Proceedings of the 2026 Winter Simulation Conference
V. Ramamohan, A. Djanatliev, M. Fakhimi, C. Krejci, C. Ruiz Martin, B. S. Onggo, and N. Mustafee, eds.

LLM-ASSISTED DIAGNOSTICS FOR SIMULATION–REALITY DISCREPANCIES IN MANUFACTURING SYSTEMS

Sanket Bhat[1], Alp Akcay[2], Tugce Martagan[2]
[1]Amazon, Bengaluru, INDIA
[2]Dept. of Mechanical and Industrial Eng., Northeastern University, Boston, USA

ABSTRACT

Simulation models are widely used for decision support in manufacturing systems, yet their accuracy degrades over time as real-world operations evolve. Existing approaches to model maintenance rely heavily on manual diagnosis by domain experts, creating a bottleneck in sustaining model fidelity. This paper proposes a methodology for root cause attribution of simulation model drift using system logs and large language models (LLMs). Given discrepancies between simulated and observed performance metrics, the approach first distinguishes systemic deviations from stochastic noise using statistical tests, and then attributes deviations to specific system components through LLM-driven reasoning. The LLM operates over structured representations of the simulation model and log-derived features to generate interpretable hypotheses about underlying causes. The proposed framework is evaluated using controlled drift scenarios in a semiconductor manufacturing simulation testbed. Results demonstrate improved attribution accuracy, highlighting the potential for LLM-driven self-diagnosing simulation models.

1 INTRODUCTION

Discrete-event simulation models play a central role in the analysis and control of complex manufacturing systems, enabling practitioners to evaluate system performance, test operational policies, and support decision-making under uncertainty. These models typically capture key system characteristics such as stochastic processing times, machine failures, resource constraints, and control logic. Despite their widespread use, a persistent challenge in practice is maintaining model fidelity over time (Akcay et al. 2023). As real-world manufacturing systems evolve due to equipment degradation, process improvements, or policy changes, discrepancies emerge between simulated outputs and observed system behavior. This phenomenon, often referred to as model drift, reduces the reliability of simulation-based insights. In industrial settings, diagnosing the root cause of the model drift is typically a manual and iterative process in which domain experts analyze system logs, identify potential causes of discrepancies, and update model parameters or logic accordingly. This process is both time-consuming and difficult to scale, particularly in complex systems with many interacting components.

Prior work has largely focused on improving simulation accuracy through calibration and validation techniques (Corlu et al. 2020). However, these approaches primarily aim to detect discrepancies in aggregate outputs and do not explicitly address the problem of root cause attribution, identifying which components of the system are responsible for observed deviations. In complex stochastic systems, multiple factors can produce similar observable effects, making attribution inherently challenging. For example, an increase in cycle time may arise from changes in processing times, machine reliability, or control policies, and distinguishing among these requires deeper system-level reasoning. A complementary line of work in manufacturing has approached this problem directly through data-driven root cause analysis, employing techniques such as case-based reasoning combined with process mining (van der Pas et al. 2024), machine signal analysis for device-level fault attribution (Braakman et al. 2025), and model-driven case building from distributed data sources (van der Pas et al. 2023). However, the reasoning process still relies heavily on manual modeling effort and domain expertise to define the causal structure of the system under study.

There is growing interest in whether large language models (LLMs) can reduce this burden, given their ability to support structured reasoning without requiring explicit manual specification. In simulation community, recent review articles examine various methodologies for integrating generative AI with simulation in different stages of the modeling and analysis process (Feldkamp 2025; Dehghanimohammadabadi et al. 2025; Guo et al. 2025). In particular, recent studies explore the use of LLMs to reduce the effort required to build and refine simulation models by translating natural language or unstructured data into formal representations (Elbasheer et al. 2025; Botello et al. 2025; Behrendt et al. 2026). In addition, several studies examine LLM-driven parameterization, validation, manufacturing system analysis, and self-evaluation of simulation models, highlighting their potential to improve model quality and reduce dependence on expert intervention (Xia et al. 2024; He et al. 2025; Selak et al. 2025; Möltner et al. 2025). These approaches also demonstrate that LLMs can improve the accessibility of simulation tools for those without deep modeling experience. In addition, a growing body of research focuses on the use of LLMs for diagnostics and explainable reasoning in complex systems (Qi et al. 2025; Dave et al. 2024). These studies show that LLMs can support model-based diagnosis by generating diagnostic structures, integrating knowledge representations such as knowledge graphs, and providing interpretable explanations for out-of-specification signals (Marandi et al. 2025; Sztyber-Betley et al. 2025). Together, these research streams motivate an approach that combines LLMs with simulation-based diagnostic systems to detect systemic deviations.

In this paper, we present a framework for identifying the root causes of simulation model drift in manufacturing systems by combining statistical analysis with LLM-driven reasoning. Given discrepancies between simulated outputs and observed system logs, the proposed approach first determines whether the discrepancy reflects a statistically significant deviation from expected variability. It then leverages LLMs to attribute the deviation to specific system components, such as resources, process steps, or control policies. The LLM operates on a structured representation of the simulation model and aggregated log features to generate interpretable hypotheses about potential causes. The proposed methodology is evaluated through a numerical study based on a semiconductor manufacturing simulation testbed, where controlled perturbations are introduced to emulate realistic system changes. The contributions of this work are: (i) a formalization of root cause attribution as a distinct problem in simulation model maintenance, (ii) a hybrid methodology combining statistical analysis, simulation experimentation, and LLM-based reasoning, and (iii) an empirical evaluation demonstrating the effectiveness of the approach in a complex manufacturing setting.

The remainder of the paper is organized as follows. Section 2 presents the proposed methodology, Section 3 describes the Mini-Fab testbed used for evaluation, Section 4 reports the numerical experiments, and Section 5 concludes with directions for future work.

2 METHODOLOGY

We consider a real-world manufacturing system R and its corresponding simulation model S, parameterized by Θ. Both the real system and the simulation model generate event logs that capture time-stamped events such as job arrivals, process completions, machine failures, and transport moves. We denote logs obtained from the real system as L^R and logs obtained from the simulation model as L^S, and use $L \in \{L^R, L^S\}$ when the distinction is not essential.

2.1 Framework Overview

We propose a framework for root cause attribution of simulation model drift based on LLM reasoning. Let $\Theta = \{\theta_1, \theta_2, \ldots, \theta_K\}$ denote the set of K simulation model components that are candidate root causes of drift, including resource-level parameters (such as processing times, failure rates, and repair durations) and control policies (such as dispatching and batching rules). We formulate the root cause attribution as a ranking problem, where each component $\theta_k \in \Theta$ is assigned a score reflecting its likelihood of being a

root cause. The framework consists of three stages, separating statistical detection from attribution while enabling flexible integration of multiple information sources.

(1) **Drift Detection**: Let $Y = \{Y_1, \ldots, Y_m\}$ denote the set of m system-level performance metrics of interest. For each metric Y_j, we assume n independent observations are available from the real system (at non-overlapping time windows such as production shifts) and we can generate n independent replications from the simulation model. To decide whether a discrepancy reflects random noise or a systemic shift, we then apply the Welch two-sample t-test (Law 2015): $D_j = \left(\bar{Y}_j^R - \bar{Y}_j^S\right) / \sqrt{(s_j^R)^2/n + (s_j^S)^2/n}$, where $\bar{Y}_j^R$ and s_j^R (respectively $\bar{Y}_j^S$ and s_j^S) are the sample mean and sample standard deviation of the metric Y_j computed across the n real-system observations (respectively simulation replications). A metric is flagged as exhibiting a systemic shift whenever $|D_j|$ exceeds a critical value at significance level $\alpha = 0.05$, which is approximately 1.96 for sufficiently large n. If no metric is flagged, the simulation is considered consistent with the observed system behavior and no further action is taken. We note that D_j need not be restricted to the form above; more generally, it can be any discrepancy measure the simulation practitioner defines to capture meaningful differences between the real system and the simulation.

(2) **Signal Construction**: For each component θ_k, we extract a vector of component-level diagnostic features E_k, namely the utilization, queue length, failure rate, scrap count, and blocking rate, from the real-system event log L^R and, independently, from the simulation log L^S. This yields a paired signal (E_k^R, E_k^S) per component, which the LLM-based attribution stage will use to contrast real-world behavior against its simulation counterpart.

(3) **LLM-Based Attribution**: We use an LLM to attribute observed drift to specific components by reasoning over the signals assembled in the previous steps. To be specific, we include the available evidence into a structured context $C = (\mathscr{G}, D, E, \mathscr{K})$ comprising four components:

- **System graph** $\mathscr{G}$: A description of how the system is structured, the components (machines, buffers, transport resources) and their relationships (material flow, shared resources, control dependencies). This enables the LLM to reason about which components could plausibly affect which metrics.
- **Discrepancy vector** D: The observed deviations in the performance metrics $Y_1, \ldots, Y_m$ between the real system and the simulation model, indicating what is wrong and how severe the drift is.
- **Evidence matrix** $E = (E^R, E^S)$: Component-level diagnostic features extracted from both the real-system logs and the simulation logs. By providing paired evidence, the LLM can identify components whose behavior diverges between the two sources.
- **Domain knowledge** $\mathscr{K}$: Expert-provided hints such as known failure modes, recent maintenance events, or historical drift patterns. This allows practitioners to inject contextual information that may not be captured in the logs.

Given C, the LLM produces a score s_k for each component, and the components are ranked in descending order of s_k, so that the component with the highest score appears at the top of the list as the most likely root cause. The prompt instructs the LLM to follow a structured reasoning process with four steps:

1. **Relevance**: First, the LLM identifies which components are structurally connected to the affected metrics via $\mathscr{G}$. This filters out components that cannot plausibly explain the observed drift, narrowing the candidate set.
2. **Discrepancy Alignment**: Next, the LLM evaluates how well each candidate component could explain the direction and magnitude of the observed deviations in D. For example, a slowdown in a bottleneck machine would align with decreased throughput and increased cycle time.
3. **Evidence Consistency**: The LLM then examines the component-level evidence E, looking for abnormal signals and, crucially, for components whose behavior differs between E^R and E^S. A

component showing elevated failure rates in the real system but not in the simulation is a strong root cause candidate.

4. **Aggregation**: Finally, the LLM combines these factors, structural relevance, discrepancy alignment, and evidence consistency, into a single score per component.

An abstracted version of the prompt used to elicit the ranked list is shown below.

Abstracted LLM Prompt

System Instruction: You are an expert in diagnosing performance issues in complex manufacturing systems. Given system structure, observed discrepancies, and supporting log-based evidence, infer the most likely root-cause components.

Input:
System Description: A summary of components and their relationships derived from $\mathscr{G}$.
Observed Discrepancies: Metric-wise deviations from expected behavior based on D.
Component Evidence: Component-level diagnostic features (e.g., utilization, queue lengths, failure rates, repair durations, scrap counts, transport metrics, batch statistics), drawn separately from system-generated logs (E^R) and simulation-generated logs (E^S) so that differences between the two sources can be reasoned about explicitly.

Reasoning Steps:

1. **Relevance:** Identify components structurally connected (via $\mathscr{G}$) to the affected metrics.
2. **Discrepancy Alignment:** Evaluate how well each component explains the observed deviations in D.
3. **Evidence Consistency:** Incorporate abnormal signals from E^R and E^S, paying particular attention to components whose behavior diverges between the real system and the simulation model.
4. **Aggregation:** Combine these factors into a single score.

Output Format: For each component $\theta_k \in \Theta$, compute score s_k and rank them by decreasing score

$$[(\theta_1, s_1), (\theta_2, s_2), \ldots, (\theta_K, s_K)]$$

where s_k is a real-valued score indicating the relative likelihood of θ_k being the root cause. Scores need not be normalized but should be comparable across components.

To evaluate this framework, we instantiate it on a semiconductor manufacturing simulation testbed that exhibits the key characteristics of complex manufacturing systems: re-entrant flows, batch processing, stochastic failures, and resource contention. We describe this testbed in the next section.

3 THE MINI FAB MODEL

In this section, we describe the The Intel Mini Fab model (Spier and Kempf 1995), a simplified reference model for commonly used in literature on the modeling and analysis of semiconductor manufacturing systems. We chose the Mini Fab model as the evaluation testbed for our proposed method because it captures many key characteristics in a complex manufacturing system: multi-product production, batch processing, sequence-dependent setups, and intralogistics resources. As shown in Figure 1, the system comprises three processing cells. Cell 1 (Diffusion) has machines M_{1a} and M_{1b}, which process batches of up to three lots. Cell 2 (Lithography) has machine M_3, which processes individual lots with sequence-dependent setup times. Cell 3 (Ion Implantation) has machines M_{2a} and M_{2b}, which process individual lots and are subject to random failures. The system produces three product types: Product A, Product B, and Test Wafers. The physical unit of production is a lot (a wafer carrier that moves through the fab) and each lot belongs to exactly one product type. All lots follow a six-step route visiting Cells 1, 3, 2, 3, 1, and 2 in this given sequence, making the Mini Fab model an example of a flexible flow shop with re-entrant flows. Between processing steps, wafer lots are stored in stockers associated with each cell, which regulate the flow of jobs across the system. The Cell 1 stocker can hold up to 18 lots (equivalent to six batches),

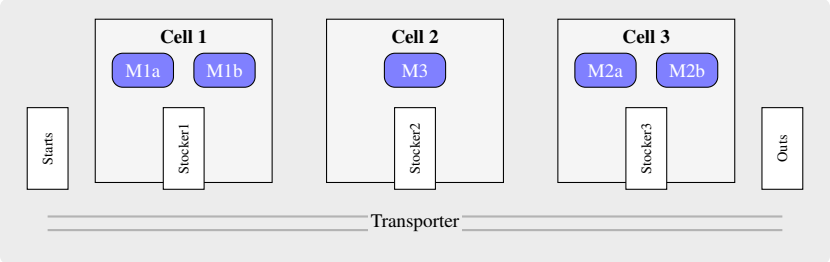

Figure 1: Facility configuration for the Mini Fab model.

Table 1: Processing, load, unload, and PM times by machine and step (minutes); PM interval (hours).

Cell	Machine(s)	Step	Process	Load	Unload	PM Duration	PM Interval (h)
Diffusion	M_{1a}, M_{1b}	1	225	20	40	75	24
		5	255	20	40		
Ion Implantation	M_{2a}, M_{2b}	2	30	15	15	120	8
		4	50	15	15		
Lithography	M_3	3	55	10	10	30	8
		6	10	10	10		

while the Cell 2 and Cell 3 stockers have a capacity of 12 lots each. when a downstream stocker reaches its capacity, upstream processing is halted to prevent overflow.

The remainder of this section details three operational components of the Mini-Fab model, each of which serves as a perturbation target in the experimental design of Section 4. We describe them separately to establish the baseline behavior of our reference model.

3.1 Batching Rules

Batching is performed at Cell 1 (Diffusion) for Steps 1 and 5. Each batch consists of up to three lots, with constraints on composition depending on the processing step. For Step 1, batches may contain any combination of Products A and B, but at most one test wafer is allowed per batch. For Step 5, batches cannot mix Product A and Product B, and test wafers may be included with either product type. If a test wafer is processed on machine M_{1a} during its first visit to Cell 1 (Step 1), it must be processed on machine M_{1b} during its second visit (Step 5), and vice versa. Finally, there is a 30-minute batch wait window: when a machine becomes available and at least one lot is waiting, the system waits up to 30 minutes to accumulate additional lots before processing an incomplete batch. This window will be the primary batching parameter perturbed in Section 4.

3.2 Processing Times

Lots are released at fixed nominal weekly rates: 51 lots/week (Product A), 30 lots/week (Product B), and 3 lots/week (Test Wafers), Within each stocker, lots are dispatched via a First-In-First-Out (FIFO) rule. Processing times and preventive maintenance (PM) schedules are summarized in Table 1. PM operations are scheduled at fixed intervals per machine; if a machine has already failed when PM is due, that interval is skipped. Lot transportation between stockers is handled by a single automated vehicle (capacity: one lot, FCFS); each move incurs 1-minute load and unload times plus 4 minutes of travel per stocker traversed. Machine M_3 (Lithography) incurs sequence-dependent setup times before processing: 0 minutes (same step, same product), 5 minutes (same step, different product), 10 minutes (different step, same product), and 12 minutes (different step, different product). The processing times of each machine type and the transport travel time are the parameters that will be perturbed in Section 4.

3.3 Machine Reliability

Machines M_{2a} and M_{2b} (Ion Implantation) are subject to random failures, with time-to-failure uniform on [24,76] hours. Upon failure, the machine becomes immediately unavailable, any in-process lot is scrapped, and repair takes between 6 and 8 hours (uniform), after which the machine returns to service. The failure rate and repair duration are the parameters that will be perturbed in Section 4.

4 NUMERICAL EXPERIMENTS

4.1 Perturbation-Based Design of Experiments

To evaluate the effectiveness of the proposed attribution framework, we systematically introduce controlled perturbations into the discrete-event simulation model of the Intel Mini Fab described in Section 3. These perturbations act as ground-truth root causes, enabling quantitative assessment of attribution accuracy. Formally, for each experimental instance we designate a subset $\mathscr{I} \subseteq \Theta$ of simulation components whose parameters are perturbed away from the baseline configuration; we refer to $\mathscr{I}$ as the *ground-truth perturbed set*. Each experiment generates a perturbed configuration, runs the simulation, and produces system-level discrepancies in metrics relative to the baseline. We apply multiplicative perturbations at three severity levels (Low, Medium, High), with operator-specific multipliers chosen to produce comparable degradation effects across different perturbation types. For processing time operators (P1–P3), we use increases of $+10\%$, $+25\%$, and $+50\%$. For reliability and transport operators (P4–P6), we use larger multipliers to account for the indirect nature of their impact on system metrics. For capacity and batching operators (P7–P8), we use reductions ranging from 25% to 100%. A summary of specific perturbation operators and their magnitudes is provided in Table 2.

Table 2: Perturbation Operators and Magnitudes

ID	Operator P_k	Low	Medium	High
P1	M_1 run time increase	$\times 1.10$	$\times 1.25$	$\times 1.50$
P2	M_2 run time increase	$\times 1.10$	$\times 1.25$	$\times 1.50$
P3	M_3 run time increase	$\times 1.10$	$\times 1.25$	$\times 1.50$
P4	M_2 failure rate increase (TTF)	$\times 0.80$	$\times 0.60$	$\times 0.40$
P5	M_2 repair time increase	$\times 1.25$	$\times 1.50$	$\times 2.00$
P6	Transport travel time increase	$\times 1.50$	$\times 2.00$	$\times 3.00$
P7	Stocker capacity reduction	$\times 0.75$	$\times 0.50$	$\times 0.33$
P8	Batch wait window reduction	$\times 0.50$	$\times 0.25$	$\times 0.00$

The LLM is tasked with identifying which of these 8 operators ($|\Theta| = 8$) are responsible for observed discrepancies. Each operator represents a distinct perturbation mechanism, and the LLM must distinguish between operators that affect the same physical equipment (e.g., P4 and P5 both degrade M2 availability but through different mechanisms: P4 increases the frequency of failures while P5 extends the duration of each repair).

We construct three families of perturbation scenarios. The first family consists of single-component perturbations, in which each of the 8 operators in Table 2 is applied at all three severity levels (Low/Medium/High), yielding $8 \times 3 = 24$ instances (S1–S24). The second family consists of 9 two-component scenarios (S25–S33, Table 3a), all at Medium magnitude, including both independent perturbations drawn from different categories and paired perturbations targeting components that are known to interact, such as a machine slowdown combined with a reduced downstream stocker capacity (bottleneck amplification). The third family consists of 9 three-component scenarios (S34–S42, Table 3b), also at Medium magnitude, targeting higher-order interactions such as failure cascades, repair-induced congestion, and multi-cell slowdowns whose combined effect on metrics cannot easily be decomposed into per-operator contributions. This yields a total of $24 + 9 + 9 = 42$ scenarios.

Table 3: Two- and Three-Component Perturbation Instances (all at Medium magnitude)

(a) Two-Component (S25–S33)

ID	Components	Mechanism
S25	P1, P6	Processing + Transport
S26	P4, P5	Reliability coupling
S27	P6, P8	Transport + Batching
S28	P1, P3	Multi-cell slowdown
S29	P2, P4	Processing + Reliability
S30	P1, P7	Bottleneck ampl. (C1)
S31	P3, P7	Bottleneck ampl. (C2)
S32	P2, P7	Bottleneck ampl. (C3)
S33	P3, P8	Processing + Batching

(b) Three-Component (S34–S42)

ID	Components	Mechanism
S34	P2, P4, P5	Failure cascade (M_2)
S35	P4, P5, P7	Repair-induced congest.
S36	P3, P6, P7	Congestion ampl.
S37	P1, P2, P3	Multi-cell slowdown
S38	P4, P5, P6	Reliability-transport
S39	P3, P6, P8	Litho-batch-transport
S40	P1, P4, P5	M1-M2 reliability
S41	P6, P7, P8	Transport-buffer-batch
S42	P1, P3, P7	Processing + buffer

4.2 Evaluation Protocol

For each perturbed system, we simulate both the baseline and perturbed systems and compute the discrepancy vector D using throughput and cycle time ($m = 2$ metrics) containing the system-level performance metrics over an 8-day horizon and a 1-day warm-up period. Cycle time is defined as the elapsed time from a lot's arrival into the system until it exits. In parallel, we extract component-level evidence E from the event logs of both systems. The evidence vector includes diagnostic features such as machine utilization, failure rates, repair durations, scrap counts, per-stocker WIP levels, batch sizes, full-batch fractions, batch-wait times, transport move rates, blocking events, and mean move durations. These features are standard outputs of Manufacturing Execution Systems (MES) and Automated Material Handling Systems (AMHS) in modern semiconductor fabs, making the evidence catalogue practically realizable. The system description $\mathscr{G}$ encodes the structural topology of the manufacturing system: the set of machines, stockers, and the transport vehicle, their material flow relationships, the re-entrant routing sequence, and the control dependencies such as batching and blocking rules. An example of how $\mathscr{G}$ is specified in a prompt is provided in Appendix A.

The attribution framework consumes the structured context $C = (\mathscr{G}, D, E)$, built from $n = 30$ independent replications of both the baseline and perturbed systems, and produces a score for every candidate component $\theta_k \in \Theta$, which is used to rank the components from most to least likely perturbed. We use Claude Sonnet 4 as the attribution model with temperature set to 0. The temperature parameter controls the randomness of LLM outputs; setting it to 0 concentrates probability mass on the highest-scoring tokens, substantially reducing output variability. Attribution performance is evaluated by comparing this ranking against the ground-truth perturbed set $\mathscr{I}$ using Top-k Recall: $\text{Recall@}k = \frac{|\mathscr{I} \cap \text{Top}_k(C)|}{|\mathscr{I}|}$, where $\text{Top}_k(C)$ denotes the k highest-ranked components by the LLM. Intuitively, Recall@k measures the fraction of true root causes recovered within the top k components ranked by LLM. In the remainder of the paper, we focus on Recall@3 as it reflects a realistic triage budget in practice, where a domain expert would inspect a short shortlist rather than the full candidate set, and with $|\Theta| = 8$ candidates, top-3 covers a meaningful fraction of the search space without being trivially large. For Recall@3, a value of 1.0 indicates that all truly perturbed components are recovered within the top three predictions, while 0.0 indicates that none are recovered. A complete, verbatim prompt-response pair for a representative three-component scenario is provided in Appendix A.1.

4.3 LLM Attribution Results

To account for potential variability in LLM inference, we run each attribution query three times and average the component scores across runs before ranking. This score-averaging protocol yields stable rankings: the standard deviation of individual component scores across runs is typically below 0.02, confirming that temperature-zero inference produces highly consistent outputs. Across all 42 scenarios, the LLM achieves

a mean Recall@3 of 0.75. A random ranker that assigns scores uniformly at random over the $|\Theta| = 8$ candidates serves as a natural baseline. Intuitively, each true root cause has an equal probability of $3/8$ of landing in the top 3. Since this probability is the same for every element of $\mathscr{I}$, the $|\mathscr{I}|$ terms cancel in the Recall@3 formula above, giving an expected random baseline of $3/8 \approx 0.38$ regardless of the number of true root causes. The proposed framework achieves a mean Recall@3 of 0.75, outperforming this random baseline by a substantial margin. Moreover, we observe that the LLM attribution recovers all ground-truth operators within its top-three predictions on 26 (62%) of the 42 scenarios.

Table 4 reports Recall@3 stratified by the number of perturbed operators. Single-component scenarios achieve the highest recall (0.79), while performance degrades modestly for multi-component scenarios. Interestingly, three-component scenarios (0.70) slightly outperform two-component scenarios (0.67). This small difference (0.03) should be interpreted cautiously given the limited sample size (9 scenarios per family). One plausible explanation is the operator composition: as shown in Table 5, reliability and transport operators achieve perfect recall due to their distinctive evidence signatures. Three-component scenarios include 15 appearances of these high-recall operators (P4, P5, P6) compared to only 5 in two-component scenarios.

Table 4: Recall@3 of the LLM attribution framework stratified by perturbation family.

Family	Number of scenarios	Recall@3
Single-component ($\|\mathscr{I}\| = 1$)	24	0.79
Two-component ($\|\mathscr{I}\| = 2$)	9	0.67
Three-component ($\|\mathscr{I}\| = 3$)	9	0.70
All	42	**0.75**

Table 5 reports Recall@3 aggregated by operator category. Reliability and Transport categories achieve perfect recall of 1.00, as their effects manifest through distinctive signals in the evidence catalog. The M2 failure-rate operator (P4) produces elevated scrap counts and failure rates, while the M2 repair-time operator (P5) is detected through explicit repair-duration tracking. Transport perturbations (P6) are similarly identifiable through move-time statistics. This highlights the importance of designing evidence catalogs with features that discriminate between specific perturbation mechanisms, not just between physical components. On the other hand, Processing, Batching, and Buffer categories prove more challenging, with recall ranging from 0.50 to 0.58. The LLM frequently misattributes processing slowdowns (P1, P2) to transport or buffer congestion, since slower processing backs up upstream stockers and triggers blocking events that dominate the evidence. This cascade effect illustrates a fundamental challenge in root-cause attribution: downstream symptoms can mask upstream causes.

Table 5: Recall@3 by operator category across all scenarios involving operators in that category.

Category	Number of scenarios	Recall@3
Reliability	17	1.00
Transport	9	1.00
Processing	26	0.58
Batching	7	0.57
Buffer	10	0.50

5 CONCLUDING REMARKS

We formalized root-cause attribution of simulation model drift as an operator-level ranking problem and showed that a general-purpose LLM (Claude Sonnet 4), conditioned on a structured context, correctly identifies the true root causes within its top-3 predictions on the Mini-Fab testbed in 62% of scenarios, and

recovers on average 75% of true root causes within the top-3 ranked components across all 42 scenarios, substantially above the 38% expected from a random ranker. The framework produces an automatable top-3 shortlist that can help domain experts quickly focus their investigation on the most likely causes of model drift, reducing the manual effort required to maintain simulation model fidelity over time.

Our evaluation is confined to the Mini-Fab testbed with a fixed candidate set of $|\Theta| = 8$ and uses the simulator itself as the "real system", so the results do not capture sensor noise, missing events, non-stationary baselines, out-of-catalogue perturbations, or the abstention question that the current formulation always side-steps by returning a non-empty ranking. The evidence features we extract, machine utilization, failure rates, scrap counts, repair durations, batch statistics, transport metrics, and per-stocker WIP levels, are standard outputs in modern semiconductor fabs' logs, making the approach practically realizable. Future work targets (i) investigating hierarchical attribution that first identifies affected equipment and then narrows to specific mechanisms, (ii) developing causal reasoning approaches that can distinguish root causes from cascade effects, and (iii) broader empirical validation on industrial data.

A APPENDIX

A.1 Example LLM Prompt

Here is the full prompt and its LLM response for scenario `S38` (Reliability-transport stress). This scenario is a three-operator perturbation in which the ground-truth perturbed set is $\mathscr{I} = \{\text{P4}, \text{P5}, \text{P6}\}$, corresponding to M2 failure rate increase, M2 repair time increase, and transport travel time increase.

Prompt (Scenario S38)

```
SYSTEM INSTRUCTION:
You are an expert in diagnosing performance issues in complex manufacturing
systems. Given the system structure, observed discrepancies between a real
system and its simulation model, and supporting log-based evidence, infer
the most likely root-cause components.

Candidate components:
Theta = {P1, P2, P3, P4, P5, P6, P7, P8}

System Description (G):
The Intel MiniFab is a re-entrant semiconductor fab with three cells
(C1 Diffusion, C2 Lithography, C3 Ion Implantation) linked by a single
automated transporter.
Route: Start -> Step 1 (C1) -> Step 2 (C3) -> Step 3 (C2)
       -> Step 4 (C3) -> Step 5 (C1) -> Step 6 (C2) -> Exit.
Machine types: M1a/M1b (batch diffusion furnaces, Cell 1), M2a/M2b
(ion implanters with random failures, Cell 3), M3 (lithography with
sequence-dependent setups, Cell 2).
Candidate perturbation operators for attribution:
  - P1 (processing, cell=C1, steps=S1,S5): Increases the processing time
    of diffusion furnaces M1a/M1b at Steps S1 and S5.
  - P2 (processing, cell=C3, steps=S2,S4): Increases the processing time
    of ion implanters M2a/M2b at Steps S2 and S4.
  - P3 (processing, cell=C2, steps=S3,S6): Increases the processing time
    of lithography tool M3 at Steps S3 and S6.
  - P4 (reliability, cell=C3, steps=S2,S4): Reduces time-to-failure for
    M2a/M2b, increasing the frequency of random failures.
  - P5 (reliability, cell=C3, steps=S2,S4): Increases the repair duration
    after M2a/M2b failures.
  - P6 (transport, cell=-, steps=-): Increases the inter-stocker travel
    time for the automated transport vehicle.
  - P7 (buffer, cell=-, steps=S1,S2,S3,S4,S5,S6): Reduces the capacity
    of stockers C1, C2, and C3.
  - P8 (batching, cell=C1, steps=S1,S5): Reduces the batch-wait window
    for M1a/M1b, causing batches to start with fewer lots.

Observed Discrepancies (D):
Metrics flagged as drifted (|z| >= 2, sorted by |z|):
  - throughput_lph: baseline=0.394 (sd=0.0252, n=30)
                 -> perturbed=0.347 (sd=0.0399, n=30);
    delta=-0.0464 (-11.8%), z=-5.38
  - cycle_time_mean: baseline=2.36e+03 (sd=176, n=30)
                  -> perturbed=2.63e+03 (sd=294, n=30);
    delta=+270 (+11.4%), z=+4.32

Other metrics (not flagged, sorted by |z|):
  - cycle_time_std: delta=+53.3 (+10.4%), z=+1.49
```

```
Component Evidence from the real system (E^sys):
  [Step-level WIP (lots waiting at each cell stocker)]
    C1: WIP_C1_mean=8.71, WIP_C1_std=1.45
    C2: WIP_C2_mean=2.99, WIP_C2_std=0.797
    C3: WIP_C3_mean=2.42, WIP_C3_std=0.644
    Total: WIP_total_mean=18.69, WIP_total_std=3.15

  [Per-operator diagnostic features]
    P1: M1_utilization_pct=96.55, M1_utilization_std=0.968
    P2: M2_utilization_pct=68.23, M2_utilization_std=8.99
    P3: M3_utilization_pct=75.72, M3_utilization_std=5.57
    P4: M2_failure_rate_per_hr=0.0239, M2_scrap_rate_per_hr=0.0458,
        scrap_count=17.60, scrap_count_std=7.96
    P5: M2_mean_repair_minutes=635.9, M2_repair_minutes_std=51.69,
        M2_failure_rate_per_hr=0.0239
    P6: moves_per_hr=2.77, blocks_per_hr=0.0181, mean_move_minutes=25.64,
        blocks_per_hr_std=14.63
    P7: mean_occupancy=4.71, occupancy_std=0.962, block_rate_per_hr=0.00602
    P8: mean_batch_size=2.12, full_batch_fraction=0.45,
        mean_batch_wait_min=16.77, batch_wait_std=14.88

Component Evidence from the simulation model (E^sim):
  [Step-level WIP (lots waiting at each cell stocker)]
    C1: WIP_C1_mean=9.19, WIP_C1_std=1.23
    C2: WIP_C2_mean=3.01, WIP_C2_std=0.412
    C3: WIP_C3_mean=1.44, WIP_C3_std=0.169
    Total: WIP_total_mean=18.30, WIP_total_std=1.76

  [Per-operator diagnostic features]
    P1: M1_utilization_pct=97.30, M1_utilization_std=1.17
    P2: M2_utilization_pct=68.16, M2_utilization_std=4.50
    P3: M3_utilization_pct=83.57, M3_utilization_std=3.21
    P4: M2_failure_rate_per_hr=0.0159, M2_scrap_rate_per_hr=0.0219,
        scrap_count=8.40, scrap_count_std=5.17
    P5: M2_mean_repair_minutes=421.5, M2_repair_minutes_std=33.87,
        M2_failure_rate_per_hr=0.0159
    P6: moves_per_hr=3.00, blocks_per_hr=0.000521, mean_move_minutes=11.49,
        blocks_per_hr_std=0.548
    P7: mean_occupancy=4.55, occupancy_std=0.602, block_rate_per_hr=0.000174
    P8: mean_batch_size=2.26, full_batch_fraction=0.509,
        mean_batch_wait_min=15.17, batch_wait_std=14.98

Reasoning Steps:
  1. Relevance: Identify components structurally connected (via G) to the
     metrics that show drift.
  2. Discrepancy Alignment: Evaluate how well each candidate component
     explains the observed deviations in D.
  3. Evidence Consistency: Incorporate abnormal signals from E^sys and
     E^sim, paying particular attention to components whose behaviour
     diverges between the system-generated and simulation-generated logs.
  4. Aggregation: Combine the above factors into a single real-valued
     score per component.

Output Format:
For each component theta_k in Theta, emit a real-valued score s_k expressing
the relative likelihood that theta_k is a root cause (scores need not be
normalised, but they must be comparable across components). Return a JSON
array of objects with fields "component" and "score", sorted by decreasing
score. Include every component in Theta exactly once.
```

A.2 Response

LLM Response (Scenario S38)

```
Looking at this systematically:

## 1. Relevance Analysis
All components can affect throughput and cycle time in this re-entrant
system, so all are structurally relevant.

## 2. Discrepancy Alignment
The key observations are:
- Throughput decreased by 11.8% (z=-5.38)
- Cycle time increased by 11.4% (z=+4.32)

## 3. Evidence Analysis

**P5 (M2 repair duration)** shows the strongest signal:
- Real system: 635.9 min mean repair time
- Simulation: 421.5 min mean repair time
- 51% longer repairs in reality -> extended downtime -> reduced throughput
```

```
**P4 (M2 failure frequency)** also shows significant deviation:
- Real system: 0.0239 failures/hr, 17.6 scrap count
- Simulation: 0.0159 failures/hr, 8.4 scrap count
- 50% higher failure rate -> more downtime and scrap

**P6 (transport time)** shows major difference:
- Real system: 25.64 min mean move time
- Simulation: 11.49 min mean move time
- 123% longer transport times -> system-wide delays

**P3 (lithography processing)** shows capacity mismatch:
- Real system: 75.72% utilization
- Simulation: 83.57% utilization
- Lower real utilization suggests processing issues

**P8 (batching efficiency)**:
- Real system: 2.12 mean batch size, 45% full batches
- Simulation: 2.26 mean batch size, 51% full batches
- Reduced batching efficiency in reality

## 4. Scoring

The evidence strongly points to M2 reliability issues (P4, P5) and
transport problems (P6) as primary causes, with secondary contributions
from lithography processing (P3) and batching inefficiency (P8).

[
  {"component": "P5", "score": 0.92},
  {"component": "P4", "score": 0.88},
  {"component": "P6", "score": 0.75},
  {"component": "P3", "score": 0.45},
  {"component": "P8", "score": 0.35},
  {"component": "P2", "score": 0.15},
  {"component": "P7", "score": 0.12},
  {"component": "P1", "score": 0.08}
]
```

Remarks. The M2 repair duration shows a 51% increase between E^{sim} (421.5 min) and E^{sys} (635.9 min), providing an unambiguous signal that the LLM correctly interprets as evidence of P5 (repair time increase). The M2 scrap count doubling and elevated failure rate lead the model to rank P4 (failure rate increase) second. The transport move time shows a 123% increase (11.49 min to 25.64 min), which the LLM correctly identifies as evidence of P6 (transport travel time increase). The step-level WIP evidence shows C3 WIP increased in the real system (2.42 vs 1.44), consistent with M2 reliability issues causing congestion in Cell 3. The key result is that all three ground-truth operators (P4, P5, P6) are recovered within the top-3, achieving perfect Recall@3 on this scenario.

REFERENCES

Akcay, A., S. Biller, B. P. Gan, C. Laroque, and G. Shao. 2023. "Maintenance and operations of digital manufacturing twins". In *proceedings of the 2023 winter simulation conference*.

Behrendt, S., T. Altenmüller, M. C. May, A. Kuhnle, and G. Lanza. 2026. "Real-to-sim: Automatic simulation model generation for a digital twin in semiconductor manufacturing". *Journal of Intelligent Manufacturing* 37(2):829–848.

Botello, J. G., B. Llinas, J. J. Padilla, and E. Frydenlund. 2025. "Toward Automating System Dynamics Modeling: Evaluating Llms in the Transition From Narratives to Formal Structures". In *2025 Winter Simulation Conference (WSC)*, 2380–2391. IEEE.

Braakman, K. J., D. M. Knotter, A. Akçay, and I. Adan. 2025. "Leveraging machine signals for device-level quality detection and automatic root cause analysis in semiconductor wire bonding". *IEEE Transactions on Semiconductor Manufacturing*.

Corlu, C. G., A. Akcay, and W. Xie. 2020. "Stochastic simulation under input uncertainty: A review". *Operations Research Perspectives* 7:100162.

Dave, A. J., T. N. Nguyen, and R. B. Vilim. 2024. "Integrating LLMs for explainable fault diagnosis in complex systems". *arXiv preprint arXiv:2402.06695*.

Dehghanimohammadabadi, M., S. Belsare, and N. Sadeghi. 2025. "A Tutorial on Generative AI and Simulation Modeling Integration". In *2025 Winter Simulation Conference*, 1197–1211. IEEE.

Elbasheer, M., Y. Laili, F. Longo, V. Solina, Y. Tao, P. Veltri, *et al*. 2025. "Natural language-driven production planning: integrating large language models with automatic simulation model generation in manufacturing systems". *Journal of Intelligent Manufacturing*:1–28.

Feldkamp, N. 2025. "On the Use of Generative AI in Simulation Studies: A Review of Techniques, Applications and Opportunities". In *2025 Winter Simulation Conference*, 1–12. IEEE.

Guo, Z., F. Tang, L. Luo, M. Zhao, and N. Kato. 2025. "A survey on applications of large language model-driven digital twins for intelligent network optimization". *IEEE Communications Surveys & Tutorials*.

He, H., X. Liu, and X. Deng. 2025. "Model Validation and LLM-Based Model Enhancement for Analyzing Networked Anagram Experiments". In *2025 Winter Simulation Conference (WSC)*, 2003–2014. IEEE.

Law, A. M. 2015. *Simulation Modeling and Analysis*. 5th ed. New York: McGraw-Hill Education.

Marandi, S., Y.-S. Hu, and M. Modarres. 2025. "Complex system diagnostics using a knowledge graph-informed and large language model-enhanced framework". *Applied Sciences* 15(17):9428.

Möltner, T., P. Manzl, M. Pieber, and J. Gerstmayr. 2025. "Creation, evaluation and self-validation of simulation models with large language models". *Neurocomputing*:132030.

Qi, Z., J. Ren, W. Gan, and P. S. Yu. 2025. "Large Language Models for Fault Diagnosis". In *2025 IEEE International Conference on Big Data (BigData)*, 6982–6991. IEEE.

Selak, M., D. Krechel, A. Ulges, S. Spieckermann, N. Stoehr, and A. Loehr. 2025. "LLM Assisted Value Stream Mapping". In *2025 Winter Simulation Conference*, 2015–2026. IEEE.

Spier, J., and K. Kempf. 1995. "Simulation of emergent behavior in manufacturing systems". In *Proceedings of SEMI Advanced Semiconductor Manufacturing Conference and Workshop*, 90–94. IEEE.

Sztyber-Betley, A., E. Chanthery, L. Travé-Massuyès, S. Merkelbach, K. Kukla, M. Glotin, *et al*. 2025. "Are Diagnostic Concepts Within the Reach of LLMs?". In *36th International Conference on Principles of Diagnosis and Resilient Systems (DX 2025)*, 2–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

van der Pas, M., A. Akçay, R. Dijkman, and I. Adan. 2024. "Combining case-based reasoning and process mining for root cause analysis in complex manufacturing environments". *Manufacturing Letters*.

van der Pas, M., R. Dijkman, A. Akçay, I. Adan, and J. Walker. 2023. "On-Demand and Model-Driven Case Building Based on Distributed Data Sources". *Manufacturing Letters*.

Xia, Y., D. Dittler, N. Jazdi, H. Chen, and M. 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)*, 1–4. IEEE.

AUTHOR BIOGRAPHIES

SANKET BHAT is a Senior Applied Scientist at Amazon. His research interests include applications of optimization and AI techniques to manufacturing systems, supply chain management, and conversational AI. His e-mail address is sanobhat@amazon.com.

ALP AKCAY is an Associate Professor in the Department of Mechanical and Industrial Engineering at Northeastern University. He received his Ph.D. in Operations Management and Manufacturing from Carnegie Mellon University. His research focuses on the planning and control of manufacturing systems and supply chains, using techniques from stochastic operations research, machine learning, and simulation. His email address is a.akcay@northeastern.edu.

TUGCE MARTAGAN is an Associate Professor in the Department of Mechanical and Industrial Engineering at Northeastern University. Her research interests include stochastic modeling and optimization with applications in manufacturing and supply chains. Her email address is t.martagan@northeastern.edu.