avg AS t2 RETURN fl.id, t1, t2, t1-t2`
What is the average time taken by each AGV to transport a package from the waiting point to the pickup point, and how does this compare to the global average?	Calculate AGV transport time (AGV_journey_end - AGV_journey_start) per AGV; compare to global average.	Variable AGV performance: Some AGVs (e.g., AGV_07, AGV_09) faster than 8.7s global avg; others (e.g., AGV_00, AGV_08) slower.	`MATCH (w)-[wta:WTA]->(a)-[atf:ATF]->(fl) RETURN a.id, avg(dur(wta.agv_start, atf.agv_end)); ...global_avg`

Agent Summary: Forklift FL_00 is the primary bottleneck within the discharge flow, evidenced by its highest average waiting time (approximately 333 seconds), substantially longer than other forklifts, and also by it taking significantly more time to move packages from AGVs to storage (around 152 seconds, which is about 29 seconds above the global average for this task). While variability was also observed in AGV performance and in total unload times among different suppliers, addressing the specific operational challenges and inefficiencies tied to FL_00 can help improve the overall efficiency of the discharge process.

Direct QA: The average travel time for an AGV to move a package from the dock to its assigned storage area is approximately 178 seconds.

Direct QA + SR: The average travel time for an AGV to move a package from the dock to its assigned storage area is approximately 178.14 seconds.

Step-wise Guide: The average travel time for an AGV to move a package from the dock to its assigned storage area is 455 seconds.

This case highlights a scenario where both baseline methods failed to determine the correct numerical value for a key performance indicator. Their answers (178s and 178.14s) were significantly different from the correct value (455s) obtained by the Step-wise Guide. This discrepancy suggests that the baselines likely misinterpreted the scope of the AGV travel (e.g., focusing on a shorter segment of the journey, incorrectly identifying the start/end events for the "dock to storage" path, or using an incorrect aggregation method over the KG). The Direct QA + SR baseline, while providing a slight numerical variation, essentially repeated the same fundamental error as the Direct QA, indicating that its self-reflection mechanism was insufficient to catch this type of error in calculation logic or path identification. The success of the proposed method implies that its structured decomposition of the query into smaller, verifiable steps allowed it to accurately identify the relevant AGV journey segments (from dock to storage for each package) and correctly aggregate these travel times from the KG to compute the true average.

Case 3: Step-wise Guide Fails, Baseline (QA + SR) Succeeds

Question: For each person, what was the total number of packages they handled during a shift?

Direct QA: Syntax Error

Direct QA + SR: Worker BW_11 handled 13 packages, Worker BW_10 handled 19 packages, Worker BW_09 handled 20 packages, Worker BW_12 handled 18 packages, Worker BW_05 handled 9 packages, Worker BW_08 handled 8 packages, Worker BW_06 handled 9 packages, Worker BW_07 handled 9 packages, Worker BW_02 handled 13 packages, Worker BW_00 handled 14 packages, Worker BW_01 handled 14 packages, Worker BW_03 handled 15 packages, and Worker BW_04 handled 12 packages.

Step-wise Guide Response: The context does not provide any information about the workers or the number of packages they handled during a shift. Therefore, it is not possible to determine the total number of packages handled by each worker.

This case presents an interesting scenario where the Step-wise Guide incorrectly concluded that the information was unavailable, while the Direct QA + SR baseline successfully retrieved the correct data. The Direct QA baseline failed completely with a syntax error, likely struggling to generate the somewhat complex Cypher query required for a group-wise aggregation (summing packages per worker). The Direct QA + SR, however, managed to overcome this, indicating that its combined query generation and self-reflection capability was sufficient for this particular aggregation task. The failure of the Step-wise Guide ("context does not provide information") suggests a potential limitation in its current decomposition strategy or schema interpretation when faced with "for each" type queries requiring specific group-by-and-aggregate operations. It's possible that its step-generation logic broke down the problem in a way that obscured the path to aggregation, or it failed to correctly map person to the Worker entity and their associated package handling events in a way that allowed for summation. This highlights an area for future refinement in the step generation and KG traversal logic within our proposed method to better handle such complex aggregation queries.

Table 6: Set of 25 Operational Questions along with their categorization.

Category	Question
SUPPLIER	What is the number of discharge processes that are completed on a hourly basis? Where and how many containers discharged from supplier DeltaDrops distributed in each block in the storage? Which supplier had the shortest total discharge time and how many packages were moved? What is the average waiting time for a supplier truck before unloading begins? Which truck waited the most? Which hour had the most total waiting time during package unload?
WORKER	For each person, what was the total number of packages they handled during a shift? What is the average time taken by a person to move a package from truck to AGV? Who is the most efficient person ? How much time does each worker take to unload all packages from supplier DeltaDrops? How many workers were used to unload packages from supplier CamelCargo? Which workers were assigned to most number of suppliers?
AGV	Which three AGVs processed the least amount of packages? What is the average travel time for an AGV to move a package from the dock to its assigned storage area? How many trips does each agv make during unloading along with the average journey time? How many packages did AGV 04 handle from each supplier ? Which AGV was the least utilized ?
FORKLIFT	Which package waited the longest for a fork lift ? How many packages are handled by each forklift? Which forklift is the most under utilized ? What is the average time taken by a forklift to move a package to its assigned storage space? What is the utilization rate (percentage of time in use) for each forklift?
PACKAGE	which storage block contains the highest number of containers? What is the average time a package discharge takes? What is the average waiting time for a package to be transferred to a forklift after AGV arrival at the storage area? Which package experienced the longest total time from arrival at the dock to placement in its final storage location? How many packages took longer than the average unload time during and what is the average discharge time? Which packages were handled by both agv 10 and forklift 00?