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 TUTORIAL ON GENERATIVE AI AND SIMULATION MODELING INTEGRATION

Mohammad Dehghanimohammadabadi[1], Sahil Belsare[1], and Negar Sadeghi[1]

[1]Department of Mechanical and Industrial Engineering, Northeastern University, Boston, MA, USA

ABSTRACT

This tutorial paper explores the applicability of Generative AI (GenAI), particularly Large Language Models, within the context of simulation modeling. The discussion is organized around three key perspectives: *Generation*, where GenAI is used to create simulation models and input data; *Execution*, where it supports real-time decision-making and adaptive logic during simulation runs; and *Analysis*, where GenAI assists in running experiments, interpreting results, and generating insightful reports. In addition to these core perspectives, the paper also covers practical implementation considerations such as prompt engineering, fine-tuning, and Retrieval-Augmented Generation. The tutorial offers both conceptual guidance and hands-on examples to support researchers and practitioners seeking to integrate GenAI into simulation environments.

1 INTRODUCTION

The rise of Generative AI and large language models (LLMs) has revolutionized the landscape of artificial intelligence (AI). LLMs like OpenAI's ChatGPT and Google's Gemini have pushed the boundaries of what AI can achieve, transitioning from basic text generation to more complex tasks involving creativity and multi-modal capabilities. These advancements have transformed traditional AI applications and opened up new avenues for innovation across industries.

The initial iterations of ChatGPT were primarily designed for text generation. However, as generative AI models have advanced, their capabilities have expanded to include image, audio, and video generation. The rapid growth of GenAI has led to a proliferation of platforms leveraging its capabilities. Users can now create websites using platforms like (Webflow)Webflow, design flowcharts with tools like (DrawIO)draw.io, and develop dashboards for business intelligence using (GenBI)GenBI.co. Additionally, platforms like Agent Laboratory facilitate the generation and review of research papers by simulating PhD-level agents and reviewers (Schmidgall et al. 2025).

Despite these advancements, the field of simulation has yet to fully harness the potential of generative AI. While there have been sporadic research efforts utilizing LLMs for simulation, most focus on agent-based models or system dynamics. For instance, recent studies have explored integrating large language models into agent-based modeling and simulation, discussing challenges and future directions (Ghaffarzadegan et al. 2023). Additionally, Hu (2025) introduced ChatPySD, a novel approach that embeds system dynamics models within ChatGPT-4 to enable users to interactively simulate and analyze models through natural language conversations (Hu 2025).

However this highlights a clear gap in applying generative AI to discrete event simulation (DES) and suggests a promising area for future exploration and development. This paper aims to bridge this gap by providing a comprehensive tutorial on the integration of GenAI into simulation modeling. The key contributions of this paper are:

- Provide a high-level tutorial on large language models and their underlying mechanisms.
- Discuss the potential integration of GenAI across different stages of simulation modeling (model generation, execution, and experimentation).
- Present an example of the use case to offer insights.

Please note that the terms *"Generative AI"* and *"GenAI"* are used interchangeably throughout this paper, both referring to the same concept. While some existing works have applied and integrated GenAI with simulation modeling, this paper intentionally focuses on the tutorial and practical aspects of such integration rather than serving as a comprehensive literature review.

2 WHAT ARE LARGE LANGUAGE MODELS?

GenAI has been one of the most transformative developments in the recent technological history.The release of ChatGPT by OpenAI in late 2022 brought powerful AI capabilities, such as natural conversation, reasoning, and creative generation—into the hands of the general public. At the core of this revolution are large language models, the technology that enables machines to generate human-like text, understand context, and perform reasoning tasks. To understand where LLMs fit within the broader AI landscape, it's helpful to understand where it fits within the hierarchy of related technologies. AI refers to the general pursuit of creating machines that can simulate human intelligence. Machine Learning (ML), a subset of AI, focuses on algorithms that improve with experience. Within ML, Deep Learning (DL) leverages neural networks with multiple layers to recognize complex patterns in data. Parallel to this is Natural Language Processing (NLP), which focuses on enabling machines to understand and use human language. As illustrated in Figure 1, LLMs sit at the intersection of these areas. It uses deep learning techniques to perform NLP tasks at unprecedented scale and accuracy (Ewa Halejak).

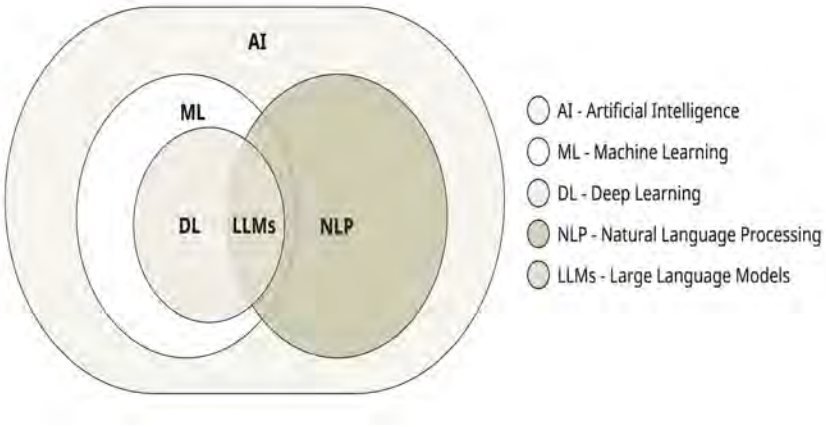


Figure 1: Relationship between LLMs and related AI concepts.

2.1 How LLMs Work

LLMs work by predicting the next word or token in a sequence, a process made possible by training on massive corpora of text data. The architecture that powers modern LLMs is the **Transformer**, introduced by a group of Google scientists in the 2017 paper *"Attention Is All You Need"* (Vaswani et al. 2017). Unlike earlier models that processed text sequentially, Transformers use a mechanism called self-attention, allowing the model to weigh and relate words across an entire sequence, regardless of their position. This ability to dynamically focus on relevant parts of the input makes Transformers especially powerful for capturing complex dependencies and generating coherent outputs.

Consider an example where an LLM is tasked with completing the sentence: *"Artificial Intelligence is transforming the ..."* To do this, the model must understand both the context and the structure of the input. As illustrated in Figure 2, the prediction process unfolds in three main stages:

(i) Embedding: The input sentence is first tokenized into individual units such as *"Art"*, *"ificial"*, *"Intelligence"*, *"is"*, *"transforming"*, and *"the"*. Each token is mapped to a unique token ID (e.g., 8001 for "Art", 9345 for "Intelligence") and then converted into a dense vector representation. A positional encoding,

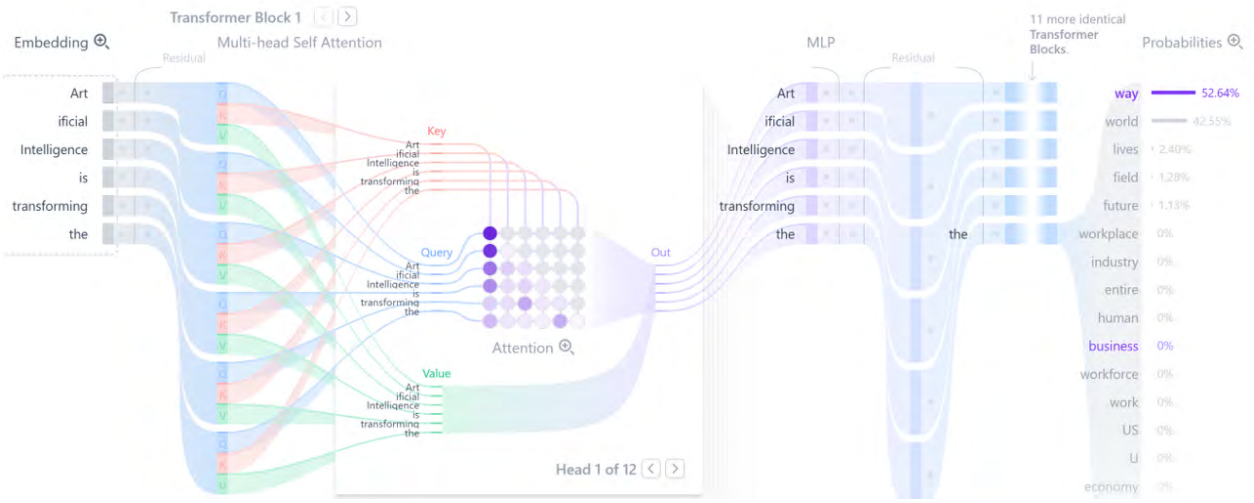


Figure 2: Visualization of the Transformer architecture in action. This figure is adapted from the interactive explainer by (Cho et al. 2025) and the original model explanation website at Repository Website.

which encodes the token's position in the sequence (e.g., 0 through 5), is added to each embedding to preserve word order. This combined embedding is what enters the Transformer (Figure 3a).

(ii) Transformer Block: Each Transformer block contains two main components: a Multi-Head Self-Attention mechanism and a feedforward neural network (MLP), both wrapped in residual connections and followed by normalization. In the self-attention mechanism, each token generates a Query (Q), Key (K), and Value (V) vector. Intuitively, the *Query* represents what a token is trying to understand, the *Key* represents the information each token offers, and the *Value* holds the content that may be passed on. For example, in the sentence *"Artificial Intelligence is transforming the ..."*, the token "transforming" might query the earlier words to understand what is being transformed. It may find a strong match with the Key vector from "Intelligence" and use the Value from "Intelligence" to better contextualize its meaning.

These vectors are used to compute attention scores (1) through dot products between the Query and Key vectors. These scores are then scaled, optionally masked, and passed through a softmax layer to determine how much attention each token should pay to the others. The resulting weights are applied to the Value vectors, producing contextualized representations of each token. These representations are then passed to subsequent layers for further refinement and understanding. Here, d_{model} is the dimensionality of the embedding space (e.g., 768 in BERT-base or 1024 in GPT-2).

$$QKV_{ij} = \left(\sum_{d=1}^{d_{\text{model}}} \text{Embedding}_{i,d} \cdot \text{Weights}_{d,j} \right) + \text{Bias}_{j.} \tag{1}$$

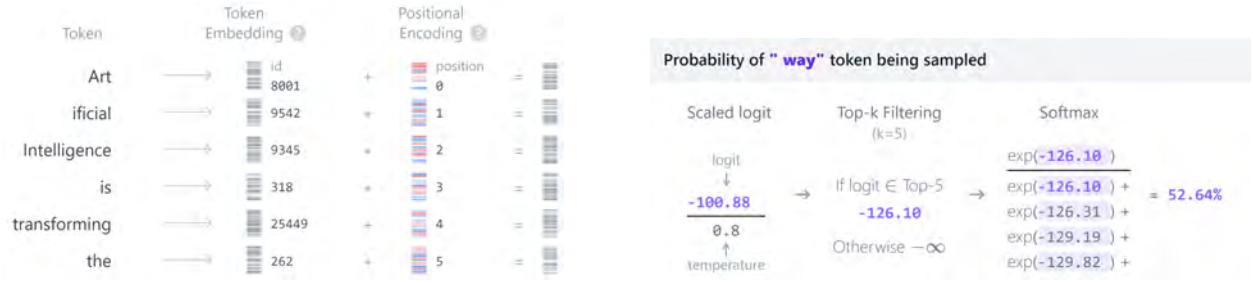
(iii) Output Probabilities: After processing through several Transformer blocks, the model generates a probability distribution over the next token. In Figure 3b, we see that the word *"way"* is predicted with the highest probability. The model uses strategies like softmax and Top-k filtering to finalize this decision.

For a deeper understanding, we encourage readers to explore the interactive visualization at the following link (Cho et al. 2025) Transformer Explainer.

.

2.2 Controlling LLM Output with Temperature and Top-k/Top-p Sampling

One of the defining features of LLMs is their probabilistic nature, which enables them to generate creative, varied, and context-sensitive responses. This behavior stems from the fuzziness inherent in language



(a) Token embedding and positional encoding.

(b) Softmax output probabilities.

Figure 3: Transformer components: (a) Embedding layer, (b) Output layer.

modeling, where predictions are made not deterministically, but based on learned probability distributions over possible next tokens.

A key parameter in LLMs that controls the fuzziness and probabilistic nature of their output is **temperature**. This setting adjusts the randomness in predictions during sampling. Lower temperatures (e.g., 0) make the model highly focused and deterministic, favoring only the most likely next token. In contrast, higher temperatures (e.g., 10) flatten the probability distribution, enabling more varied and creative outputs. This effect is visualized in Figure 4, where the model's prediction shifts from a dominant token (*"way"*) at low temperature to a more even spread across multiple options at high temperature.

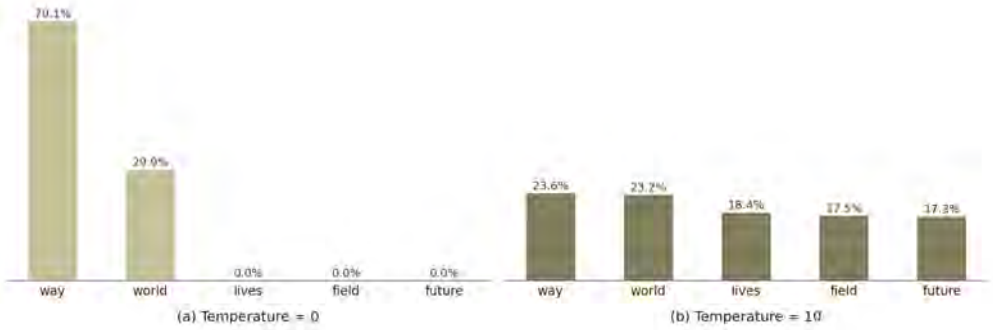


Figure 4: Comparison of token probabilities at two temperature settings.

In addition to temperature, two other parameters help control output diversity:

- **Top-k Sampling:** Restricts the next-token candidates to the top k highest-probability tokens. For example, Top-5 limits selection to only the five most likely tokens.
- **Top-p (Nucleus) Sampling:** Dynamically includes the smallest set of top tokens whose cumulative probability exceeds a threshold p (e.g., $p = 0.9$). This is more adaptive than Top-k and balances coherence with diversity.

Together, these parameters give users powerful control over how confident, precise, or imaginative a model's responses should be—making them essential tools for tailoring LLM behavior to different tasks or audiences.

3 EXAMPLE USE CASE: EMERGENCY ROOM MODEL

This Emergency Room (ER) simulation model represents the patient flow in a hospital's emergency department. Patients arrive randomly and are assigned a severity score upon triage. Depending on their

severity, they are directed either to CriticalCareBeds (for severe cases) or StandardBeds (for moderate cases). Treatment times are modeled using triangular distributions based on the bed type: for CriticalCareBeds, the treatment time follows Triangular$(20, 30, 45)$ minutes, and for StandardBeds, it follows Triangular$(10, 15, 25)$ minutes. Patients wait in queues if beds are not immediately available, and selection for treatment is based on either a First-In-First-Out (FIFO) rule or a Shortest Expected Treatment Time rule. The system collects key performance metrics such as waiting time, service time, and resource utilization. Figure 12 illustrates the patient journey through the ER system. Throughout this paper, the implications of Generative AI within this ER model are explained in the *Case-in-Point* boxes to provide insights and practical context.

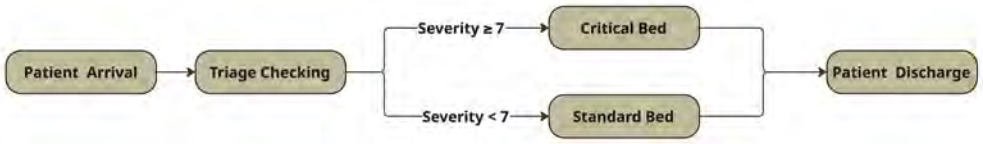


Figure 5: Exemplary ER flowchart.

4 GENAI-DRIVEN SIMULATION: A NEW FRONTIER

Building a high-fidelity simulation model is a time-intensive task that often requires domain-specific knowledge, strong modeling and coding skills, and a deep understanding of optimization techniques (Belsare et al. 2022). Generative AI has the potential to automate and accelerate this process by learning from historical data, technical documentation, or existing models to generate complex, behavior-rich simulation environments. While simulation has long relied on structured logic and rules, recent advances in Generative AI are introducing new levels of flexibility and creativity. Figure 6 presents a comprehensive framework illustrating how Generative AI can transform the traditional simulation modeling lifecycle through three sequential phases. The Creation (Phase 1) leverages GenAI for foundational activities, including concept ideation to generate modeling approaches, input data modeling to address data scarcity through intelligent parameter suggestions, and automated model generation via "vibe coding" techniques. The Execution (Phase 2) integrates GenAI during runtime operations, enabling real-time simulation narration for improved transparency, dynamic process orchestration for adaptive decision-making, and outcome forecasting to predict system bottlenecks. Finally, the Experimentation (Phase 3) employs GenAI for post-simulation analysis, including automated results interpretation, scenario generation for comprehensive what-if analysis, and optimization recommendations. This linear progression demonstrates how GenAI can enhance simulation modeling from initial conception through final optimization.

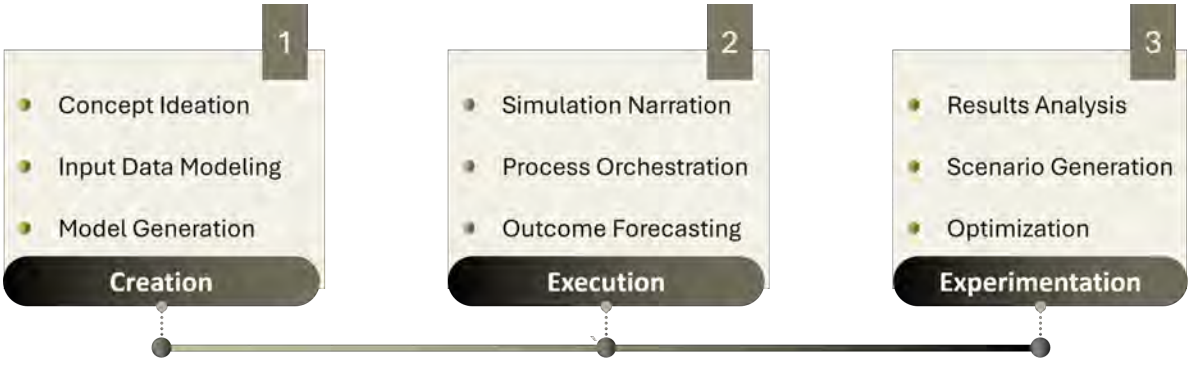


Figure 6: How Generative AI can enhance the simulation modeling process: from *Creation* and *Execution* to *Experimentation*.

5 CREATION

In the Creation phase, Generative AI supports early-stage modeling by suggesting potential system structures, filling data gaps with smart parameter estimates, and generating initial code frameworks using intuitive, example-based methods.

5.1 Concept Ideation

Based on a study conducted by Harvard, *ideation* is identified as one of the top use cases for GenAI (Zao-Sanders 2025). This underscores the importance of GenAI in generating ideas and novel approaches to challenges, fostering creativity and enabling innovative problem-solving. In the context of simulation modeling, regardless of the specific approach, GenAI acts as a *co-pilot* for decision-makers and practitioners by supporting the development, refinement, and enhancement of ideas for generating simulation models. This assistance includes, but is not limited to:

- **Model Complexity Selection:** Deciding whether the model should be simple, detailed, or a hybrid, depending on the problem requirements and data availability (e.g., a basic ER model vs. a full hospital system simulation).
- **Process Identification:** Identifying the key processes and events to include in the model, such as patient triage, resource allocation, queue management, or treatment stages.
- **Decision Variables and Optimization Goals:** Clarifying the variables that influence outcomes and defining optimization objectives (e.g., number of available beds, scheduling, minimizing wait times).
- **Input Data Modeling:** Suggesting relevant data types and statistical distributions for modeling uncertainty (e.g., exponential distribution for inter-arrival times, triangular distribution for service durations).
- **Platform and Method Recommendation:** Recommending appropriate simulation platforms or modeling paradigms based on complexity and user needs (e.g., SimPy for custom code-based models, AnyLogic for hybrid simulation needs).
- **Experimentation Strategy Design:** Assisting in defining how experiments should be set up, such as choosing replication sizes, defining experimental factors, or setting performance metrics (e.g., comparing FIFO vs. priority-based patient selection strategies).
- **Constraint and Assumption Framing:** Helping users frame reasonable assumptions and constraints for the model (e.g., assuming constant staff availability, ignoring patient transfers between hospitals).

This phase often involves conversational prompting with tools like ChatGPT or Gemini to iteratively explore and refine modeling options. Please note, although using GenAI for idea generation is gaining popularity, it is important to recognize that the innovation level of current GenAI models is not extraordinary. As a reminder, LLMs are trained on existing human knowledge, and while they can recombine information in novel ways, their outputs are fundamentally grounded in pre-existing data. Consequently, while GenAI can enhance the quantity and diversity of ideas, the originality and disruptive innovation of those ideas may remain limited. Several recent studies have pointed out these constraints, including observations that GenAI-augmented ideation tends to produce familiar or incremental variations rather than truly groundbreaking concepts (Sun et al. 2024).

5.2 Input Data Modeling

Input data modeling is one of the most critical, yet often most challenging steps in simulation model development, especially when real-world data is unavailable, incomplete, or difficult to obtain. Traditional approaches for addressing data scarcity typically rely on expert opinion, rough estimations ("guestimation"), or using historical analogies to define input parameters and distributions.

Generative AI offers a powerful alternative in these situations. Given that LLMs are trained on vast and diverse datasets, they can serve as intelligent fillers, proposing plausible input assumptions when empirical data is lacking. GenAI can support practitioners by suggesting distribution types, parameter values, or reasonable approximations based on contextual prompts, making it a valuable complement or substitute for traditional input modeling methods in data-scarce environments.

Case-in-Point 1: Generating Patient Attributes with LLMs

❑ An LLM was used via API for two main purposes in the ER simulation (Figure 7):

- **Patient Profile Generation:** As soon as an entity (patient) is created, an API call is made to ChatGPT to generate their profile. Only the patient ID is predefined. All other characteristics—first name, last name, gender, age, and vital signs like blood pressure, temperature, and symptoms—are generated by the LLM.
- **Arrival Pattern Prediction (Optional):** While the simulation currently uses an exponential distribution (mean 10 minutes) for inter-arrival times, LLMs could estimate these based on contextual factors such as day of the week, time of day, and ER crowding—introducing realistic, context-aware arrival dynamics.

❑ This approach enhances realism and individuality of simulated patients. Detailed profiles support more context-aware decisions, such as triage prioritization, care assignment, or estimating patient length of stay.

Figure 7: Key categories of patient data generated by the LLM.

5.3 Model Generation

Developing discrete-event simulation models has traditionally been a complex and time-consuming task, often requiring specialized knowledge and substantial coding effort. Even when using commercial off-the-shelf (COTS) simulation tools, practitioners may face steep learning curves or the need to adapt to proprietary interfaces and scripting languages. These challenges can discourage domain experts without strong programming backgrounds from engaging in simulation modeling.

The emergence of Generative AI (GenAI) has opened new pathways to streamline this process. Popular GenAI models, such as OpenAI's ChatGPT and Anthropic's Claude, are capable of producing high-quality code from structured natural language prompts. These models support a wide range of programming languages, including Python, C#, and HTML, enabling both technical and non-technical users to generate functional applications with minimal barriers.

These capabilities have given rise to a new programming style known as *vibe coding*, where users rely entirely on natural language prompts to generate code without writing it manually. The term *vibe coding* was popularized by Karpathy in early 2025 (Karpathy), describing a shift toward prompt-driven

programming logic. In simulation modeling, this opens the door for intuitive, conversational generation of fully functional simulation environments. Accordingly, the simulation community can benefit from GenAI through two primary approaches: (i) **Vibe Coding**, which refers to prompt-based, LLM-guided model generation; and (ii) **Agentic AI**, which involves automatic model generation using coordinated AI agents. These two approaches are discussed in detail below.

5.3.1 Vibe Coding: Prompt-based (LLM-guided) model generation

One of the most effective frameworks for discrete-event simulation in Python is SimPy, a process-based open-source library. SimPy allows users to model active components such as customers, vehicles, or patients using Python generator functions. It also includes shared resource constructs like servers and counters to simulate capacity-constrained operations. Its clarity and modularity make it an ideal match for integration with LLMs such as ChatGPT.

In this approach, users define the model structure and its parameters using natural language. For example, a user might prompt: *"Simulate a 2-server ER triage system with patient arrivals every 10 minutes."* The LLM then generates a corresponding SimPy model with parameterized logic, modular design, and inline documentation—greatly reducing the modeling burden.

Case-in-Point 2: Generating an ER Model via Vibe Coding

❑ The simulation ER model developed in this study is entirely generated using GenAI via vibe coding. The user simply provided descriptive prompts, and the LLM produced a complete, executable SimPy model. All source code is available at this GitHub link. Figure 8 illustrates a structured prompt consisting of five key components.

Role Assignment → **Modeling Goal** → **Scenario Description** → **Modeling Requirements** → **Expected Output**

This structured prompt enabled the model to automatically generate a full SimPy implementation. ❑ For experienced modelers, writing this simulation by hand may take 30–40 minutes. Using GenAI, the same result was achieved through just a few prompt iterations.

Although vibe coding can significantly reduce the effort required to build simulation models, it still requires careful refinement to ensure high-quality outputs. This refinement process is known as *prompt engineering*, which involves crafting, testing, and optimizing instructions given to the LLM.

Figure 9 illustrates the *Prompt Engineering Cycle*. This process is essential in simulation modeling, where vague or poorly structured prompts can lead to flawed logic or incomplete implementations. Systematic prompt engineering ensures that GenAI produces robust and realistic simulation models. For readers interested in further understanding the role of prompt design, we refer to the framework proposed by Sahoo et al. (Sahoo et al. 2024).

5.3.2 Agentic AI: Automatic Model Generation

As GenAI capabilities grow, future systems will likely support automatic simulation generation via collaborative AI agents. For example, agent-based frameworks such as LangChain or CrewAI can orchestrate multiple specialized agents to construct simulation models based on high-level user input.

In this paradigm, users may specify their model intent through natural language, structured forms (e.g., "2 servers, 3 queues, 1 resource pool"), or even upload images of process layouts. A team of agents then translates this information into structured components; one agent may parse specifications, another generates SimPy code, while another validates the logic.

ROLE: You are a simulation modeling expert and Python programmer specialized in discrete-event systems.

GOAL: Generate a clean, modular SimPy-based simulation model of an Emergency Room (ER) for experimentation, instructional use, and performance analysis.

SCENARIO DESCRIPTION: Patients arrive to the ER following an exponential inter-arrival distribution (mean = 6 minutes).

Each patient is immediately triaged and assigned a severity score (1–10). Based on the score:
- Severity $\geq 7 \rightarrow$ CriticalCareBeds
- Severity $< 7 \rightarrow$ StandardBeds

Treatment durations follow triangular distributions:
- Critical: triangular(20, 30, 45 minutes)
- Standard: triangular(10, 15, 25 minutes)

All major events (arrival, triage, queue entry, bed assignment, treatment start, treatment end) must be logged with time, patient ID, and description. Logs should be human-readable and traceable.

REQUIREMENTS:
- Define two SimPy resources: CriticalCareBeds, StandardBeds with configurable capacities.
- Patients must be uniquely identified (ID) and tracked throughout the simulation.
- Implement a function select_next_patient(waiting_list, strategy) that can use:
• "FIFO"
• "ShortestExpectedTreatment" (use severity-based proxy)
- Modularize logic with reusable components:
• patient_process()
• treat_patient()
• run_single_simulation(settings)
• run_experiment(settings)
- Track performance metrics:
• Per patient: time in system, wait time, treatment time
• Per resource: utilization rate, queue length, patient served count
- Allow user to configure via a settings dictionary:
• Arrival rate, bed counts, selection strategy, number of replications, duration

EXPECTED OUTPUT:
- A single Python script using SimPy that runs without modification.
- Executes a full simulation run (default: 1 replication, 500 simulated minutes).
- Produces structured log output and summary statistics.
- Designed for readability, extensibility, and reuse in simulation education.

Figure 8: Structured prompt for generating a SimPy-based ER simulation using Generative AI.

Figure 13 illustrates a conceptual architecture where a pipeline of specialized AI agents—Problem Interpreter, System Designer, Code Generator, Validation Agent, and Experiment Agent—collaborate sequentially to automate discrete event simulation development.

Figure 9: Prompt engineering steps for SimPy-based discrete-event simulation: (i) role and goal definition (e.g., "You are a simulation expert modeling an ER"), (ii) system setup and entities (e.g., patients, doctors, beds), (iii) event logic and flow (e.g., arrival → triage → treatment), (iv) stochastic elements (e.g., exponential inter-arrival, triangular treatment time), and (v) expected outputs (e.g., logs, wait times, utilization). This process should be iterative to ensure accurate and high-quality code generation.

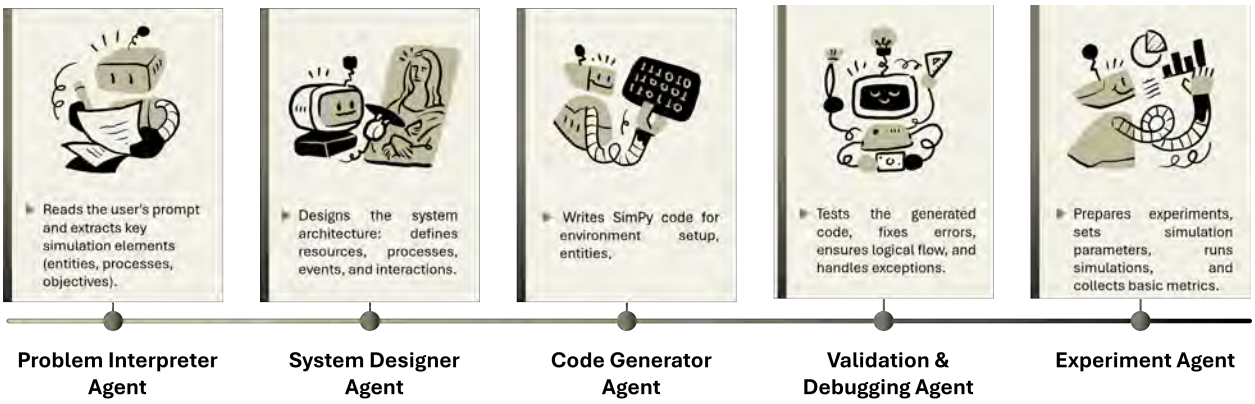


Figure 10: Agentic AI pipeline for discrete event simulation code synthesis.

While this approach requires careful system design, the payoff is significant: simulation creation becomes accessible to non-experts. Moreover, as simulation software platforms evolve, we expect off-the-shelf tools to embed GenAI features directly reducing the burden on practitioners and speeding up experimentation. However, this also necessitates responsible oversight: automatically generated models must still be validated by simulation professionals to ensure correctness and avoid black-box assumptions.

The human-based scripting process involves defining the code architecture, identifying the processes to be modeled, specifying required parameters, sequencing events, and writing explanatory comments. These tasks are typically completed before the first output is generated or any debugging begins. For a proficient coder, this initial code development phase generally takes 30–40 minutes.

This approach promises greater accessibility and automation. However, it also introduces the need for responsible oversight, as auto-generated models must still be verified by simulation experts to ensure validity and avoid black-box risks.

5.4 Retrieval-Augmented Generation (RAG)

Retrieval-augmented generation (RAG) enhances large language models by combining them with external knowledge sources, enabling responses grounded in recent or domain-specific information beyond pre-trained knowledge. Claude's automatic RAG capability enables sophisticated decision-making without explicit domain instructions, when provided minimal prompt information about patient queuing, it autonomously retrieves and applies medical triage knowledge to balance urgency, severity, and service times. Instead of requiring modelers to program complex decision trees, Claude automatically draws from healthcare protocols,

emergency medicine practices, and queuing theory to consider clinical urgency, fairness, operational efficiency, and ethical factors. For example, Claude's patient selection responses demonstrate understanding of medical triage protocols by prioritizing higher severity cases while considering equity and system flow.

6 EXECUTION

In this article, execution refers to the integration of GenAI into the simulation *run* itself. Therefore, the focus of this section is on incorporating GenAI-powered support during runtime. It explores how GenAI can enhance simulation execution and highlights key insights.

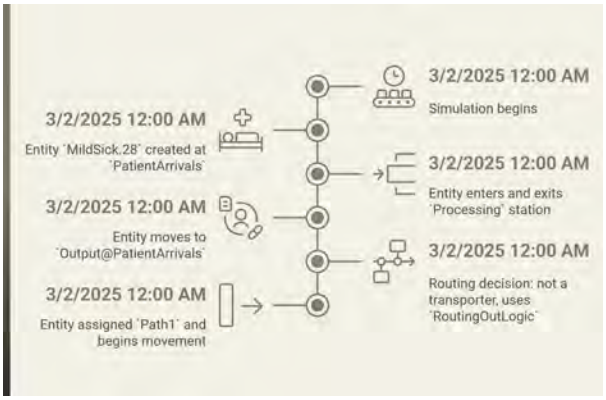
6.1 Simulation Narration

Generative AI can track simulation logs in real time and convert them into natural language explanations. This ability transforms raw, verbose event logs into interpretable narratives, improving transparency and helping users understand what happened in the simulation—and why. This is especially valuable in complex, event-driven systems.

Trace Interpretation and Summary: Interpreting trace data can be overwhelming due to the volume and structure of the data. It is often tedious and not straightforward. GenAI can identify key milestones, extract relevant context, and summarize the simulation flow in an intuitive way. This supports clearer understanding and enables visualization of critical events (e.g., breakpoints) for better debugging and stakeholder communication, as illustrated in Figure 11.

Step	Action
[Transfer] FromPath	Entity 'MildSick.28' transferring from '[EndOfLink] Path1' into node 'Inpu...
[Begin]	Process 'Input@Registration.OnEnteredToAssociatedObject' execution...
[Fire] EnteredEvent	Firing event 'Input@Registration.Entered'.
[Transfer] ToAssociate...	Entity 'MildSick.28' transferring from '[Node] Input@Registration' into o...
	Entity 'MildSick.28' transferring from '[Node] Input@Registration' into s...
[Begin]	Process 'Registration.OnEnteredInputBuffer' execution started.
[EndTransfer] IntoInpu...	Entity 'MildSick.28' ending transfer into station 'Registration.InputBuffer'.
[Begin]	Process 'Input@Registration.OnExited' execution started.
[Fire] ExitedEvent	Firing event 'Input@Registration.Exited'.
[End]	Process 'Input@Registration.OnExited' execution ended.
[Seize] Server	Object 'MildSick.28' checking availability to seize '1' resource(s) of reso...
	Object 'MildSick.28' checking availability to seize '1' capacity unit(s) of r...
	Object 'MildSick.28' found '2' currently available capacity unit(s) of reso...
	Object 'MildSick.28' found '1' currently available resource(s) of resourc...
	Object 'MildSick.28' seized '1' capacity unit(s) of resource 'Registration'.

(a) Trace Log of a simulation run (Simio output).



(b) Narrative summary generated using Napkin.ai.

Figure 11: Simulation trace visualization. (a) shows the raw trace log generated by a COTS simulation tool. (b) illustrates how Generative AI transforms the trace into an intuitive visual narration, enabling easier interpretation and storytelling.

Debugging Support: While some programming platforms now offer GenAI-assisted debugging, most commercial off-the-shelf (COTS) simulation tools do not yet provide this support. In the future, GenAI could assist with interpreting model errors, suggesting fixes, or even guiding users through help documentation and simulation-specific learning materials.

6.2 Process Orchestration

Implementing decision-making logic is one of the most critical areas of research in simulation modeling. Decisions such as how to dispatch jobs, set routing paths for transporters, release entities into the system, or adjust server capacity all fall under the category of *process orchestration*. These decisions share a common goal: enhancing system flow, reducing bottlenecks, and improving performance metrics within

the simulation model. A large body of research has focused on developing techniques, heuristics, and optimization models (Dehghanimohammadabadi et al. 2017) as part of the broader field of *simulation intelligence* (Dehghanimohammadabadi and Keyser 2017) to handle these decision problems effectively. Generative AI, although still in its early stages for simulation applications, offers promising new methods to support decision-making *within the simulation run*. During execution, GenAI can analyze system-level metrics (e.g., number of processed entities, queue lengths, available resources) and entity-level attributes (e.g., severity, priority, size) to dynamically adjust the process flow in real time. Example applications include:

- **Dispatching Rules:** GenAI can choose between FIFO, shortest processing time, or hybrid rules based on the system state.
- **Transport Routing:** AI models can help transporters select optimal delivery paths under congestion.
- **Capacity Adjustment:** GenAI can decide when to add or remove capacity (e.g., activate more servers) to balance workloads.
- **Release Control:** By throttling or accelerating the release of jobs into the system, AI can smoothen flow and reduce queue spikes.

Case-in-Point 3: LLM-Guided Bed Assignment in ER Simulation

❑ To demonstrate GenAI's use during simulation execution, the ER model was extended with an AI-assisted module for real-time bed assignment. A custom `PatientSelectionLLM` class was used to interface with an LLM (Claude), handling prompt formatting, API calls, and response parsing.

❑ When a bed becomes available, the system constructs a prompt including all waiting patients' details (ID, age, temp, severity), system metrics (queue, wait times), and strategy descriptions (FIFO, Hybrid, etc.). The LLM selects a patient based on clinical urgency, fairness, efficiency, and ethical factors, returning both the patient ID and justification.

❑ The system includes fallback mechanisms, error handling, and a mock mode for testing. This adaptive reasoning enables explainable, context-aware decisions beyond static rules.

LLM Response Samples:

- *"Patient #1 should be treated next due to a higher severity score (9) despite similar temperature."*
- *"Patient #4 is more critical than Patient #3 and should be prioritized for treatment."*

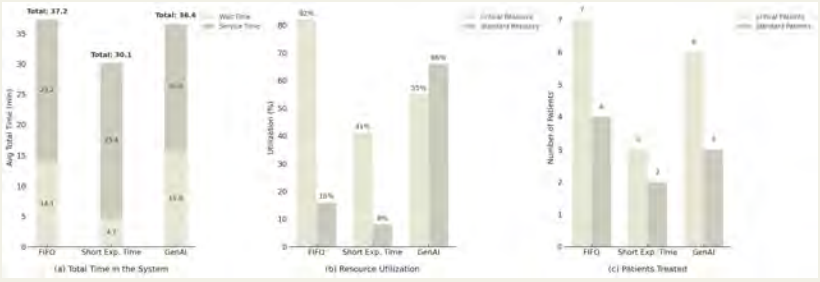


Figure 12: KPI analysis.

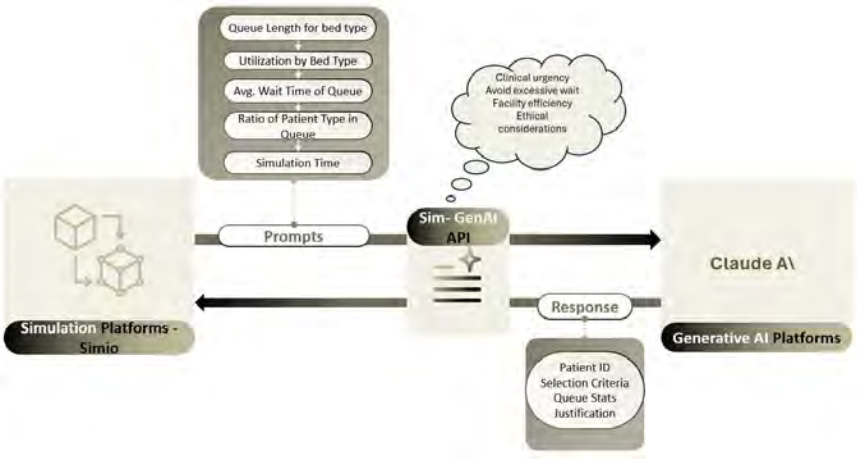


Figure 13: Simulation - GenAI API framework.

6.3 Outcome Forecasting

Leveraging historical and real-time simulation data, GenAI can forecast likely outcomes, such as bottlenecks, service delays, or queue buildup. These predictions support proactive decisions and deeper insights into system performance.

7 EXPERIMENTATION

Interpreting simulation results and gaining system insights is often the most challenging aspect of simulation exercises. This process requires critical thinking to connect system KPIs with business knowledge, ultimately explaining system functionality. LLMs can identify patterns, root causes, and bottlenecks for optimal resource configurations and operational policies.

Case-in-Point 4: LLM-Guided Simulation Analysis

❑ To showcase GenAI's capabilities for inference and system explanation, the model was designed to maintain a comprehensive event trace log, enabling GenAI to draw inferences about system dependencies and sequential patterns.
❑ The LLM was provided with supporting materials and web resources on ER capacity optimization and experimental design methodologies to guide system analysis approaches.
LLM Analysis Extracts:

- GenAI assisted analysis rightfully identified 3 crucial parameters for KPI analysis (Bed Utilization, Patient Wait times, and Patient arrival rates) Moreover, it provided root cause of the system behiavior, concerning system functions, and potential causes.
- *Wait time patterns varied significantly throughout operations, with dramatic increases during peak periods.High bed utilization combined with extended wait times indicates system overcapacity, with resources consistently operating beyond design limits, causing persistent queues and care delays.*

GenAI creates new analytical possibilities through memory retention, enabling causality tracing throughout the analysis process. By maintaining detailed logs of operational metrics—queue lengths, patient categorization, and resource utilization—these models identify causal relationships obscured in conventional

simulations. The temporal sequencing of events allows GenAI to recognize not just correlations but actual causal mechanisms driving system performance. For example, the system can identify that specific arrival patterns typically overburden certain bed types and predict downstream effects on wait times, projecting when beds will reach capacity. This transforms simulation from a planning tool into a dynamic decision support system that recognizes complex interdependencies and recommends tailored interventions for emerging conditions.

7.1 Scenario Generation

LLMs can significantly enhance simulation scenario analysis by autonomously generating diverse what-if scenarios, proposing new input combinations, and exploring edge cases with minimal setup effort. In this case, GenAI conducted a systematic sensitivity analysis by varying critical and standard bed quantities to optimize patient wait times and resource utilization.

Case-in-Point 5: LLM-Guided Scenario Analysis

❑ To showcase GenAI's capabilities for inference and system explanation, the model was designed to maintain a comprehensive event trace log.
❑ The LLM was provided with supporting materials and web resources on ER capacity optimization and experimental design methodologies to guide system analysis approaches.
LLM Analysis Extracts:

- The model designed an experimental approach, testing 12 different bed configurations.
- By using the support reference material, it intelligently defined the scope of the experiment - ***optimal configuration with minimal wait times with reasonable resource utilization.***
- The suggested configuration is a testament that GenAI is able to understand complex interrelation between optimizing for system behavior, patient experience and operational efficiency.
- *Analysis demonstrated that the optimal configuration (1 Critical Bed and 3 Standard Bed) achieved minimal wait time with reasonable utilization rates, balancing efficiency without overcapacity.*

8 CONCLUSION

This tutorial demonstrated that GenAI represents a transformative opportunity for simulation modeling through our three-phase framework of Creation, Execution, and Experimentation. GenAI democratizes simulation by reducing technical barriers, automating complex tasks, and enabling sophisticated decision-making processes. The Creation phase revolutionizes model development through concept ideation, input data modeling, and "vibe coding," allowing domain experts to generate models via natural language prompts rather than extensive programming. During Execution, GenAI introduces dynamic intelligence through real-time narration, adaptive orchestration, and outcome forecasting, exemplified by our RAG-guided bed assignment system. The Experimentation phase transforms result interpretation into automated analysis, identifying causal relationships and generating strategic recommendations. However, important limitations remain. GenAI's innovation potential is constrained by training on existing knowledge, producing incremental rather than groundbreaking insights. Current LLMs excel at standard discrete-event simulations but struggle with complex interdependencies and sophisticated optimization logic that comes with scaled or large models. Automatically generated models require careful validation to ensure accuracy and avoid black-box assumptions. We recommend a hybrid approach combining GenAI automation with expert oversight, implementing rigorous validation protocols, and establishing clear boundaries between automated generation and human validation. This strategy maximizes accessibility benefits while maintaining precision

for complex environments. The integration of GenAI with simulation modeling represents a paradigm shift that fundamentally changes how we approach complex system modeling, making sophisticated capabilities accessible to broader audiences while preserving the rigor simulation practitioners demand. As platforms embed GenAI features, simulation will evolve from a specialized discipline into a broadly accessible decision-support tool.

REFERENCES

Belsare, S., E. D. Badilla, and M. Dehghanimohammadabadi. 2022. "Reinforcement Learning with Discrete Event Simulation: The Premise, Reality, and Promise". In *2022 Winter Simulation Conference (WSC)*, 2724–2735 https://doi.org/10.1109/WSC57314.2022.10015503.

Cho, A., G. C. Kim, A. Karpekov, A. Helbling, Z. J. Wang, S. Lee, *et al*. 2025. "TRANSFORMER EXPLAINER: interactive learning of text-generative models". In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25: AAAI Press https://doi.org/10.1609/aaai.v39i28.35347.

Dehghanimohammadabadi, M., and T. K. Keyser. 2017. "Intelligent Simulation: Integration of SIMIO and MATLAB to Deploy Decision Support Systems to Simulation Environment". *Simulation Modelling Practice and Theory* 71:45–60.

Dehghanimohammadabadi, M., T. K. Keyser, and S. H. Cheraghi. 2017. "A Novel Iterative Optimization-based Simulation (IOS) Framework: An Effective Tool to Optimize System's Performance". *Computers & Industrial Engineering* 111:1–17.

DrawIO. "Create Flow Charts". https://app.diagrams.net/. Accessed: 2025-04-25.

GenBI. "Create Insights from Data". https://genbi.co/home. Accessed: 2025-04-25.

Ghaffarzadegan, N., A. Majumdar, R. Williams, and N. Hosseinichimeh. 2023. "Generative Agent-Based Modeling: An Introduction and Tutorial". *System Dynamics Review* 39(1):1–15 https://doi.org/10.1002/sdr.1761.

Ewa Halejak. "ML, NLP, LLM, and Deep Learning Explained – Exploring the Business Potential of AI". https://inwedo.com/blog/business-potential-of-ai-solutions/. Accessed: 2025-05-25.

Hu, B. 2025. "ChatPySD: Embedding and Simulating System Dynamics Models in ChatGPT-4". *System Dynamics Review* 41(1):e1797.

Andrej Karpathy. "Vibe coding is the new programming paradigm". https://x.com/karpathy/status/1886192184808149383. Tweet, accessed May 2025.

Sahoo, P., A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha. 2024. "A Systematic Survey of Prompt Engineering In Large Language Models: Techniques And Applications". *arXiv preprint arXiv:2402.07927*.

Schmidgall, S., Y. Su, Z. Wang, X. Sun, J. Wu, X. Yu, *et al*. 2025. "Agent Laboratory: Using LLM Agents as Research Assistants". In *Agent Laboratory: Using LLM Agents as Research Assistants*.

Sun, T., S. Dow, and C. Callison-Burch. 2024. "Rapid AIdeation: Generating Ideas With the Self and in Collaboration With Large Language Models". *arXiv preprint arXiv:2403.12928*.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, *et al*. 2017. "Attention Is All You Need". In *Advances in Neural Information Processing Systems*. December 4th-9th, Long Beach, USA, 5998-6008.

Webflow. "Create Custom Website". https://https://webflow.com. Accessed: 2025-04-25.

Zao-Sanders, M. 2025, April. "How People Are Really Using Gen AI in 2025". *Harvard Business Review*.

AUTHOR BIOGRAPHIES

MOHAMMAD DEHGHANIMOHAMMADABADI is an Associate Teaching Professor in the Department of Mechanical and Industrial Engineering at Northeastern University. His research focuses on developing intelligent simulation frameworks integrated with optimization and AI-based models to support decision-making. His work has applications in healthcare systems, manufacturing operations, and supply chain logistics. This paper is part of his broader effort to democratize the use of AI in simulation modeling, particularly the application of Generative AI across various stages of simulation to enhance automation, adaptability, and insight generation. His email address is m.dehghani@northeastern.edu.

SAHIL BELSARE is a simulation scientist with expertise in digital twin technologies and simulation modeling. He applies machine learning models to enhance decision-making in complex systems, focusing on optimization and reinforcement learning integrated with simulation platforms. His email is belsare.s@northeastern.edu.

NEGAR SADEGHI is a Ph.D. student in Industrial Engineering at Northeastern University. Her research focuses on developing intelligent simulation models for pharmaceutical supply chains and production-distribution systems. Her e-mail address is: sadeghi.ne@northeastern.edu.