Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

A FOUNDATIONAL FRAMEWORK FOR GENERATIVE SIMULATION MODELS: PATHWAY TO GENERATIVE DIGITAL TWINS FOR SUPPLY CHAIN

Surdeep Chotaliya[1] , John Fowler[1] , Giulia Pedrielli[1] , David Bayba[2] , Mikayla Norton[2] , Priyanka Sain[2] , and Kaixu Yu[1]

[1]Arizona State University, Tempe, AZ, USA
[2]Intel Corporation, Chandler, AZ, USA

ABSTRACT

Designing high-fidelity simulation models from natural language descriptions remains a significant challenge, particularly in complex domains like supply chains. Pretrained Large Language Models (LLMs), when used without domain-specific adaptation, often fail to translate unstructured inputs into executable model representations. In this work, we present a fine-tuned LLM-based pipeline specifically tailored for simulation model generation. We construct a high-quality dataset using a novel data generation process that captures diverse supply chain scenarios. The fine-tuned LLM first converts human-like natural language scenario descriptions into structured representations, then translates these into executable code for a modular Python-based simulation engine built to support a wide range of supply chain configurations. Quantitative and qualitative evaluations show that the fine-tuned model consistently generates high-fidelity simulation models, significantly outperforming pretrained LLMs in terms of structural accuracy, simulation behavior, and its ability to robustly extract relevant information from linguistically variable natural language descriptions.

1 INTRODUCTION

Simulation provides a virtual representation of a system's behavior, enabling analysis of its operational dynamics (Banks 2000). In supply chain management, it serves as a critical tool for modeling logistical flows, evaluating uncertainty, identifying inefficiencies, and assessing strategic decisions (Terzi and Cavalieri 2004). Discrete Event Simulation (DES), which models systems through time-ordered events, is particularly suited to supply chains due to its ability to represent asynchronous state changes in interacting entities (Tako and Robinson 2012).

Digital twins represent the next evolution of simulation, enabling continuously updated digital replicas of physical systems. In supply chains, they facilitate real-time monitoring, scenario testing, and predictive analytics by integrating live data streams from Internet of Things (IoT) sensors, (Enterprise Resource Planning (ERP) systems, and operational logs (Agalianos et al. 2020). The emergence of digital twins has expanded simulation's utility by linking virtual models with real-time data from physical systems (Lu et al. 2020). However, building such models remains labor-intensive, requiring deep domain knowledge and significant technical effort.

Recent advances in AI, especially Large Language Models (LLMs) (Chen et al. 2021)—have introduced new possibilities for automating simulation development through natural language understanding and code generation. Yet, LLMs struggle with structured domain logic and deterministic output, limiting their current use in simulation (Jackson et al. 2024).

This study introduces a novel architecture that involves fine-tuning LLMs to generate executable DES models from natural language supply chain scenarios. Using a structured dataset aligned with simulation goals and operational logic, our approach bridges the gap between high-level descriptions and executable models. It reduces development effort and enables seamless integration into digital twin systems:supporting real-time decision-making, predictive analytics, and scalable, adaptive supply chain modeling. For the source code and data, refer to https://github.com/Suri597/SimAI.git.

2 RELATED WORK

2.1 Discrete Event Simulation in Supply Chain

Discrete-event simulation (DES) has been widely applied across diverse supply chain contexts—ranging from construction logistics under demand uncertainty to hybrid EV charging infrastructure and general manufacturing logistics—demonstrating its strength in modeling variability, system performance, and trade-offs in operational design (Vidalakis et al. 2013; Erdeş and Kesen 2025). To enhance scalability, maintainability, and adaptability, researchers have emphasized modularization in DES, with previous work introducing modular frameworks for smart manufacturing, collaborative architectures for distributed execution, and methodologies that decouple model design from implementation by clearly separating the roles of simulation experts and end-users (Longo et al. ; Terzi and Cavalieri 2004; Johansson 2006).

In response to the growing complexity and multi-scale nature of modern supply chains, there is a notable shift toward hybrid simulation approaches. These integrate DES with System Dynamics (SD)(Ogata 2004) and Agent-Based Modeling (ABM)(Macal and North 2005), allowing researchers to capture both granular operational details and emergent, system-level behaviors (Ferreira et al. 2025; D'Ambrosio and Luke). Such hybrid and modular frameworks enable large-scale simulations for disruption analysis and sustainability-oriented decision-making, reflecting an increasing demand for scalable, reusable, and intelligent simulation architectures.

2.2 Digital Twins in Supply Chain

As supply chain systems become increasingly complex and data-driven, the integration of simulation with real-time data and cyber-physical systems has paved the way for digital twin approaches. (Flores-García et al. 2021) provided a foundational definition of digital twins, emphasizing their three core components: physical entities, digital models, and data connections. Building on this, (Eckhart and Ekelhart 2019) explored their role in cyber-physical systems, highlighting their value for resilient production planning and risk mitigation.

Recent advancements in supply chain digital twins can also be found in (van der Valk et al. 2022; Chen and Huang 2021). Despite their potential, a key challenge remains: the lack of automated model generation tools that can adapt as system requirements evolve. The proposed architecture in this paper is a foundational establishment towards ultimately addressing this gap, particularly in the design phase of the digital twin life-cycle, which begins with a well-defined architecture for generative simulation models which are essential for mirroring real-world supply chain policies and operational scenarios with high fidelity.

2.3 Large Language Model for Simulation Modeling

To enable automated and adaptive simulation model generation, we leverage the capabilities of Large Language Models (LLMs), which have shown strong performance in code synthesis, semantic understanding, and system modeling (Achiam et al. 2023; Touvron et al. 2023).

While LLMs are effective across general tasks, domain-specific fine-tuning or pretraining significantly improves their relevance and accuracy for specialized applications (Song et al. 2025). However, applying LLMs to simulation modeling introduces domain-specific challenges due to the need for precise temporal logic, inter-process communication, and domain constraints—factors that general models often struggle to encode without targeted adaptation. Prior studies in AI-assisted supply chain modeling (Nweje and Taiwo 2025) do not focus on automating simulation generation or domain-specific LLM training. The most comparable work by (Jackson et al. 2024) used GPT-3 Codex to generate logistics simulations from natural language but lacked fine-tuning and struggled with scalability in complex scenarios. In contrast, our framework generalizes to dynamic supply chain environments involving multiple raw materials, procurement strategies, and suppliers. The resulting modular, object-oriented Python code-base is designed for extensibility and reuse, with independently modifiable components and encapsulated behaviors that support scalability across multi-actor systems.

2.4 Contributions

We introduce a complete generative simulation architecture that enables LLMs to synthesize executable DES models from natural language descriptions of simulation scenarios. The contributions include:

- A domain-specific data generation pipeline that systematically generates aligned pairs of natural language descriptions and structured representation of supply chain simulation scenarios, enabling the model to learn fine-grained operational semantics and supply chain behaviors.
- We incorporate a modular simulation engine into the architecture and a domain-specific fine-tuned large language model to translate scenario descriptions into executable DES models, ensuring structural validity, domain alignment, and scalability across diverse supply chain configurations.

3 METHODOLOGY

3.1 Overview

The core objective of this work is to enable Large Language Models (LLMs) to autonomously generate high-fidelity, executable Discrete-Event Simulation (DES) models from natural language descriptions of supply chain environments. To this end, we propose an end-to-end framework that transforms user-provided textual input into object-oriented, modular, and standalone Python code for DES modeling.

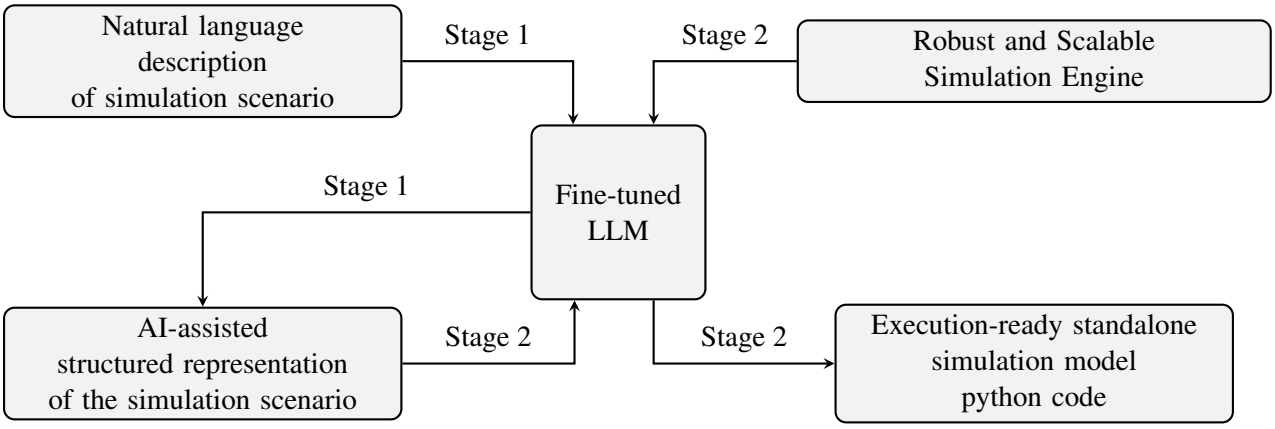


Figure 1: Overview of the fine-tuned LLM-based simulation modeling framework.

Figure 1 outlines the flowchart of the architecture that begins with a high-level domain-specific natural language scenario description. Stage 1 of the framework is to obtain a structured representation of the simulation scenario. These representation encode key operational entities—including raw materials, procurement scheme, suppliers, assemblers, and customers—along with their attributes and interdependencies. The example of the input and output from stage 1 is demonstrated in the following example.

As the below example demonstrates, this phase transforms ambiguous natural language input into structured, machine-readable representations that form the semantic foundation for subsequent simulation generation. The generation process for dataset used to fine-tune the LLM for generating these structured representations is detailed in Section 3.2.

In the second stage, the fine-tuned LLM synthesizes Python-based simulation code by inserting these representations into a predefined, modular framework referred to as the Simulation Engine. This engine is designed to accommodate varying levels of complexity, including multiple raw materials, diverse suppliers, and different procurement strategies. Currently, the framework supports three procurement schemes:

- **Inventory Threshold**: Orders are triggered when inventory falls below a minimum threshold and are refilled to a predefined maximum level.
- **Demand-driven**: Orders are placed based on customer demand rather than inventory levels.

Example: Natural language description to structured representation

Input: Generate a simulation model for a supply chain scenario that involves a single raw material, Silicone Gel, managed under an inventory threshold procurement scheme. The procurement interval for Silicone Gel ranges from 5 to 6 days. The inventory threshold is set at 15 units, meaning that an order is placed to replenish the inventory as soon as it falls below this point, up to the maximum inventory level of 17 units .....:

```
Output -> {"raw_materials": [{"name": "Silicone Gel", "inventory": 5,
"procurement_scheme": "INVENTORY_THRESHOLD","procurement_interval": [5, 6],
"inventory_threshold": 15, "inventory_full": 17, "periodic_supply": 400,
"check_interval_range": [1,2]......
```

- **Periodic Supply**: Materials are delivered in fixed quantities at regular time intervals.

Additionally, the simulation engine performs feasibility checks on the structured representation, with some criteria that may vary depending on the user's specific supply chain context. Examples of feasibility checks include:

- For the *Inventory Threshold* procurement scheme, the inventory threshold must be lower than the maximum inventory level.
- Every finished product must require at least one type of raw material for assembly.
- Every raw material must be linked to at least one supplier.

Upon completion of both stages, the architecture enables a seamless end-to-end workflow in which a descriptive prompt is transformed by the fine-tuned LLM into a structured representation and integrated into the simulation engine, yielding an executable model. Although injecting this structured input does not inherently require LLM assistance, its involvement is critical for establishing a streamlined user interface for post-generation modifications. To accurately interpret and apply natural language corrections or refinements, the LLM must possess an internal understanding of the simulation engine's structure and semantics.

A key challenge in this architecture is enabling the LLM to reliably convert unstructured natural language into structured representations. This requires a carefully curated dataset of aligned input–output pairs, where textual scenarios are mapped to machine-readable representations that encode inventory logic, lead times, supplier roles, assembly rules, etc. Ensuring syntactic validity and semantic coherence is essential for training a model that generalizes across diverse supply chain scenarios and generates structurally sound, high-fidelity simulation models

3.2 Dataset Generation Process

The development of a robust fine-tuning dataset is critical for enabling LLMs to autonomously generate executable simulation models from natural language inputs. The dataset was carefully constructed to ensure semantic coherence, structural validity, and alignment with DES logic in supply chain environments.

Figure 2 illustrates the data generation process designed to produce a high-quality, large-scale dataset for fine-tuning the LLM to translate natural language scenario descriptions into structured representations. The process began with defining a schema that captures key components of a DES-based supply chain simulation, including raw materials, suppliers, procurement strategies, finished products, customer demand profiles, and simulation control parameters.

To populate the schema, logically constrained randomized sampling was used to ensure operational validity. A subset of these structured representations was then manually annotated with natural language descriptions that conveyed both high-level intent and detailed operational logic. These examples were used to prompt the LLM to generate descriptions for the remaining entries, enabling the model to learn human-like language patterns with variability, contextual richness, and fluency. To generate simulation scenario

descriptions spanning a spectrum of linguistic complexity, from accessible, non-technical summaries to highly detailed, expert-level narratives, we also specified expertise levels to the LLM alongside the structured representations.

The generated dataset consist of input-output pairs where:

- Input: Natural language description of a supply chain simulation scenario.
- Output: Structured representation in the form of a Python dictionary.

This dataset was used to fine-tune the model to accurately translate unstructured scenario descriptions into structured simulation inputs. The LLM used for fine-tuning and the fine-tuning process is discussed in the next section.

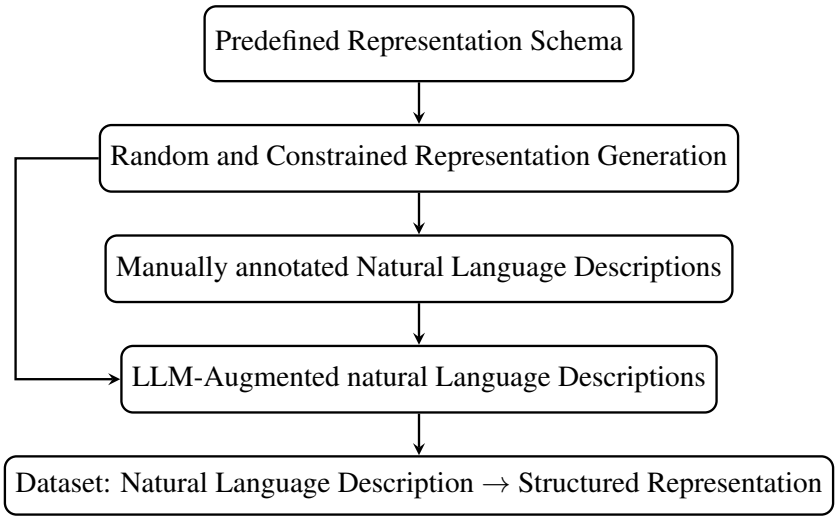


Figure 2: Overview of the dataset generation process.

The primary advantage of this dataset generation architecture lies in its ability to produce an arbitrarily large number of samples, effectively addressing a key challenge in domain-specific fine-tuning of large language models—namely, the scarcity of extensive, high-quality training data (Chan et al. 2024). Following the availability of a large, high-quality dataset, the next critical consideration is the selection of an appropriate Large Language Model (LLM) that aligns with the specific requirements and characteristics of the target task.

3.3 Large Language Model Selection for Fine-Tuning

The selection of a suitable base LLM is a critical design decision in the fine-tuning pipeline, particularly when working with tasks involving long-form structured inputs (e.g., Python dictionaries) and large, executable code blocks as outputs. The task at hand poses unique constraints on both input and output lengths, as well as the model's capacity for syntactic precision and structural consistency. The selection of the Large Language Model (LLM) was based on the following three key criteria:

- **Context Window:** The context window refers to the maximum number of tokens (input and output combined) the model can handle in a single inference. A larger context window supports longer prompts and more complex tasks, such as multi-step reasoning and code generation. Since the proposed work involves generating simulation code as an executable Python script, a minimum context window of 32,000 (32K) tokens is essential.
- **Fine-tuning via API:** Fine-tuning the model via an API allows for efficient customization using task-specific or domain-specific data, without the need for managing the underlying model infrastructure.

This approach is especially beneficial for aligning the model's behavior with specialized goals. Given that the proposed architecture serves as a foundational framework, our priority is to validate its effectiveness before committing resources to building a local fine-tuning setup. Therefore, LLMs that support fine-tuning through an API are more suitable for our current needs.

- **Fine-tuning Cost:** The fine-tuning cost typically include a one-time training fee based on the number of tokens used during fine-tuning, and ongoing usage fees for inference with the fine-tuned model. These costs vary by provider but are generally proportional to the size of the model and the volume of fine-tuning data.

Table 1 lists LLMs considered for selection process. The bold entries highlights the reason why the particular model does not comply with the framework requirement.

Table 1: Comparison of Candidate Models for Fine-Tuning.

Model	**Context Window**	**Fine-Tuning Available**	**Fine-Tuning Cost**	**Selected?**
`Phi-2` (Abdin et al. 2024)	**4K**	**Yes (Offline)**	-	✗
`CodeLLaMA-Python` (Touvron et al. 2023)	16K	**Yes (Offline)**	-	✗
`Mistral-7B` (Albert et al. 2023)	32K	**No**	-	✗
`Mixtral-8x7B` (Jiang et al. 2024)	32K	**No**	-	✗
`GPT-3.5-Turbo` (Brown et al. 2020)	16K	Yes (API)	**High**	✗
`GPT-4` (Achiam et al. 2023)	128K	Yes (API)	**High**	✗
`GPT-4.1 mini` (OpenAI 2025)	128K	Yes (API)	Low	✓

Given these constraints, we selected the OpenAI's one of the latest releases `GPT-4.1 mini` model for fine-tuning. `GPT-4.1 mini` supports fine-tuning through OpenAI's API infrastructure and offers a sufficiently large context window of up to 128K tokens (OpenAI 2025) with relatively minimal cost per million tokens, making it well-suited for our task.

4 RESULTS

4.1 Quantitative Analysis

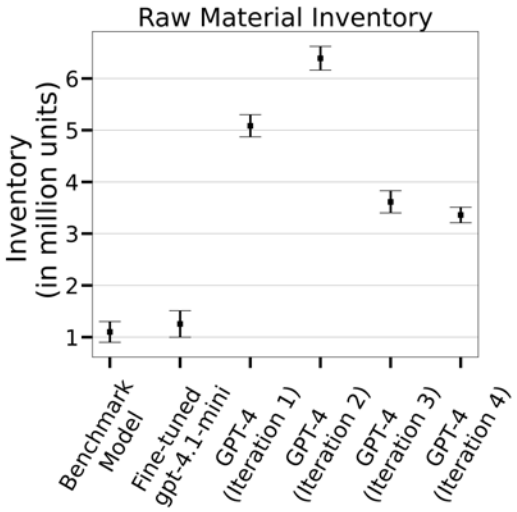
We quantitatively compare simulation models generated by our proposed architecture (fine-tuned `GPT-4.1 mini`) with those produced by the general-purpose `GPT-4`. For fine-tuning the `GPT-4.1 mini`, a dataset of 900 samples were utilized for training and 100 samples for validation. The model was trained over 2 epochs with a learning rate set to 0.1 and no batching. Although `GPT-4` exhibits superior general reasoning capabilities (OpenAI 2025), we hypothesize that fine-tuning imparts domain-specific knowledge to the smaller model, allowing it to outperform `GPT-4` in structured simulation tasks. Both models are evaluated against a manually validated benchmark implemented in SimPy and Arena, ensuring high fidelity. Two simulation scenarios of varying complexity—spanning both simulation logic and linguistic variability—are used to assess the generalization capacity of our architecture. Note that the test scenarios are synthetically constructed for quantitative analysis and may not reflect fully realistic or logically complete supply chain scenarios.

4.1.1 Test Scenario 1

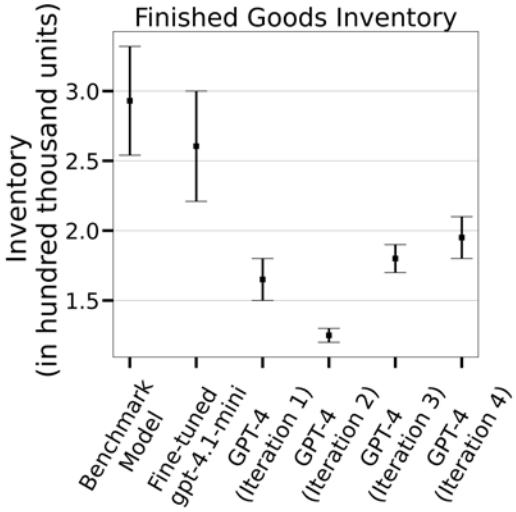
Test Scenario 1 is designed with comprehensive and unambiguous information necessary for model simulation, including explicit specifications for all parameters required for simulating the corresponding supply chain scenario. The prompt provides all required details in a clear and direct manner, resulting in low complexity from both a simulation and natural language interpretation perspective.

Scenario 1 Prompt

Generate a simulation model using SimPy library for following simulation scenario. Following is the simulation scenario. We have two raw materials: **Microprocessors** and **Memory Chips**. Microprocessors are supplied **every day** in batch of **60 thousand units**. Memory Chips are ordered **based on its demand**. We start with **no inventory** for raw materials. Microprocessors are supplied by Fabrication Co. with delivery delay of **6 to 8 days** and costs **200 dollars** for each microchip. Memory chips are supplied by Supply Memory Co. based on customer orders with delivery lead time of **10 to 18 days** . All the supplier payments are paid in **25 to 30 days** of order delivery. The memory chip suppliers charges **100 dollars per unit** of memory chip. We use **2 memory chips and 1 microprocessor** to assemble one unit of Thermo Cell. We start with **initial inventory of 3 millions units** of Thermo Cell and assemble the finished products **daily** and assembly takes **1** day. The customers arrive every month with demand of **1.8 to 2.2 million units** of Thermo Cell and pays **500 dollars** per unit of Thermo Cell. The customer order is completed in two weeks and the payment is completed by the customers in **7 days**. Simulate the scenario for **1** year with no warm-up and replicate the simulation **10 times** to account for any variability and report 99% confidence interval for daily cash balance, daily raw material inventory levels and daily finished product inventory levels.

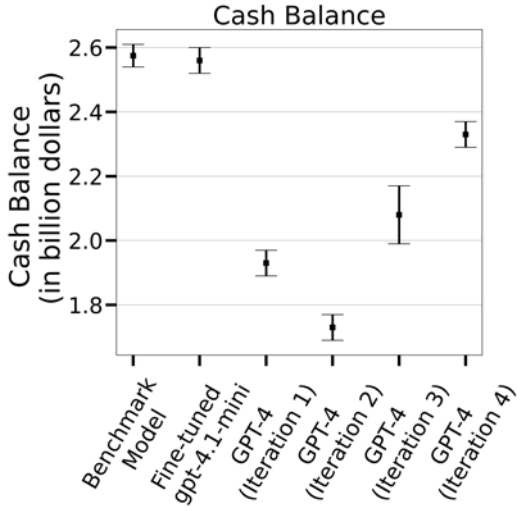


(a) 99% confidence interval for daily raw material inventory levels across 10 macro-replications.



(b) 99% confidence interval for daily finished goods inventory levels across 10 macro-replications.

(continued)



(c) 99% confidence interval for daily cash balance across 10 macro-replications.

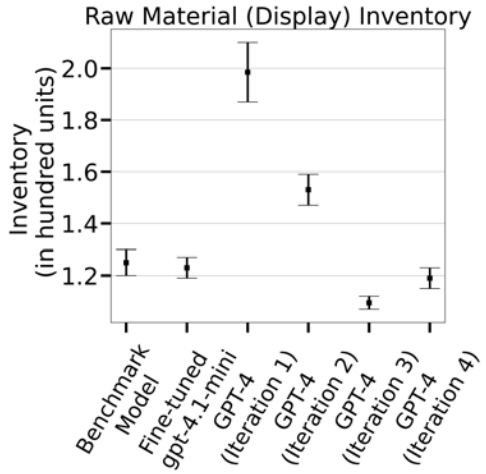
Figure 3: Quantitative analysis for Test Scenario 1.

Figure 3 shows 99% confidence intervals for key inventory metrics, comparing outputs from fine-tuned `GPT-4.1 mini`, iterative `GPT-4`, and a benchmark. Each `GPT-4` iteration reflects corrective feedback, though some adjustments worsened performance due to unintended side effects—illustrating the fragility of unstructured refinement. Only iterations with notable behavioral shifts are shown. Despite multiple attempts, `GPT-4` diverges from the benchmark, whereas `GPT-4.1 mini` aligns in one pass—highlighting the value of domain-specific fine-tuning.

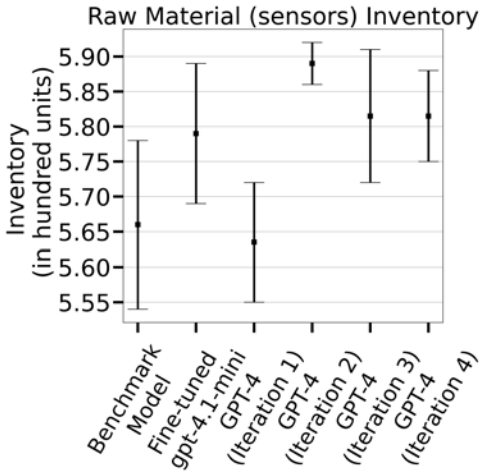
4.1.2 Test Scenario 2

Scenario 2 Prompt

Generate a simulation model using SimPy library for following simulation scenario. Following is the simulation scenario: There are three raw materials, namely **OLED display, sensor, and battery**. The initial inventory of display, sensors, and battery is **100, 500, and 100 units** respectively. The displays are supplied by High Def Manufacturers when the **inventory drops below 50 units and is restocked to 90 units**, and costs **40 dollars** per unit of display screen. The delivery period for displays is **2 to 4 days**, and the payment is cleared after **7 days of order delivery**. Sensors and Batteries are supplied by the Gada Electronics. The sensors are procured every **15 to 17 days** in a batch of **100 units**, and delivery takes **3 to 4 days** with payment terms of **30 dollars** per sensor and payment after **10 days** of order received. Lastly, the batteries are also supplied every **30 days** with a lot of **600 units**. All the delivery and payment terms are similar to sensors. A smart watch is manufactured using **1 display, 4 sensors, and 2 batteries**. The assembly for smart watch batches begins every **3 to 4 days**, and it takes **1 day** to assemble a batch. The customer, Smart Gadgets orders **200 to 300 units** of smart watches every **30 to 35 days** and expects the delivery within **5 to 7 days** of order placement, and pays **500 dollars** per watch within **2 days** of order delivery. Simulate for 2 years with **10-day** of warm-up period and replicate the scenario **10 times** and report 99% confidence interval for daily cash balance, daily raw material inventory levels and daily finished product inventory levels.

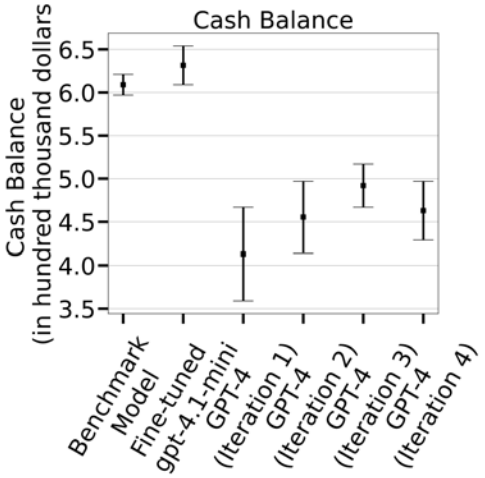


(a) 99% CI for daily raw material inventory levels across 10 macro-replications.



(b) 99% CI for daily raw material inventory levels across 10 macro-replications.

(c) 99% CI for daily finished goods inventory levels across 10 macro-replications.



(d) 99% CI for daily cash balance across 10 macro-replications.

Figure 4: Quantitative analysis for Test Scenario 2.

As shown in Figure 4, the fine-tuned `GPT-4.1 mini` maintains robust performance in complex scenarios. While Figures 4a and 4b show that `GPT-4` occasionally matches the benchmark at specific inventory levels, it lacks the consistency across key metrics needed for accurate real-world modeling, even after multiple refinements.

The quantitative analysis underscores the superior quality of fine-tuned `GPT-4.1 mini`'s simulation output and shows that fine-tuning, coupled with a robust simulation engine, effectively handles both simulation complexity and natural language variability.

4.2 Qualitative Analysis

To evaluate the effectiveness of the fine-tuned model beyond quantitative metrics, a qualitative analysis was conducted by comparing outputs generated from identical prompts. This assessment focused on the structural quality, modeling capabilities, and completeness of the simulation code produced. Table 2 presents

a side-by-side comparison across key simulation components, highlighting the enhanced modeling fidelity, modularity, and scalability achieved through fine-tuning.

Table 2: Qualitative comparison: `GPT-4` vs. fine-tuned `GPT-4.1 mini`.

Category	**`GPT-4`**	**Fine-tuned `GPT-4.1 mini`**
Simulation Modeling Logic	Limited to basic inventory threshold. Improperly handled: lead time, data logging, inventory update and warm-up period	Fully supports advanced procurement logic with accurate handling of lead time, inventory updates, data logging, and warm-up initialization.
Modularity	Low cohesion, high interdependency.	Modular, object-oriented design.
Scalability	Not extensible.	Easily adaptable to large-scale complex systems.
Information Extraction	Misses explicit details.	Extracts both explicit and implicit parameters.
Code Generation	Prone to hallucinations, incomplete outputs.	Generates accurate, executable code.

5 CONCLUSION AND FUTURE DIRECTIONS

This study showcases the effectiveness of fine-tuned LLMs in automating discrete-event simulation (DES) model generation from natural language descriptions in supply chain settings. The fine-tuned `GPT-4.1 mini` outperforms base models in fidelity, structure, and robustness. By structuring the pipeline into description-to-representation and representation-to-code stages, we propose a scalable, extensible architecture that links high-level input to executable simulations. The framework not only reduces manual effort but also lays the groundwork for LLM-driven digital twins, with modularity supporting adaptation via fine-tuning or in-context prompting. In terms of simulation model limitation, the current version of generated simulation models does not support multi-level bill-of-materials, policy optimization and strategy recommendation

Future work includes expanding training data to enhance generalization across broader supply chain scenarios and linguistic variability, extending model capabilities and developing a privacy-preserving local fine-tuning architecture tailored to enterprise data. We also aim to evolve the framework toward fully autonomous, real-time digital twins with continuous learning and predictive analytics.

REFERENCES

Abdin, M., J. Aneja, H. Behl, S. Bubeck, R. Eldan, S. Gunasekar, *et al*. 2024. "Phi-4 Technical Report". *arXiv preprint arXiv:2412.08905*.

Achiam, J., S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, *et al*. 2023. "Gpt-4 Technical Report". *arXiv preprint arXiv:2303.08774*.

Agalianos, K., S. Ponis, E. Aretoulaki, G. Plakas, and O. Efthymiou. 2020. "Discrete Event Simulation and Digital Twins: Review and Challenges for Logistics". *Procedia Manufacturing* 51:1636–1641.

Albert, Q. J., A. Sablayrolles, A. Mensch, C. Bamford, and D. S. Chaplot. 2023. "Mistral 7B". *arXiv preprint arXiv:2310.06825*.

Banks, J. 2000. "Introduction to Simulation". In *2000 Winter Simulation Conference*, 9–16 https://doi.org/https://doi.org/10.1109/WSC.2000.899690.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, *et al*. 2020. "Language Models are Few-Shot Learners". *Advances in Neural Information Processing Systems* 33:1877–1901. December $6^{th} - 12^{th}, Virtual Conference$.

Chan, Y.-C., G. Pu, A. Shanker, P. Suresh, P. Jenks, J. Heyer *et al*. 2024. "Balancing Cost and Effectiveness of Synthetic Data Generation Strategies for LLMs". *arXiv preprint arXiv:2409.19759*.

Chen, M., J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, *et al*. 2021. "Evaluating Large Language Models Trained on Code". *arXiv preprint arXiv:2107.03374*.

Chen, Z., and L. Huang. 2021. "Digital Twins for Information-sharing in Remanufacturing Supply Chain: A Review". *Energy* 220:119712.

D'Ambrosio, G., and S. Luke. "Hybrid Agent-based and Discrete Event Simulation in Mason". In *Proceedings of Annual Modeling and Simulation Conference (ANNSIM) (2023)*. May $23^{th} - 26^{th}, Hamilton, ON, Canada$.

Eckhart, M., and A. Ekelhart. 2019. "Digital Twins for Cyber-Physical Systems Security: State of the Art and Outlook". *Security and Quality in Cyber-Physical Systems Engineering: With Forewords by Robert M. Lee and Tom Gilb*:383–412.

Erdeş, H., and S. E. Kesen. 2025. "Searching for Adoption of Electric Vehicles to our Modern Life: A Discrete Event Simulation Analysis". *Simulation* 101(2):177–194.

Ferreira, A., A. L. Ramos, J. V. Ferreira, and L. P. Ferreira. 2025. "The Role of Hybrid Simulation for Sustainability Supply Chains 4.0 Dynamics". *Procedia Computer Science* 253:67–73.

Flores-García, E., Y. Jeong, M. Wiktorsson, S. Liu, L. Wang, and G. Kim. 2021. "Digital Twin-based Services for Smart Production Logistics". In *2021 Winter Simulation Conference (WSC)*, 1–12 https://doi.org/https://doi.org/10.1109/WSC52266.2021.9715526.

Jackson, I., M. Jesus Saenz, and D. Ivanov. 2024. "From Natural Language to Simulations: Applying AI to Automate Simulation Modelling of Logistics Systems". *International Journal of Production Research* 62(4):1434–1457.

Jiang, A. Q., A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, *et al*. 2024. "Mixtral of Experts". *arXiv preprint arXiv:2401.04088*.

Johansson, B. 2006. "A Methodology for Modular Discrete Event Simulation of Manufacturing Systems". *IFAC Proceedings Volumes* 39(3):449–454.

Longo, E., A. E. Redondi, M. Bianchini, P. Bolzan, and S. Maffei. "Smart Gate: A Modular System for Occupancy and Environmental Monitoring of Spaces". In *Proceedings of International Conference on Smart and Sustainable Technologies 2020 (SpliTech)*. September $23^{rd} - 26^{th}$, Split, Croatia.

Lu, Q., A. K. Parlikad, P. Woodall, G. Don Ranasinghe, X. Xie, Z. Liang, *et al*. 2020. "Developing a Digital Twin at Building and City Levels: Case Study of West Cambridge Campus". *Journal of Management in Engineering* 36(3):05020004.

Macal, C. M., and M. J. North. 2005. "Tutorial on Agent-based Modeling and Simulation". In *2005 Winter Simulation Conference*, 14–27 https://doi.org/https://doi.org/10.1109/WSC.2005.1574234.

Nweje, U., and M. Taiwo. 2025. "Leveraging Artificial Intelligence for Predictive Supply Chain Management, Focus on How AI-driven Tools are Revolutionizing Demand Forecasting and Inventory Optimization". *International Journal of Science and Research Archive* 14(1):230–250.

Ogata, K. 2004. *System dynamics*. New Jersey:Pearson Education. Inc.

OpenAI 2025, April. "Introducing GPT-4.1 in the API". https://openai.com/index/gpt-4-1/. Accessed: 2025-04-18.

Song, Z., B. Yan, Y. Liu, M. Fang, M. Li, R. Yan *et al*. 2025. "Injecting Domain-Specific Knowledge into Large Language Models: A Comprehensive Survey". *arXiv preprint arXiv:2502.10708*.

Tako, A. A., and S. Robinson. 2012. "The Application of Discrete Event Simulation and System Dynamics in the Logistics and Supply Chain Context". *Decision Support Systems* 52(4):802–815.

Terzi, S., and S. Cavalieri. 2004. "Simulation in the Supply Chain Context: A Survey". *Computers in Industry* 53(1):3–16.

Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, *et al*. 2023. "Llama: Open and Efficient Foundation Language Models". *arXiv preprint arXiv:2302.13971*.

van der Valk, H., G. Strobel, S. Winkelmann, J. Hunker, and M. Tomczyk. 2022. "Supply Chains in the Era of Digital Twins–A Review". *Procedia Computer Science* 204:156–163.

Vidalakis, C., J. E. Tookey, and J. Sommerville. 2013. "Demand Uncertainty in Construction Supply Chains: A Discrete Event Simulation Study". *Journal of the Operational Research Society* 64(8):1194–1204.

AUTHOR BIOGRAPHIES

SURDEEP CHOTALIYA is a Ph.D. student in Industrial Engineering at Arizona State University. He received his Master's in industrial and Systems Engineering in 2020 from San Jose State University. His main research interests includes Bayesian Optimization, Simulation-based Optimization, Statistical Machine Learning and Generative AI. His email address is schotali@asu.edu.

JOHN FOWLER is the Motorola Professor of International Management in the Supply Chain Management department of the W.P. Carey School of Business at Arizona State University. His research interests include discrete event simulation, deterministic scheduling, multi-criteria decision making, and applied operations research with applications in semiconductor manufacturing and healthcare. He has published over 135 journal articles and over 100 conference papers. He was the Program Chair for the 2008 Winter Simulation Conference (WSC). He is also the founding Editor-in-Chief of Applied Operations and Analytics and was an Editor of the Journal of Simulation and Associate Editor of IEEE Transactions on Semiconductor Manufacturing. He is a Fellow of the Institute of Industrial and Systems Engineers (IISE) and INFORMS. He served as the IIE Vice President for Continuing Education, is a former INFORMS Vice President, and served on the WSC Board of Directors. His email address is john.fowler@asu.edu.

GIULIA PEDRIELLI is an Associate Professor in the School of Computing and Augmented Intelligence at Arizona State University. She is interested in the area of stochastics and simulation based optimization, and deals with applications in biomanufacturing, power systems, supply chains, safety critical systems. Her email address is giulia.pedrielli@asu.edu.

DAVID BAYBA is an Intel Principal Engineer that manages the Decision Science and AI group in the E2E Supply Chain Data and Solutions team. He has developed many mathematical, simulation, and machine learning models and also enjoys teaching and mentoring. He holds a Master of Engineering in Modeling and Simulation from Arizona State University and a Bachelor of Science in Industrial Engineering from the University of Arizona. His email address is david.m.bayba@intel.com.

MIKAYLA NORTON is a Data Scientist in Intel's Global Supply Chain Organization. Her experience includes generative AI, computer vision, statistical models, machine learning, and simulation models. She applies these to Intel's supply chain needs, including inventory planning, forecasting, sourcing intelligence, and construction planning. Mikayla holds a bachelor's in engineering and a master's in data science from Michigan State University. Her email address is mikayla.norton@intel.com.

PRIYANKA SAIN is a Data Engineer at Intel Corporation and holds a Bachelors Degree in Computer Engineering. Her expertise lies in integrating advanced analytics into supply chain operations, and enabling simulation and optimization. She has hands-on experience with cloud technologies and actively teaches Power BI within her organization. She is passionate about bridging the gap between data engineering and supply chain planning. Her email address is priyanka.sain@intel.com.

KAIXU YU is a Ph.D. student at Arizona State University. He received his masters in Business Analysis from George Washington University. His research interests are in auto-generation digital twin, story-telling data analysis, and data visualizations. His email address is kaixuyu@asu.edu.