IndustryAssetEQA: A Neurosymbolic Operational Intelligence System for Embodied Question Answering in Industrial Asset Maintenance

Chathurangi Shyalika
Artificial Intelligence Institute,
University of South Carolina
USA
jayakodc@email.sc.edu

Dhaval Patel
Software Innovation Lab
IBM Yorktown
USA
pateldha@us.ibm.com

Amit Sheth
Artificial Intelligence Institute,
University of South Carolina
Indian AI Research Organization
amit@sc.edu, amit@iairo.ai

Abstract

Industrial maintenance environments increasingly rely on AI systems to assist operators in understanding asset behavior, diagnosing failures, and evaluating interventions. Although large language models (LLMs) enable fluent natural-language interaction, deployed maintenance assistants routinely produce generic explanations that are weakly grounded in telemetry, omit verifiable provenance, and offer no testable support for counterfactual or action-oriented reasoning that undermine trust in safety-critical settings. We present **IndustryAssetEQA**, a neurosymbolic operational intelligence system that combines episode-centric telemetry representations with a Failure Mode and Effects Analysis Knowledge Graph (FMEA-KG) to enable Embodied Question Answering (EQA) over industrial assets. We evaluate on four datasets covering four industrial asset types, including rotating machinery, turbofan engines, hydraulic systems, and cyber–physical production systems. Compared to LLM-only baselines, IndustryAssetEQA improves structural validity by up to $+0.51$, counterfactual accuracy by up to $+0.47$, and explanation entailment by $+0.64$, while reducing severe expert-rated overclaims from 28% to 2% ($\approx$ 93%). Code, datasets, and the FMEA-KG are available at: https://github.com/IBM/AssetOpsBench/tree/IndustryAssetEQA/IndustryAssetEQA.

1 Introduction

Industrial asset maintenance is rapidly shifting from periodic inspections and offline analysis toward continuous, data-driven decision making (Patel et al., 2025; Seneviratne et al., 2018). Modern plants generate large volumes of multivariate telemetry, alert streams, and maintenance records. Operators increasingly rely on AI-based operational intelligence systems to interpret this information (Peres et al., 2020; Lee et al., 2020).

Many deployed and prototype systems today use large language models (LLMs) with natural-language interfaces layered over documents, dashboards, and structured data. While these systems produce fluent explanations, practitioner feedback and recent analyses report recurring reliability failures (Ji et al., 2023; Schmidtová et al., 2025): (i) generic explanations that cite textbook failure patterns without grounding in the episode's sensors or context; (Ji et al., 2023; Schmidtová et al., 2025) (ii) missing verifiable provenance as answers rarely cite the supporting time window, sensors, events, or maintenance records, hindering auditability; (Chen et al., 2021) and (iii) non-testable counterfactuals and action suggestions that state qualitative effects without an explicit model of how failure risk changes under interventions. These failures show that LLM-centered QA frames maintenance as language generation rather than a sensor-and-risk decision problem; we therefore require time-situated, provenance-backed, risk-anchored QA systems.

The field of *Embodied AI* provides a natural conceptual foundation for addressing this gap. Embodied AI concerns agents with sensors and actuators that perceive the environment, reason about causes, predict the effects of interventions, and act accordingly (Fung et al., 2025). This *perception–prediction–reasoning–decision* loop is structurally isomorphic to industrial maintenance workflows: sensing telemetry, diagnosing faults, planning interventions, and executing maintenance actions. We argue that industrial maintenance QA should be formulated as an *embodied decision problem* rather than purely textual question answering. In this setting, a system must satisfy four requirements:

- **Time-situated:** Answers must explicitly reference the relevant telemetry window and asset context.
- **Evidence-grounded:** Claims must include verifiable provenance linking to sensors, engineered features, events, and maintenance records.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track), pages 716–735
July 2-7, 2026 ©2026 Association for Computational Linguistics

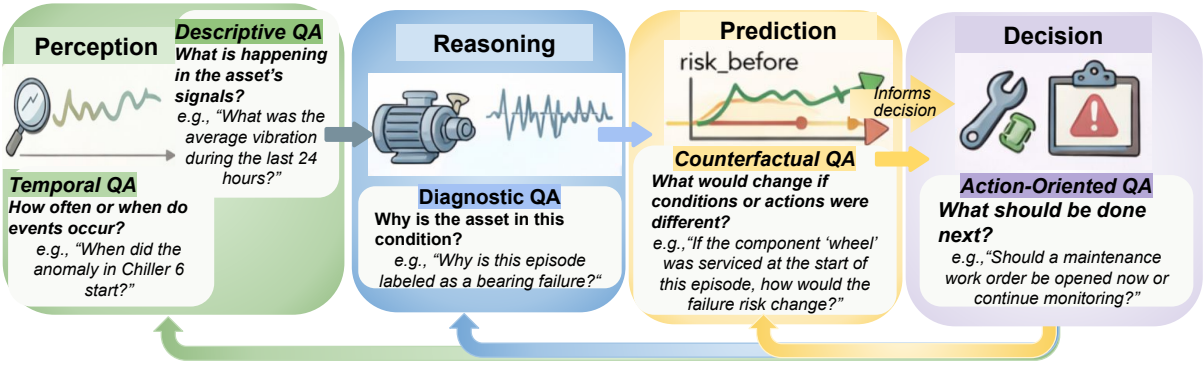

Figure 1: QA taxonomy mapped to the perception–reasoning–prediction–decision loop. Forward arrows indicate cognitive dependencies; backward arrows show the embodied interaction loop (decisions → actions → system → observations).

- **Risk-constrained:** Counterfactual predictions and action recommendations must be derived from an explicit probabilistic risk model.
- **Knowledge-grounded:** Explanations must align with domain semantics and respect structured failure constraints.

To operationalize this formulation, we present **IndustryAssetEQA**, a neurosymbolic operational intelligence system for *Embodied Question Answering* over industrial asset operations. Neurosymbolic integration combines neural learning with symbolic reasoning to enable transparent, knowledge-aligned decision support (Sheth et al., 2023).

Our key contributions are:

- We introduce IndustryAssetEQA, a deployment-oriented embodied QA framework for asset maintenance, and a dataset of structured, time-situated episode facts spanning five QA task types.
- We construct a domain-specific Failure Mode and Effects Analysis Knowledge Graph (FMEA-KG) to enable neurosymbolic grounding of industrial failure semantics across assets and components.
- We implement a parametric intervention-style risk estimator trained on time-series telemetry, which yields testable estimates of risk direction and magnitude under maintenance interventions.
- We design provenance- and structure-aware evaluation protocols that measure structural validity, evidence grounding, temporal alignment, and counterfactual consistency to assess deployable reliability in industrial QA.

2 Related Work

Industrial AI and QA for Asset Maintenance. Prior work spans autonomous industrial agents and evaluation (Patel et al., 2025), domain-specific QA (Terziyan et al., 2025), and decision-support systems based on knowledge-enhanced LLMs for manufacturing (Chatterjee and Dethlefs, 2022; Lee et al., 2023; Lin et al., 2025; Constantinides et al., 2025a). These systems typically operate on static documents and global datasets.

Time Series and Sensor-Grounded Question Answering. Recent work reformulates temporal reasoning as time series QA, enabling querying over numerical sequences, multimodal signals, and temporal events (Constantinides et al., 2025b; Su et al., 2025; Kong et al., 2025; Chen et al., 2025; Wang et al., 2025; Uddin et al., 2024). These approaches introduce benchmarks and models for structured reasoning, multimodal fusion, and error correction, including reviewer-based CoT verification (Su et al., 2025), large-scale multitask time series datasets (Kong et al., 2025; Wang et al., 2025), and cross-modal temporal QA combining text and time series (Chen et al., 2025). However, even strong LLMs struggle with long-range dependencies, causal interpretation, parallel event reasoning, and zero-shot temporal generalization (Uddin et al., 2024; Merrill et al., 2024).

Embodied Question Answering and Decision-Centric AI. Prior work on embodied question answering extends perception-driven QA to interactive 3D environments (Li et al., 2026), incorporating external knowledge (Tan et al., 2021), planning (Ginting et al.), and domain-specific benchmarks for industrial and safety-critical settings (Li et al., 2026; Tan et al., 2021; Ginting et al.; Majumdar et al., 2024; Li et al., 2025a,b; Zhang et al., 2025). While these approaches advance embodied perception, memory, and language grounding, they largely evaluate semantic, spatial, or temporal understanding from observations or simulators, and do not

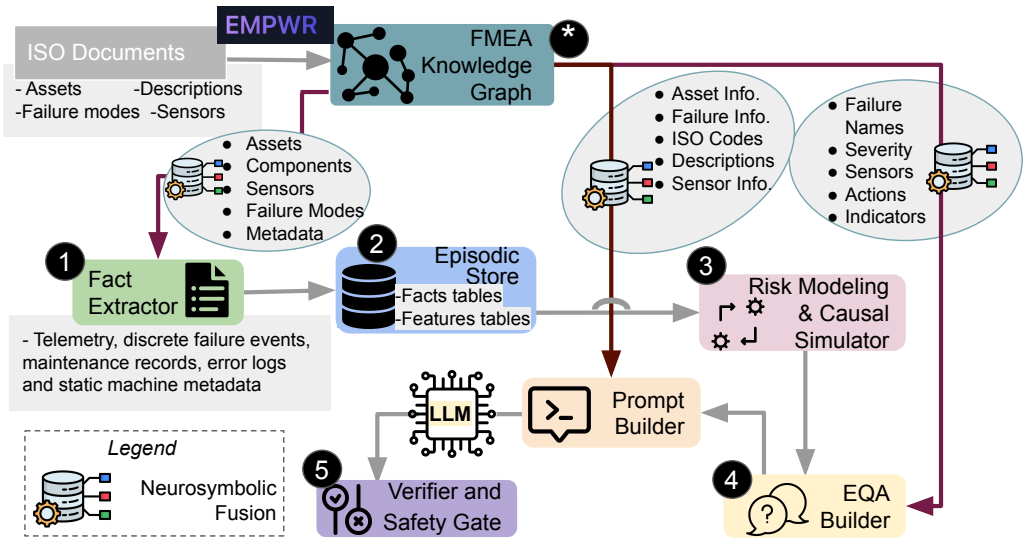

Figure 2: IndustryAssetEQA Architecture integrating Fact Extractor, Episodic Store, FMEA-KG, Causal Simulator, Verifier and Safety Gate

explicitly verify the correctness or risk consistency of operational decisions.

3 Industrial Setting and QA Task Taxonomy

We target industrial asset maintenance (manufacturing, utilities, process plants) where operators inspect multivariate telemetry, alerts, and maintenance logs to make time-sensitive decisions. Each query is framed as *episode-centric (an asset + a concrete time window)* and evaluates whether answers are (i) semantically grounded in episodic and KG evidences, and (ii) when relevant consistent with a simple, data-driven risk model for interventions. The framework covers five compact QA types and are directly aligned with perception → reasoning → prediction → decision pipeline as visualized in Figure 1.

4 IndustryAssetEQA System Overview

Figure 2 illustrates the overall architecture that includes the core components of IndustryAssetEQA.

FMEA Knowledge Graph and Neurosymbolic Fusion. We integrate domain knowledge derived from expert-curated Failure Mode and Effects Analysis (FMEA) specifications from ISO-documentation in the form of a machine-interpretable knowledge graph using the EMPWR platform (Yip and Sheth, 2024). The graph comprises **63 failure modes** mapped to **9 asset categories** and their component types (Appendix A).

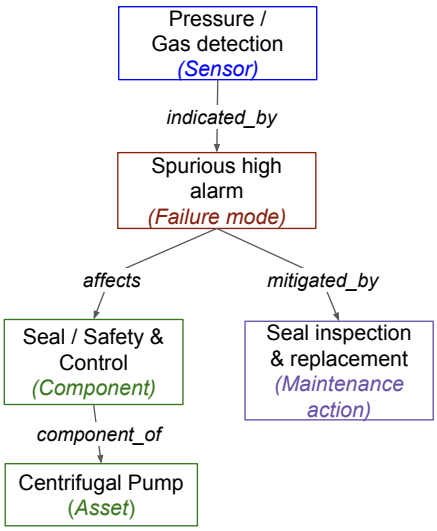

Figure 3: Representative FMEA-KG fragment

Nodes represent asset classes, subcomponents, failure modes, sensor abstractions, and maintenance actions, and edges capture relations such as *affects*, *component_of*, *indicated_by*, and *mitigated_by* (Figure 3). Failure mode nodes are associated with expected telemetry signatures, severity and occurrence metadata, and admissible interventions. FMEA-KG serves as a symbolic grounding (via neurosymbolic fusion) layer that interfaces with neural components at well-defined points in the pipeline using *rdflib* (RDFLib Developers, 2026). Extracted nodes and relations are used as semantic context, so LLM outputs are auditable, domain-valid explanations that link sensor patterns to legitimate failure modes and actions.

Fact Extractor. It converts heterogeneous time series data into structured episode-level facts. For each failure occurring at time t_f on machine m, the extractor defines a configurable historical window $[t_f - \Delta,\, t_f]$ and aggregates all telemetry samples in that interval. For every numeric sensor signal s it computes a compact set of summary descriptors $\mathcal{F}_s = \{\mu_s, \sigma_s, \min_s, \max_s, \mathrm{trend}_s\}$, where μ_s and σ_s are the sample mean and standard deviation, and trend_s is the slope of a least-squares linear fit to the samples in the window. It forms contextual features such as error event counts, number of distinct error types, and component-specific *hours_since_last_maintenance*. Static attributes (e.g., machine age and one-hot model indicators) are appended to the feature vector. Healthy episodes are sampled by selecting center timestamps t with no failures in a subsequent horizon $[t,\, t + H]$ and building the corresponding history window $[t - \Delta,\, t]$. Each episode is labeled ("failure" or "healthy"), enriched with FMEA-KG (asset profile, failure profile, sensor semantics), and emitted as a JSONL record with full provenance that include source files, time ranges, and counts (Example in Appendix B).

Episodic Store. This is a lightweight persistence and retrieval layer for episode-level facts. Serialized facts are ingested and stored in a SQLite database with a two-table schema: a `facts table` that retains each episode's full serialized metadata and a `features table` that decomposes each episode's feature vector into atomic (*feature_name*, *feature_value*) rows. Each episode f is indexed by a stable identifier fact_id and associated metadata (e.g., asset_id, label, source file, and row index). Storing features in an indexed, per-feature form enables efficient numeric queries and threshold searches of the form $\{\, f_i \mid x_{i,j} \bowtie \tau \,\}$, where $x_{i,j}$ denotes the value of feature j in episode i, τ is a numeric threshold, and $\bowtie \in \{<, \leq, =, \geq, >\}$. The store exposes a compact API for retrieval by asset, label, feature constraints, and balanced sampling across labels (See Appendix C).

Risk Modeling and Causal Simulator. Implements a compact, data-driven risk model and a parametric counterfactual estimator. Given an episode fact represented by a fixed-length feature vector $\boldsymbol{x}$, the trainer fits a multinomial logistic regression to approximate the conditional distribution $P(y \mid \boldsymbol{x})$ over discrete episode labels y (e.g., `failure modes` and `healthy`). At inference the simulator interprets a maintenance or sensor intervention as an explicit substitution $\boldsymbol{x} \mapsto \boldsymbol{x}^{\mathrm{do}}$ (i.e., setting one or more coordinates to new values) and computes the pre- and post-intervention risks as:

$$\begin{aligned} r_{\mathrm{before}} &= 1 - P\big(y = \mathrm{healthy} \mid \boldsymbol{x}\big), \\ r_{\mathrm{after}} &= 1 - P\big(y = \mathrm{healthy} \mid \boldsymbol{x}^{\mathrm{do}}\big), \end{aligned} \tag{1}$$

so that the estimated (signed) risk change is:

$$\Delta r \;=\; r_{\mathrm{after}} - r_{\mathrm{before}}. \tag{2}$$

The simulator returns class-probability maps and a lightweight confidence heuristic based on probability extremity (Details in Appendix D). It outputs the pre- and post-intervention probability vectors

$$\big\{P(y = k \mid \boldsymbol{x})\big\}_k \qquad \text{and} \qquad \big\{P(y = k \mid \boldsymbol{x}^{\mathrm{do}})\big\}_k.$$

EQA Builder. Given an episodic fact, EQA Builder deterministically constructs task-specific questions, derives answers from telemetry-derived features and the risk model, and assembles structured reasoning traces with provenance. Where available, it enriches QA instances by querying the FMEA-KG to retrieve failure-mode metadata (e.g., severity, associated sensors, typical indicators, recommended actions) and asset descriptors (equipment category and class)(Samples in Appendix E).

Prompt Builder. A deterministic, safety-aware interface converting an episodic fact and QA into scoped LLM prompts. For each fact, it renders a compact, human- and machine-readable evidence block containing `provenance` (`fact_id`, `asset`, `source`, `row`), `the episode window`, `key diagnostic features`, and `FMEA context` when available. The builder then maps the QA task type to a short, task-specific instruction and produces an explicit user instruction that (a) asks the question, (b) constrains the LLM to use only the supplied evidence, (c) requires a strict, machine-parsable JSON output with keys `direct_answer`, `reasoning_answer`, `provenance`, and `confidence`, and (d) for counterfactual and action tasks mandates a numeric `counterfactual` object containing `risk_before`, `risk_after` and a direction label consistent with those numbers (Example prompt in Appendix F).

Verifier and Safety Gate. A deterministic assessment module that validates the output, and cited metadata against the Episodic Store and returns a structured diagnostic bundle of issues. The safety

gate is a downstream policy layer that consumes verifier's report and enforces admissibility. It flags answers that fail checks, logs incidents for audit, and routes problematic cases to human review.

Runtime Deployment Flow. In a deployed setting, IndustryAssetEQA operates as a closed-loop operational intelligence service. An operator submits a natural-language query through a maintenance interface (e.g., a dashboard or a Computerized Maintenance Management System (CMMS) frontend), which is first classified by question type. The system then retrieves relevant episodic facts and FMEA knowledge, invokes the risk simulator when counterfactual reasoning is required, and generates a structured, provenance-enforced answer. Before any recommendation is surfaced, a safety and action gate validates evidence consistency and risk thresholds. Approved recommendations (e.g., work-order creation or escalation) are then forwarded to downstream maintenance systems, while low-confidence cases are flagged for human review.

5 Datasets and QA Task Design

Dataset (& Asset)	Episodes	Desc.	Temp.	Diag.	CF.	Act.
Microsoft PdM (Rotating machinery)	5716	5716	5716	5716	761	902
C-MAPSS (Turbofan engines)	4842	4842	4842	4842	-	-
Genesis CPS (Cyber–physical systems)	478	120	478	214	210	23
Hydraulic systems (Hydraulic test rig)	2205	2205	2205	50	2184	2205
Total	13241	12883	13241	10822	3155	3130

Table 1: Datasets, asset statistics and number of QA instances per task category. "Desc." = descriptive, "Temp." = temporal, "Diag." = diagnostic, "CF." = counterfactual, "Act." = action-oriented.

We use four industrial datasets spanning predictive maintenance (Arnab Biswas, 2020), degradation modeling (Saxena et al., 2008; NASA Open Data Portal, 2025), cyber–physical systems (von Birgelen and Niggemann, 2018) and hydraulic systems (Schneider et al., 2018; Helwig et al., 2015a,c). Table 1 includes a summary of them, and the episodes and QAs were validated by SMEs and used as the ground truth for the experiments. Detailed dataset descriptions provided in Appendix G. These ground-truth represent expert-approved interpretations of the provided evidence under a closed-world assumption.

6 Evaluation Methodology

Evaluation Overview. IndustryAssetEQA is evaluated as an operational QA system rather than as a purely linguistic model. Each evaluation instance consists of a single QA query grounded in a specific episodic fact. For each QA instance, we generate one model response under each system configuration (*baselines*: LLM-only, episodic evidence, KG-grounding, provenance-enforced; and the *full IndustryAssetEQA*) (Details in Appendix H.1). We evaluate using black-box API access to `GPT-4o-mini` and `Claude Sonnet 4`. Models are prompted with a structured episode-level fact, optional FMEA-KG context, and a strict JSON output contract, without access to ground-truth data.

Metrics. We evaluate IndustryAssetEQA for *reliability* and *operational validity* in industrial settings, rather than surface-level answer fluency. In safety- and cost-critical scenarios, a QA system must produce structurally valid outputs (*Struct.OK*), ground its answers in verifiable episodic evidence (*Prov.OK*), and generate counterfactual predictions consistent with an explicit risk model (*CF Acc.*). Beyond final-answer correctness, we assess the *semantic faithfulness* of generated explanations. Direct answers are evaluated against SME-validated references using task-appropriate accuracy measures (*Label Cons.*), while descriptive and temporal questions are evaluated via exact and tolerance-based matching. Reasoning quality is evaluated independently of gold explanations by checking entailment to episodic and symbolic evidence (*Entail.Pass*) and verifying extracted claims against the episodic store and FMEA knowledge graph (*Claim Prec.*). Expert evaluation is used for action-oriented questions where no single oracle exists. Metrics are computed per QA instance and then aggregated across all datasets and applicable task types (See Appendix H.2).

7 Experiments and Results

We evaluate IndustryAssetEQA through a set of focused research questions (RQs).

RQ1: Does Embodied Question Answering Improve Answer Reliability? We measure reliability via structural validity, provenance accuracy, diagnostic label consistency, counterfactual direction

Method	GPT-4o-mini						Claude Sonnet 4					
	Struct. OK ↑	Prov. OK ↑	Label Cons. ↑	CF Acc. ↑	Entail. Pass ↑	Claim Prec. ↑	Struct. OK ↑	Prov. OK ↑	Label Cons. ↑	CF Acc. ↑	Entail. Pass ↑	Claim Prec. ↑
LLM-only QA	0.42	0.47	0.62	0.45	0.08	0.12	0.39	0.44	0.59	0.44	0.1	0.12
LLM + Episodic	0.52	0.62	0.71	0.45	0.23	0.25	0.54	0.65	0.73	0.44	0.27	0.28
LLM + Episodic + KG	0.52	0.62	0.73	0.45	0.56	0.51	0.55	0.62	0.73	0.44	0.58	0.54
+ Provenance-Enforced	0.82	0.83	0.89	0.45	0.63	0.59	0.80	0.84	0.87	0.44	0.61	0.60
Full IndustryAssetEQA	**0.88**	**0.89**	**0.94**	**0.88**	**0.72**	**0.67**	**0.90**	**0.89**	**0.95**	**0.91**	**0.78**	**0.74**

Table 2: Per-model results for answer correctness, grounding, and explanation faithfulness. *Struct.OK* = structural validity, *Prov.OK* = evidence grounding, *Label Cons.*= KG-normalized agreement between the predicted diagnostic label and the SME-validated reference, *CF Acc.*= agreement with the expert-reviewed surrogate model on counterfactual direction, *Entail. Pass* = NLI (`FacebookAI/roberta-large-mnli`) ≥ 0.80; and *Claim Prec.* = measure explanation faithfulness against episodic evidence and the KG.

accuracy, and explanation faithfulness(Table 2). Both models follow the same qualitative trajectory. LLM-only baseline performs poorly. Adding episodic evidence improves grounding and explanations, while leaving counterfactual accuracy unchanged, and neurosymbolic fusion with the FMEA-KG raises explanation faithfulness and label consistency. Provenance enforcement produces the largest jump in deployable reliability, and the full IndustryAssetEQA yields the best results (GPT: Struct.OK: 0.88, Prov.OK: 0.89, Label Cons.: 0.94, CF Acc.: 0.88; Claude: Struct.OK: 0.90, Prov.OK: 0.89, Label Cons.: 0.95, CF Acc.: 0.91). Episodic grounding and KG context improve semantic faithfulness, whereas provenance enforcement and simulator integration drive the principal gains required for deployable reliability.

Statistical Significance Analysis. We assess whether differences between model variants are statistically significant using McNemar's test (Laerd Statistics, 2018; MCP Analytics, 2025) for predictions of the Microsoft PdM dataset (Table 3). For descriptive, diagnostic, and counterfactual QA, the differences are statistically significant ($p < 0.05$), while for temporal and action-oriented QA, no statistical significance observed ($p > 0.1$). For action-oriented QA, which is central to decision-making, the two variants exhibit nearly identical behavior ($p = 0.93$). This indicates that, model choice alone is insufficient to address challenges in action-oriented reasoning, underscoring the need for structured, provenance-aware components.

RQ2: Can Models Predict the Effects of Actions on Physical Asset Risk? We evaluate counterfactual reasoning using direction accuracy, which measures whether system outputs are directionally consistent with expert-reviewed surrogate risk model under hypothetical maintenance interventions(Table 2). In an embodied QA framing, this tests whether answers are consistent with modeled action–effect relationships for the asset rather than merely producing plausible-sounding text. Baselines without simulator access perform near chance (≈ 0.45). IndustryAssetEQA achieves a counterfactual direction accuracy of 0.88 (GPT) and 0.91 (Claude), indicating strong agreement with expert-reviewed surrogate model. These results suggest that providing an explicit surrogate intervention model materially improves directional consistency between proposed actions and predicted effects.

Question type	# Questions	p-value	Significance
Descriptive	5716	0.003	Yes
Diagnostic	5716	0.021	Yes
Temporal	5716	0.14	No
Counterfactual	761	0.04	Yes
Action-oriented	902	0.93	No

Table 3: Statistical significance analysis using McNemar's test across QA types on the Microsoft PdM dataset.

Configuration	Entail.Pass ↑	Full Pass ↑	CF Acc. ↑
Full IndustryAssetEQA	**0.72**	**0.89**	**0.88**
w/o Risk simulator	0.59	0.72	0.49
w/o Provenance enforcement	0.42	0.19	0.81
w/o FMEA-KG	0.59	0.35	0.61
w/o Episodic memory	0.27	0.36	0.34

Table 4: Ablation results for `GPT-4o-mini`. Full pass= Avg(Struct.OK+Prov.OK+Label Cons.)

RQ3: Which Architectural Components Enable Embodied, Decision-Grounded QA? We measure each component's contributions via controlled ablations (Table 4, Appendix H.3). Removing the risk simulator primarily breaks counterfactual rea-

soning (CF Acc. 0.88 → 0.49) while only moderately lowering entailment and full-pass rates, showing simulation is essential for reliable what-if judgments. Disabling provenance enforcement causes the largest collapse in deployable reliability (Full Pass 0.89 → 0.19) and strongly reduces explanation faithfulness (Entail.Pass 0.72 → 0.42), indicating that numeric predictions without verifiable grounding are unsafe. Removing episodic memory degrades all metrics (Entail.Pass 0.72 → 0.27; Full Pass 0.89 → 0.36; CF Acc. 0.88 → 0.34), and dropping FMEA-KG lowers semantic alignment and reliability (Entail.Pass 0.72 → 0.59; Full Pass 0.89 → 0.35), so episodic grounding, provenance, KG context, and simulation are complementary and jointly required for deployable embodied QA.

RQ4: How Reliable and Semantically Valid is the FMEA-KG Used for Grounding? We assess the quality of the FMEA-KG along three axes: *(i) expert validation*: the schema and derived QA instances are reviewed by SMEs (Section 5); *(ii) functional validation*: ablations show consistent performance degradation without the KG (e.g., lower entailment and full-pass rates in Tables 2 and 4); and *(iii) consistency checks*: KG-derived outputs are verified against episodic telemetry via provenance constraints (Section 4 and Table 2).

To further quantify quality, we perform triple-level verification using text entailment against ISO-style specifications. Among 1004 candidate triples, 96.0% are judged valid. Support is strongest for semantic relations such as *description* (≈ 76.7%) and *involves* (≈ 71.4%), while structurally weaker relations (e.g., sample/example links, < 10%) exhibit higher noise, indicating areas for refinement. In a 50-triple spot check, 92% of triples achieve high entailment confidence (> 0.6), providing additional evidence of semantic consistency, although some cases remain near the decision boundary. These results indicate that the FMEA-KG reliably captures domain-relevant semantics required for explanation grounding and reasoning, while highlighting limitations in structurally weaker relations that do not directly impact downstream QA performance.

RQ5: Do Expert Evaluations Reveal Failure Modes Not Captured by Automatic Metrics? We conducted a blinded expert evaluation[1] over 22 QA pairs assessed by 10 experts. QA pairs were stratified across four datasets and five task types, with uniform allocation per category. Judges assessed each QA pair on answerability, data grounding (1–5), ambiguity (1–5), scope correctness, and overclaim severity. IndustryAssetEQA outputs were judged answerable in 97% of cases, compared to 46% for the LLM-only baseline (absolute gain = 45%). Mean data grounding scores were significantly higher for IndustryAssetEQA (4.5 ± 0.6) than for LLM-only outputs (3.0 ± 0.9; paired t-test, $p < 0.001$). Severe overclaims occurred in 2% of responses, compared to 28% for LLM-only. Inter-annotator agreement was substantial (Fleiss' $\kappa = 0.63$). These expert judgments corroborate the automatic metrics and demonstrate that provenance enforcement, task-scoped prompting, and simulator alignment materially reduce unsafe and unsupported claims in industrial QA. Figure 4 provides a representative qualitative comparison between LLM-only QA and IndustryAssetEQA on a diagnostic and counterfactual query. The LLM-only response offers a plausible but unsupported explanation without quantitative risk estimates, whereas IndustryAssetEQA cites episode-level sensor data, provides an explicit counterfactual risk shift, and reports a calibrated confidence level.

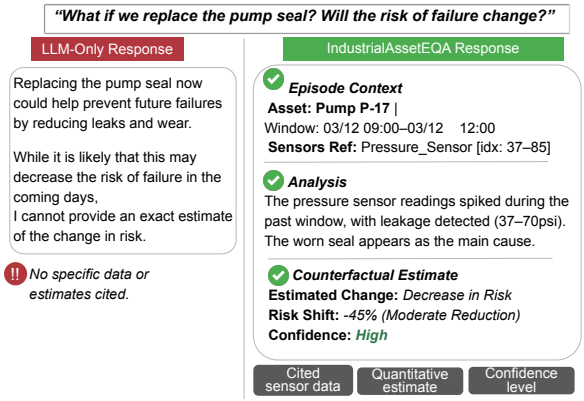

Figure 4: Qualitative comparison on a diagnostic and counterfactual query.

8 Conclusion

IndustryAssetEQA reframes industrial QA as an embodied decision problem and is engineered for live deployment. We are integrating it as a reliability layer into enterprise agents (e.g., SAP Joule Agents), deploying a controlled pilot to measure latency, operator agreement, escalation behavior, reduction in unsupported claims, and extending the integration for multi-asset production deployment.

[1] Survey instrument (sample): https://github.com/IBM/AssetOpsBench/blob/IndustryAssetEQA/IndustryAssetEQA/UserStudy.pdf.

Limitations

We have identified several limitations of our work.

- **Counterfactual modeling:** The counterfactual module is a lightweight parametric intervention estimator rather than a fully identified structural causal model. Its outputs are surrogate risk estimates, not proven causal effects.
- **FMEA-KG quality and coverage:** The KG captures core failure semantics but exhibits variability across relation types, with structurally weaker links being noisier. It is domain-level and may not capture asset-specific variations and rare failure patterns.
- **Fixed episode windowing:** Episodes use fixed windows and may miss long-horizon or multi-scale precursors. Adaptive windowing is left to future work.
- **Evaluation scope and pilot plan:** Evaluation so far is offline on four benchmark datasets. We are preparing a controlled pilot integration to measure operational impact (latency, operator agreement, escalation rate, and incidence of unsupported claims) and to collect data to empirically calibrate counterfactual estimates.
- **System overhead and hosting:** The full pipeline adds engineering and runtime overhead relative to LLM-only systems. The design supports on-premises, hybrid, or Virtual Private Cloud (VPC) deployment and configurable operating points (confidence thresholds, retraining cadence) to manage cost, latency, and privacy trade-offs.

Ethical Considerations

This work targets decision support in safety- and cost-critical industrial maintenance settings, where incorrect or overconfident recommendations could lead to operational risk. To mitigate this, IndustryAssetEQA is designed as an advisory system; it enforces explicit evidence citation, provenance verification, and simulator-grounded counterfactuals, and routes low-confidence or inconsistent outputs to human review rather than automating actions. The system does not learn from personal data, does not involve human subjects beyond expert annotation, and operates on publicly available or industrial benchmark datasets. The deployment requires careful integration with organizational safety policies, clear communication of uncertainty, and human-in-the-loop oversight to prevent misuse or over-reliance on automated recommendations.

Acknowledgments

This work was supported in part by NSF grant #2119654, "RII Track 2 FEC: Enabling Factory to Factory (F2F) Networking for Future Manufacturing". It is also part of a collaborative effort enabled by the open-source project AssetOpsBench (IBM).

References

Arnab Biswas. 2020. Microsoft azure predictive maintenance. https://www.kaggle.com/datasets/arnabbiswas1/microsoft-azure-predictive-maintenance. Accessed: 2026-02-01.

Joyjit Chatterjee and Nina Dethlefs. 2022. Automated question-answering for interactive decision support in operations & maintenance of wind turbines. *IEEE Access*, 10:84710–84737.

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. 2025. Mtbench: A multimodal time series benchmark for temporal reasoning and question answering. *ArXiv*, abs/2503.16858.

Sihao Chen, Fan Zhang, Kazoo Sone, and Dan Roth. 2021. Improving faithfulness in abstractive summarization with contrast candidate generation and selection. In *North American Chapter of the Association for Computational Linguistics*.

Christodoulos Constantinides, Shuxin Lin, and Dhaval C Patel. 2025a. Generalized embedding models for industry 4.0 applications. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2234–2251.

Christodoulos Constantinides, Dhaval C Patel, Shuxin Lin, Claudio Guerrero, SUNIL DAGAJIRAO PATIL, and Jayant Kalagnanam. 2025b. Failuresensoriq: A multi-choice qa dataset for understanding sensor relationships and failure modes. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Pascale Fung, Yoram Bachrach, Asli Celikyilmaz, Kamalika Chaudhuri, Delong Chen, Willy Chung, Emmanuel Dupoux, Hongyu Gong, Hervé Jégou, Alessandro Lazaric, Arjun Majumdar, Andrea Madotto, Franziska Meier, Florian Metze, Louis-Philippe Morency, Théo Moutakanni, Juan Pino, Basile Terver, Joseph Tighe, and 2 others. 2025. Embodied ai agents: Modeling the world. *arXiv preprint arXiv:2506.22355*.

Muhammad Fadhil Ginting, Dong-Ki Kim, Xiangyun Meng, Andrzej Marek Reinke, Bandi Jai Krishna, Navid Kayhani, Oriana Peltzer, David Fan, Amirreza Shaban, Sung-Kyun Kim, and 1 others. Enter the mind palace: Reasoning and planning for long-term

active embodied question answering. In *9th Annual Conference on Robot Learning*.

Nikolai Helwig, Eliseo Pignanelli, and Andreas Schütze. 2015a. Condition monitoring of a complex hydraulic system using multivariate statistics. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, pages 210–215. IEEE.

Nikolai Helwig, Eliseo Pignanelli, and Andreas Schütze. 2015b. Condition monitoring of hydraulic systems. UCI Machine Learning Repository. Accessed: 2026-01-30.

Nikolai Helwig, Eliseo Pignanelli, and Andreas Schütze. 2015c. D8. 1-detecting and compensating sensor faults in a hydraulic condition monitoring system. *Proceedings SENSOR 2015*, pages 641–646.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Yaxuan Kong, Yiyuan Yang, Yoontae Hwang, Wenjie Du, Stefan Zohren, Zhangyang Wang, Ming Jin, and Qingsong Wen. 2025. Time-mqa: Time series multi-task question answering with context enhancement. In *Annual Meeting of the Association for Computational Linguistics*.

Laerd Statistics. 2018. Mcnemar's test using spss statistics. https://statistics.laerd.com/spss-tutorials/mcnemars-test-using-spss-statistics.php. Accessed: 2026-04-20.

Jay Lee, Jaskaran Singh, Moslem Azamfar, and Vibhor Pandhare. 2020. Industrial ai and predictive analytics for smart manufacturing systems. In *Smart manufacturing*, pages 213–244. Elsevier.

Sang-Hyuk Lee, So Won Choi, and Eul-Bum Lee. 2023. A question-answering model based on knowledge graphs for the general provisions of equipment purchase orders for steel plants maintenance. *Electronics*.

Yifan Li, Yuhang Chen, Anh Dao, Lichi Li, Zhongyi Cai, Zhen Tan, Tianlong Chen, and Yu Kong. 2025a. Industryeqa: Pushing the frontiers of embodied question answering in industrial scenarios. In *Proceedings of the Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Yifan Li, Lichi Li, Anh Dao, Xinyu Zhou, Yicheng Qiao, Zheda Mai, Daeun Lee, Zichen Chen, Zhen Tan, Mohit Bansal, and Yu Kong. 2025b. Industrynav: Exploring spatial reasoning of embodied agents in dynamic industrial navigation. *ArXiv*, abs/2511.17384.

Zechuan Li, Hongshan Yu, Yihao Ding, Yan Li, Yong He, and Naveed Akhtar. 2026. Embodied intelligence for 3d understanding: A survey on 3d scene question answering. *Information Fusion*, 126:103624.

Shuxin Lin, Dhaval Patel, and Christodoulos Constantinides. 2025. Fine-tuned thoughts: Leveraging chain-of-thought reasoning for industrial asset health monitoring. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 20687–20700.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent-Pierre Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, and 5 others. 2024. Openeqa: Embodied question answering in the era of foundation models. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16488–16498.

MCP Analytics. 2025. Mcnemar's test: Practical guide for data-driven decisions. https://mcpanalytics.ai/articles/mcnemars-test-practical-guide-for-data-driven-decisions. Published: December 26, 2025; Accessed: 2026-04-20.

Mike A Merrill, Mingtian Tan, Vinayak Gupta, Thomas Hartvigsen, and Tim Althoff. 2024. Language models still struggle to zero-shot reason about time series. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3512–3533.

NASA Open Data Portal. 2025. C-MAPSS Jet Engine Simulated Data. https://data.nasa.gov/dataset/cmapss-jet-engine-simulated-data. Accessed: 2026-02-01.

Dhaval Patel, Shuxin Lin, James Rayfield, Nianjun Zhou, Chathurangi Shyalika Jayakody, Suryanarayana R Yarrabothula, Roman Vaculin, Natalia Martinez, Fearghal O'Donncha, and Jayant Kalagnanam. 2025. Assetopsbench: A real-world evaluation benchmark for ai-driven task automation in industrial asset management. *arXiv preprint arXiv:2506.03828*.

Ricardo Silva Peres, Xiaodong Jia, Jay Lee, Keyi Sun, Armando Walter Colombo, and Jose Barata. 2020. Industrial artificial intelligence in industry 4.0-systematic review, challenges and outlook. *IEEE access*, 8:220121–220139.

RDFLib Developers. 2026. Rdflib: A python library for working with rdf. https://github.com/RDFLib/rdflib. Version 7.6.0, Accessed: 2026-04-20.

Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. 2008. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 international conference on prognostics and health management*, pages 1–9. IEEE.

Patrícia Schmidtová, Eduardo Calò, Simone Balloccu, Dimitra Gkatzia, Rudali Huidrom, Mateusz Lango, Fahime Same, Vilém Zouhar, Saad Mahamood, and Ondřej Dušek. 2025. Do my eyes deceive me? a survey of human evaluations of hallucinations in nlg. In *Proceedings of the 18th International Natural Language Generation Conference*, pages 60–79.

Tizian Schneider, Steffen Klein, and Manuel Bastuck. 2018. Condition Monitoring of Hydraulic Systems Data Set. Accessed: 2026-01-30.

Dammika Seneviratne, Lorenzo Ciani, Marcantonio Catelani, Diego Galar, and 1 others. 2018. Smart maintenance and inspection of linear assets: An industry 4.0 approach. *Acta Imeko*, 7:50–56.

Amit Sheth, Kaushik Roy, and Manas Gaur. 2023. Neurosymbolic artificial intelligence (why, what, and how). *IEEE Intelligent Systems*, 38(3):56–62.

Chen Su, Yuanhe Tian, and Yan Song. 2025. Chain-of-thought reviewing and correction for time series question answering. *ArXiv*, abs/2512.22627.

Sinan Tan, Mengmeng Ge, Di Guo, Huaping Liu, and Fuchun Sun. 2021. Knowledge-based embodied question answering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45:11948–11960.

Vagan Y. Terziyan, Oleksandra Vitko, and Oleksandr Terziyan. 2025. A conceptual design of industrial asset maintenance system by autonomous agents enhanced with chatgpt. *Intelligent and Sustainable Manufacturing*.

Md Nayem Uddin, Amir Saeidi, Divij Handa, Agastya Seth, Tran Cao Son, Eduardo Blanco, Steven Corman, and Chitta Baral. 2024. Unseentimeqa: Time-sensitive question-answering beyond llms' memorization. In *Annual Meeting of the Association for Computational Linguistics*.

Alexander von Birgelen and Oliver Niggemann. 2018. Anomaly detection and localization for cyber-physical production systems with self-organizing maps. In *IMPROVE-Innovative Modelling Approaches for Production Systems to Raise Validatable Efficiency: Intelligent Methods for the Factory of the Future*, pages 55–71. Springer Berlin Heidelberg Berlin, Heidelberg.

Yilin Wang, Peixuan Lei, Jie Song, Yuzhe Hao, Tao Chen, Yuxuan Zhang, Lei Jia, Yuanxiang Li, and Zhongyu Wei. 2025. Itformer: Bridging time series and natural language for multi-modal qa with large-scale multitask dataset. In *42nd International Conference on Machine Learning*, volume abs/2506.20093.

Hong Yung Yip and Amit P. Sheth. 2024. The empwr platform: Data and knowledge-driven processes for the knowledge graph lifecycle. *IEEE Internet Computing*, 28:61–69.

Chaoran Zhang, Chenhao Zhang, Zhaobo Xu, Qinghongbing Xie, Pingfa Feng, and Long Zeng. 2025. Embodied intelligent industrial robotics: Concepts and techniques. *ArXiv*, abs/2505.09305.

Appendix Overview

This appendix details the data processing, system components, evaluation, and reproducibility of IndustryAssetEQA. It is organized as follows:

- **Appendix A: FMEA Knowledge Graph Construction.** Describes how the domain-level FMEA knowledge graph is constructed from ISO-style specifications using the EMPWR platform.
- **Appendix B: Fact Extraction from Time Series Telemetry.** Describes the deterministic time series extractor used to convert raw telemetry, error logs, and maintenance records into episode-level facts. Also includes representation of a sample fact generated.
- **Appendix C: Episodic Store (Ingest and Query Layer).** Details the SQLite-backed episodic store, its schema, ingestion pipeline, query APIs, and how it supports deterministic retrieval and verification.
- **Appendix D: Risk Modeling and Causal Simulator.** Provides implementation details, assumptions, and limitations of the parametric risk model and counterfactual estimator used for "what-if" reasoning, including the confidence heuristic.
- **Appendix E: Example QA Instances.** Presents representative generated QA instances across different QA tasks to illustrate the FMEA-KG enrichment and structured answer format.
- **Appendix F: Prompt Builder and Prompt Templates.** Documents the deterministic prompt construction process, including system and user prompt structure, evidence rendering, task-specific instructions, and JSON output contracts.
- **Appendix G: Datasets Used.** Describes the industrial datasets employed in evaluation (PdM, C-MAPSS, Genesis CPS, and Hydraulic Systems), their characteristics, and how episodic windows and QA instances are derived.
- **Appendix H: Evaluation.** Specifies the evaluation metrics, baselines, ablation design, and automatic verification procedures.

A Appendix: FMEA Knowledge Graph Construction.

The FMEA knowledge graph is constructed from ISO-style failure mode specifications and expert-curated maintenance documentation using the EMPWR platform (Yip and Sheth, 2024). EMPWR provides a structured workflow for ingesting semi-structured documents, defining domain ontologies, validating entity and relation consistency, and managing KG evolution over time. We use EMPWR to define a reusable asset-centric schema covering asset classes, components, failure modes, sensors, and maintenance actions, and to populate this schema with FMEA-aligned metadata.

The FMEA-KG comprises **63 distinct failure modes** (e.g., `AIR` - Abnormal instrument reading, `ELP` - External leakage, `OHE` - Overheating, . . . , `STU` - Stuck) mapped to component- and asset-level concepts. The metadata associates each failure mode with multiple equipment classes and components across **nine asset categories** (*drilling equipment, marine equipment, electrical systems, mechanical systems, rotating machinery, safety and control, subsea equipment, well completion, and well intervention*). The final graph includes **210 entities** (e.g., Subsea pipelines (*assets*), Pressure or Temperature detection (*sensors*), Heating failure (*Failure modes*)) and **1004 relationships** (e.g., *affects*, *component_of*, *indicated_by*, and *mitigated_by*). The KG is *domain-level* rather than dataset-specific. It is constructed independently of any particular dataset and reused across all evaluated datasets. Each dataset activates a subset of the KG corresponding to its asset type (e.g., rotating machinery, hydraulic systems, turbofan engines), sensors, and failure modes. This design supports cross-dataset generalization while ensuring that explanations remain semantically aligned with domain-valid concepts. A snapshot of the FMEA-KG showing the sensor-asset-failure relationships is available at `https://github.com/IBM/AssetOpsBench/blob/IndustryAssetEQA/IndustryAssetEQA/assets/KG.png`

B Appendix: Fact Extraction from Time Series Telemetry

The fact-extraction stage converts raw time-series data into deterministic, time-windowed episode facts suitable for ingestion into the episodic store. We provide a reproducible extractor. Key configurable parameters are the

historical window length (`-window-hours`), the healthy-sampling horizon (`-horizon-hours`), and the maximum healthy episodes per machine (`-max-healthy-per-machine`). The extractor produces one JSON object per episode and writes them to a JSONL file (one JSON per line).

B.1 Fact Extraction Procedure

1. **Failure-centered episodes.** For each failure row (time t) the extractor builds the episode window $(t - \text{window\_hours},\ t]$ and aggregates telemetry, error, and maintenance records within that interval.
2. **Healthy episodes.** For each machine the extractor subsamples telemetry timestamps (at most `max_healthy_per_machine`). A candidate center t is accepted as healthy if there is no failure in $[t,\ t + \text{horizon\_hours}]$, and we construct $(t - \text{window\_hours},\ t]$.
3. **Feature engineering.** For each episode the extractor computes named scalar features:
   - per-sensor aggregates and trend: `{volt,rotate,pressure,vibration}_mean,std,min,max,trend`;
   - error aggregates: `error_count_last_window`, `distinct_error_types_last_window`;
   - maintenance recency: `hours_since_last_maint_<component>` or `hours_since_last_maint_any`;
   - machine static: `machine_age` and one-hot model flags `model_<name>`.

   Missing numeric values are emitted as JSON `null` (or a canonical sentinel such as `-1` for `hours_since_last_maint` when no prior maintenance exists).
4. **Provenance and identifiers.** Each fact includes a deterministic `fact_id`, dataset and source filenames, telemetry time range, the failure row index (when applicable), and a `row_index` for traceability back to raw records.
5. **Output schema.** Facts are written to an output JSONL file. Each fact follows a compact schema (example below).

CLI (example). The extractor was executed in our experiments with the following command (parameters shown):

Listing 1: Extractor CLI (example)

```
python -m src.utils.ts_fact_extractor \
  --telemetry
      ↪ ../msft_azure_pdm/PdM_telemetry.csv
      ↪ \
  --failures
      ↪ ../msft_azure_pdm/PdM_failures.csv \
  --errors ../msft_azure_pdm/PdM_errors.csv \
  --maint ../msft_azure_pdm/PdM_maint.csv \
  --machines
      ↪ ../msft_azure_pdm/PdM_machines.csv \
  --out ../pdm_facts.jsonl \
  --window-hours 24 \
  --horizon-hours 24 \
  --max-healthy-per-machine 50
```

B.2 Episodic Representation

The extraction procedure yields a unified episodic representation. Across all datasets, each maintenance-relevant episode is encoded as a structured fact with a common schema.

- **Asset identifiers**: asset instance, component (when applicable), and dataset source.
- **Episode type**: designation of failure-centered or healthy operation windows.
- **Temporal scope**: start and end timestamps defining the episode window, and failure time when applicable.
- **Engineered features**: summary statistics and trends computed from raw telemetry, along with error and maintenance recency indicators.
- **Observed label**: diagnostic state or failure mode associated with the episode.
- **Domain-aligned contexts**: asset, failure, and sensor profiles queried from FMEA Knowledge graph.
- **Provenance**: references to source files, record indices, and window-level statistics supporting verification and audit.

This explicit episodic representation serves as the single source of truth for question generation, answer verification, and evaluation, enabling temporal grounding, cross-dataset consistency, and reproducible, auditable embodied QA.

Example single output fact (without KG enrichment), and output fact with KG enrichment produced by the extractor in our experiments are as follows:

Listing 2: Representative extracted fact (without KG enrichment)


```
{"fact_id":"pdm_m56_comp3_2015-01-02T03",
 "dataset":"pdm","source_file":
 "PdM_telemetry.csv",
 "asset_id":"machine_56",
 "machineID":56,"episode_type":"failure_window",
 "failure_component":"comp3",
 "failure_time":"2015-01-02 03:00:00",
 "start_time":"2015-01-01
     ↪ 03:00:00","end_time":"2015-01-02
     ↪ 03:00:00",
 "label":"comp3",
 "features":[
   {"name":"volt_mean","value":169.0608},
   {"name":"volt_std","value":17.8556},
   {"name":"volt_min","value":139.2351},
   {"name":"volt_max","value":209.8819},
   {"name":"volt_trend","value":0.2177},
   ...,
   {"name":"hours_since_last_maint_comp3",
   "value":477.0},
   {"name":"machine_age","value":10.0},
   {"name":"model_model1","value":1.0}
 ],
 "provenance":{
   "telemetry_source_file":"PdM_telemetry.csv",
   "telemetry_time_range":["2015-01-01
       ↪ 03:00:00","2015-01-02 03:00:00"],
   "failure_source_file":"PdM_failures.csv",
   "failure_index":0,
   "errors_source_file":"PdM_errors.csv",
   "maint_source_file":"PdM_maint.csv",
   "machines_source_file":"PdM_machines.csv"
 },
 "row_index":0}
```


Listing 3: Example episodic fact that contains engineered features, provenance, and KG / FMEA enrichment ("asset_profile", "failure_profile", "sensor_profiles").


```
{"fact_id":"pdm_m73_comp4_2015-02-16T06",
"dataset":"pdm",
 "source_file":"PdM_telemetry.csv",
 "asset_id":"machine_73","machineID":73,
 "episode_type":"failure_window",
 "failure_component":"comp4",
 "failure_time":"2015-02-16 06:00:00",
 "start_time":"2015-02-15
     ↪ 06:00:00","end_time":"2015-02-16
     ↪ 06:00:00",
 "label":"comp4",
 "features":[
  {"name":"volt_mean","value":169.9217},
  {"name":"volt_std","value":13.1173},
  {"name":"volt_min","value":147.0643},
  {"name":"volt_max","value":193.2928},
  {"name":"volt_trend","value":0.2907},
  {"name":"rotate_mean","value":446.0634},
  {"name":"rotate_std","value":65.0448},
  {"name":"rotate_min","value":305.7116},
  {"name":"rotate_max","value":569.4687},
  {"name":"rotate_trend","value":-1.3637},
  {"name":"pressure_mean","value":99.8202},
  {"name":"pressure_std","value":10.2935},
  {"name":"vibration_mean","value":49.6625},
  {"name":"vibration_std","value":6.3090},
  {"name":"error_count_last_window",
  "value":1.0},
  {"name":"distinct_error_types_last_window",
  "value":1.0},
  {"name":"hours_since_last_maint_comp4",
  "value":5520.0},
  {"name":"machine_age","value":20.0},
  {"name":"model_model2","value":1.0}
 ],
 "asset_profile":{"asset_name":"model2"},
 "failure_profile":{
   "failure_label":"comp4","display_name":"comp4",
   "asset_name":"model2",
   "iso_metadata":{
     "failure_mode":"comp4",
     "name":"Rotor / bearing vibration fault",
     "description":"Mechanical wear,
         ↪ misalignment, or unbalance in
         ↪ the rotor, bearings, or
         ↪ coupling...",
     "equipment_category":"rotating_equipment",
     "associated_sensors":["vibration","rotate"],
     "typical_indicators":{
       "vibration_mean":"significantly above
           ↪ healthy baseline",
       "vibration_max":"high peaks"
     },
     "recommended_actions":[
       "perform vibration analysis and
           ↪ balancing",
       "inspect bearings, lubrication, and
           ↪ alignment",
       "check coupling condition and
           ↪ soft-foot or foundation issues"
     ],
     "severity":"very_high"
   }
 },
 "sensor_profiles":[
   {"sensor_name":"volt","description":"Sensors
       ↪ used to monitor the voltage..."},
   {"sensor_name":"rotate",
   "description":"Specialized sensors used to
       ↪ analyze vibrations..."},
   {"sensor_name":"pressure",
   "description":"Monitors the pressure..."},
   {"sensor_name":"vibration",
   "description":"Vibration sensors monitor
       ↪ machinery..."}
 ],
 "provenance":{
   "telemetry_source_file":"PdM_telemetry.csv",
   "telemetry_time_range":["2015-02-15
       ↪ 06:00:00","2015-02-16 06:00:00"],
   "failure_source_file":"PdM_failures.csv",
   "failure_index":120,
   "maint_events_in_window":1,
   "errors_in_window":1,
   "telemetry_points_in_window":24
 },
 "row_index":120}
```


C Appendix: Episodic Store (Ingest and Query layer)

The episodic store persists extracted episode facts and provides an API for deterministic retrieval, numeric feature queries, and verification used by the

prompt builder and verifier. We implement a compact SQLite-backed store that stores the full JSON fact and an exploded features table for numeric search and ML export.

C.1 Ingestion pipeline (example).

After producing `../pdm_facts.jsonl`, the store was created and populated using the following programmatic call (example shown as a one-liner for reproducibility):

Listing 4: Programmatic ingest into episodic store (example)

```
python -c "from src.utils.episodic_store
    ↪ import EpisodicStore;
s = EpisodicStore('./pdm_episodic_store.db');
print('Ingested',
    ↪ s.ingest_jsonl('../pdm_facts.jsonl'));
print('Assets:', s.list_assets()); s.close()"
```

Example output from our run:

Listing 5: Episodic store ingest output (example)

```
Ingested 5716
Assets:
    ↪ ['machine_1','machine_3',..,'machine_7']
```

C.2 Schema and APIs.

Each ingested episode is stored in a `facts` table and its named features are exploded into a `features` table. Key stored fields and API primitives:

- **Facts:** `fact_id` (PK), `dataset`, `source_file`, `asset_id`, `row_index`, `label`, `start_time`, `end_time`, `fact_json` (full serialized fact), `ingest_time`.
- **Features:** `fact_id`, `feature_name`, `feature_value` (numeric if convertible), `feature_text` (fallback for non-numeric entries).
- **APIs used by the pipeline:**
  - `get_fact(fact_id)`: return the full fact JSON (used by prompt builder and verifier).
  - `get_features(fact_id)`: return a name→value/text mapping for prompt construction and ML export.
  - `query_by_asset(asset_id, limit)` and `query_by_label(label, limit)`: deterministic retrieval used in evaluation sampling and prompt contexts.
  - `search_by_feature_threshold (feature_name, operator` ($\bowtie \in \{<, \leq, =, \geq, >\}$, `value)`: numeric threshold search (e.g., `vibration_max > 60`) for evidence selection.
  - `query_by_time_range(asset_id, start, end)`: retrieve episodes overlapping a given time window (used by temporal QA and provenance checks).
  - `export_features_csv(out_csv)`: helper to materialize ML-ready CSVs.

C.3 Operational notes and reproducibility.

- Ingestion is idempotent (upsert semantics) and records an `ingest_time` for traceability. Facts with the same `fact_id` are replaced if `overwrite=True` is passed.
- Numeric conversions are attempted for feature values; non-numeric text is preserved in `feature_text` so provenance and human inspection remain possible.
- The lightweight SQLite store is sufficient for experimental evaluation and reproducibility; it supports deterministic retrievals used in prompt construction and in automatic verification. For high-throughput production deployments a scalable store (e.g., a managed DB or vector store with feature indexing) would be recommended.

C.4 Traceability example.

The prompt builder and verifier rely on entries such as `telemetry_time_range` and `failure_index` to show exactly which records support a claim. For example, given `fact_id = "pdm_m56_comp3_2015-01-02T03"`, the QA builder can:

1. call `get_fact(fact_id)` to obtain the episode-level fact and its provenance;
2. expose only the minimal fields required by a question (e.g., features and provenance) in the constructed prompt;
3. after the LLM produces a response, the verifier can confirm that all cited `fact_id`s, sensor windows, and referenced features exist in the store.

D Appendix: Risk Modeling and Causal Simulator

The risk modeling and causal simulator provides a local, parametric counterfactual estimate of intervention effects; it is not a structural causal model and does not guarantee identifiability. The 'do' is

implemented by feature replacement (it ignores latent confounders and assumes the learned $P(y \mid \boldsymbol{x})$ remains valid under the intervention). Unknown intervention features are silently ignored, and the approach is parametric and model-dependent.

A simple confidence heuristic is used by the estimator:

$$c \;=\; 0.5 + 0.5\,\min\!\Big(1,\; \frac{|r_{\text{before}} - 0.5|}{0.5}\Big),$$

which increases when the baseline risk is more extreme (i.e., moves away from 0.5).

E Appendix: Example QA Instances

This section provides representative JSON examples of generated QA instances (diagnostic, descriptive, temporal, counterfactual, action-oriented). Each QA shows how FMEA-KG fields (when available in the fact) are surfaced in the question, answer, reasoning, and provenance.

E.0.1 Diagnostic (with KG excerpt)

Listing 6: Diagnostic QA example


```
{"qa_id":"pdm_diag_pdm_m56_comp3_2015-01-02T03",
 "fact_id":"pdm_m56_comp3_2015-01-02T03",
 "task_type":"diagnostic",
 "question":"Why is this diagnostic episode
     ↪ for asset machine_56 labeled 'comp3'
     ↪ (comp3) over the time window
     ↪ 2015-01-01 03:00:00 to 2015-01-02
     ↪ 03:00:00?",
 "direct_answer":"This episode is labeled
     ↪ 'comp3' (comp3) because the observed
     ↪ features match the typical
     ↪ indicators associated with this
     ↪ failure mode.",
 "reasoning_answer":"This episode is labeled
     ↪ 'comp3' (comp3) because, key
     ↪ observed features:
     ↪ volt_mean=169.061, volt_std=17.856,
     ↪ volt_min=139.235. These features
     ↪ deviate from normal thresholds and
     ↪ sensor patterns and belong to the
     ↪ failure modes of a compressor",
 "provenance":{
   "fact_id":"pdm_m56_comp3_2015-01-02T03",
   "features":["volt_mean","volt_std",
   "volt_min"],
   "file":"PdM_telemetry.csv","row":0,
   "asset_profile":{"asset_name":"model1",
   "equipment_category":"rotating_equipment"},
   "failure_profile_id":156,
   "telemetry_points_in_window":22},
 "label":"comp3","asset_id":"machine_56",
 "asset_profile_brief":{"asset_name":"model1",
 "equipment_category":"rotating_equipment"},
 "failure_profile_brief":{
   "failure_mode":"Rotor / bearing vibration
       ↪ fault","display_name":"comp3",
   "severity":"very_high",
   "associated_sensors":["vibration"],
   "recommended_actions":["perform vibration
       ↪ analysis and balancing", "..."]}}
```


E.0.2 Descriptive (sensor description surfacing)

Listing 7: Descriptive QA example


```
{"qa_id":"pdm_desc_pdm_m56_comp3_2015-01-02T03",
 "fact_id":"pdm_m56_comp3_2015-01-02T03",
 "task_type":"descriptive",
 "question":"During the time window
     ↪ 2015-01-01 03:00:00 to 2015-01-02
     ↪ 03:00:00 for asset machine_56, what
     ↪ was the average vibration level
     ↪ (Vibration sensors monitor machinery
     ↪ for abnormal vibrations, which can
     ↪ indicate issues like misalignment,
     ↪ imbalance, or wear.)?",
 "direct_answer":"The average vibration level
     ↪ was approximately 39.26.",
 "reasoning_answer":"In this episode for
     ↪ asset machine_56, the feature
     ↪ vibration_mean is 39.26, computed
     ↪ over 22 telemetry points in the
     ↪ window.",
 "provenance":{"fact_id"
 :"pdm_m56_comp3_2015-01-02T03",
 "features":["vibration_mean"],
 "file":"PdM_telemetry.csv","row":0,
 "telemetry_points_in_window":22,
 "sensor_description":"Vibration sensors
     ↪ monitor machinery for abnormal
     ↪ vibrations..."},
 "label":"39.2568","asset_id":"machine_56"}
```


E.0.3 Temporal / Counting

Listing 8: Temporal / counting QA example


```
{"qa_id":"pdm_temp_pdm_m56_comp3_2015-01-02T03",
 "fact_id":"pdm_m56_comp3_2015-01-02T03",
 "task_type":"temporal_count",
 "question":"Between 2015-01-01 03:00:00 and
     ↪ 2015-01-02 03:00:00 for asset
     ↪ machine_56, how many distinct error
     ↪ types occurred?",
 "direct_answer":"There were 2 distinct error
     ↪ types in this window.",
 "reasoning_answer":"In this episode the
     ↪ feature
     ↪ distinct_error_types_last_window is
     ↪ 2, and the total error count is 2.",
 "provenance":{"fact_id":
 "pdm_m56_comp3_2015-01-02T03",
 "features":["error_count_last_window",
 "distinct_error_types_last_window"],
 "file":"PdM_errors.csv","row":0},
 "label":"0","asset_id":"machine_56"}
```


E.0.4 Counterfactual (with KG fields in provenance)

Listing 9: Counterfactual QA example


```
{"qa_id":"pdm_ts_cf_pdm_m56_comp3_2015-01-02T03",
 "fact_id":"pdm_m56_comp3_2015-01-02T03",
```

```
"task_type":"counterfactual",
"question":"For asset machine_56 (failure mode: comp3) in the window 2015-01-01 03:00:00 to 2015-01-02 03:00:00, if maintenance targeting comp3 had been performed immediately before the window (resetting hours_since_last_maint_comp3 from 477.0 to 0), how would the risk of failure change?",
"direct_answer":"The risk of failure would decrease.",
"reasoning_answer":"Baseline P(failure) $\approx$ 1.000; post-intervention P(failure) $\approx$ 0.000 (Δ = -1.000).",
"provenance":{
  "fact_id":"pdm_m56_comp3_2015-01-02T03",
  "features":["hours_since_last_maint_comp3"],
  "file":"PdM_telemetry.csv","row":0,
  "telemetry_points_in_window":22,
  "asset_profile_brief":{"unit_subunit":

  ["rotor","bearings","coupling"],
  "asset_name":"model1"},
  "failure_profile_brief":{"failure_mode"
  :"comp3",
  "recommended_actions":["perform vibration analysis and balancing", "..."],"severity":"very_high"},
  "sensor_profiles_brief":
  [{"sensor_name":"vibration",
  "description":"..."}]
},
"counterfactual":{
  "intervention":"do
  (hours_since_last_maint_comp3 = 0.0)","risk_before":1.0,
  "risk_after":9.256e-06,"delta_risk":-0.99999,
  "direction":"decrease",
  "probs_before":{...},"probs_after":{...}
},
"confidence":1.0,"label":"comp3",
"asset_id":"machine_56"}
```


E.0.5 Action recommendation (with KG fields in provenance)

Listing 10: Action-recommendation QA example


```
{"qa_id": "pdm_ts_action_"
"pdm_m56_comp3_2015-01-02T03",
"fact_id": "pdm_m56_comp3_2015-01-02T03",
"task_type": "action_recommendation",
"question": "For rotating_equipment machine_56 in the time window 2015-01-01 03:00:00 to 2015-01-02 03:00:00, should a maintenance work order be opened now, or is it acceptable to continue monitoring?",
"direct_answer": "A maintenance work order should be opened now.",
"reasoning_answer": "The learned risk model estimates probability of any failure $\approx$ 1.00 (threshold=0.50). Failure severity: high. Recommended diagnostics/actions: check discharge and downstream valves for incorrect positions or sticking; inspect filters, strainers, and lines for blockage; review process conditions and control setpoints. Most informative sensors: pressure. Equipment class/type: electric_motor_driven_rotating_machine.",
"provenance": {
"fact_id": "pdm_m56_comp3_2015-01-02T03",
"features": ["volt_mean", "volt_std", "volt_min", "volt_max", "volt_trend", "rotate_mean", "rotate_std", "rotate_min", "rotate_max", "rotate_trend", "pressure_mean", "pressure_std", "pressure_min", "pressure_max", "pressure_trend", "vibration_mean", "vibration_std", "vibration_min", "vibration_max", "vibration_trend", "error_count_last_window", "distinct_error_types_last_window", "hours_since_last_maint_comp3", "machine_age", "model_model1"],
"file": "PdM_telemetry.csv",
"row": 0,
"telemetry_points_in_window": 22,
"errors_in_window": 0,
"failure_profile_id": "comp3",
"asset_profile_brief": {
"equipment_category": "rotating_equipment",
"equipment_class_type": "electric_motor_driven_rotating_machine",
"unit_subunit": ["process_path", "discharge_line", "valves"],
"asset_name": "model1"
},
"failure_profile_brief": {
"failure_mode": "comp3",
"display_name": "comp3",
"short_description": "Restriction, valve malfunction, or process-side blockage causing sustained high discharge pressure while rotational speed and vibration remain near normal.",
"associated_sensors": ["pressure"],
"typical_indicators": {
"pressure_mean": "significantly above healthy baseline",
"pressure_trend": "often positive or sustained high",
"rotate_mean": "near nominal",
"vibration_mean": "near nominal or slightly increased"
},
"recommended_actions": ["check discharge and downstream valves for incorrect positions or sticking",
"inspect filters, strainers, and lines for blockage",
"review process conditions and control setpoints"],
"severity": "high"
}
},
"label": "open_work_order",
"asset_id": "machine_56",
"confidence_estimator": 1.0,
"risk": 1.0,
"probs_before": {
"comp1": 5.666517784187859e-26,
"comp2": 1.6855443558067901e-53,
```

```
"comp3": 1.0,
"comp4": 4.746799049527151e-27,
"healthy": 1.1467891488958534e-43
},
"asset_profile_brief": {
"equipment_category": "rotating_equipment",
"equipment_class_type":
    ↪ "electric_motor_driven_rotating_machine",
"unit_subunit": ["process_path",
    ↪ "discharge_line", "valves"],
"asset_name": "model1"
},
"failure_profile_brief": {
"failure_mode": "comp3",
"display_name": "comp3",
"short_description": "Restriction, valve
    ↪ malfunction, or process-side blockage
    ↪ causing sustained high discharge
    ↪ pressure while rotational speed and
    ↪ vibration remain near normal.",
"associated_sensors": ["pressure"],
"typical_indicators": {
"pressure_mean": "significantly above healthy
    ↪ baseline",
"pressure_trend": "often positive or
    ↪ sustained high",
"rotate_mean": "near nominal",
"vibration_mean": "near nominal or slightly
    ↪ increased"
},
"recommended_actions": ["check discharge and
    ↪ downstream valves for incorrect
    ↪ positions or sticking",
"inspect filters, strainers, and lines for
    ↪ blockage",
"review process conditions and control
    ↪ setpoints"],
"severity": "high"
}}
```

F Appendix: Prompt Builder and Prompt Templates.

Below is a compact version of the template used in our experiments (user prompt shown; the system prompt contains the role and the output contract):

Listing 11: Example prompt generated

```
TASK TYPE:Diagnostic

TASK DESCRIPTION:
You must answer diagnostic questions
    ↪ explaining why the episode has the
    ↪ given label.
Use only the evidence; do not invent features
    ↪ or events.

EVIDENCE:
fact_id: pdm_m56_comp3_2015-01-02T03
asset_id: machine_56
window_start: 2015-01-01 03:00:00
window_end: 2015-01-02 03:00:00
diagnostic_features:
  - vibration_mean: 39.26
  - pressure_mean: 124.67
  - hours_since_last_maint_comp3: 477.0
asset_profile:
  asset_name: model1
failure_profile:
  name: High discharge pressure / process
      ↪ restriction
  associated_sensors: [pressure]

QUESTION:
Why is this episode labeled as failure for
    ↪ asset 'compressor 3'?

RESPONSE FORMAT:
Return ONLY a single JSON object with keys:
  direct_answer, reasoning_answer,
      ↪ provenance, confidence
```

Listing 12: Prompt Builder CLI (example)

```
python -m src.utils.prompt_builder_static \
  --db ../pdm_episodic_store.db \
  --qa ../pdm_qa_diag.jsonl \
  --qa-id pdm_diag_pdm_m56_comp3_2015-01-02T03
```

G Appendix: Datasets Used

Microsoft Azure Predictive Maintenance Dataset (PdM). The *Microsoft Azure Predictive Maintenance dataset*(Arnab Biswas, 2020) is a multi-table corpus designed for developing and evaluating ML models for predictive maintenance. It contains:

- Hourly telemetry (voltage, rotation, pressure, vibration) for 100 machines: `PdM_telemetry.csv`.
- Failure log (component replacements after breakdowns): `PdM_failures.csv`.
- Non-fatal error events (do not cause shutdown): `PdM_errors.csv`.
- Maintenance records (scheduled pro-active and unscheduled reactive interventions): `PdM_maint.csv`.
- Machine metadata (model, age, etc.): `PdM_machines.csv`.

Times in all tables are rounded to the nearest hour to align with the telemetry sampling; failures are a subset of maintenance events (component replacements following a breakdown). The dataset therefore supports episode construction that pairs fixed historical telemetry windows with downstream outcomes (failure vs. healthy) and contextual covariates (error counts, hours-since-last-maintenance, machine model/age), making it well suited for supervised risk modelling, time series feature engineering, and counterfactual or what-if analyses in predictive-maintenance research.

Turbofan Engine Degradation Dataset (C-MAPSS). We use the *NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) turbofan engine dataset* (Saxena et al., 2008; NASA Open Data Portal, 2025), a widely adopted benchmark for prognostics and health management. The dataset consists of multivariate run-to-failure time series collected from a fleet of simulated turbofan engines, where each trajectory corresponds to a distinct engine with unknown initial wear and manufacturing variability. Engines operate under varying operational settings and are monitored via multiple sensor channels corrupted by realistic noise. Faults emerge during operation and progressively grow until system failure in the training sets, while test trajectories terminate prior to failure with ground-truth Remaining Useful Life (RUL) values provided.

We use the training and test splits of all four standard subsets: FD001 (100/100 trajectories, single operating condition, single fault mode: high-pressure compressor degradation), FD002 (260/259 trajectories, six operating conditions, single fault mode), FD003 (100/100 trajectories, single operating condition, two fault modes: high-pressure compressor and fan degradation), and FD004 (248/249 trajectories, six operating conditions, two fault modes). Each time step represents one operational cycle and includes three operational settings and 21 sensor measurements, yielding a total of 26 variables per cycle. For IndustryAssetEQA, raw sequences are segmented into fixed-length episodic windows aligned with degradation progression, enabling descriptive, temporal, diagnostic, counterfactual, and action-oriented question answering over realistic degradation dynamics.

Cyber-Physical Production Systems (Genesis). *Genesis cyber-physical production system* (von Birgelen and Niggemann, 2018) captures sensor-level behavior of an industrial demonstrator that autonomously sorts two materials (wood and metal) into designated storage locations. The system consists of four modular stations—storage magazine, sensor module, wood storage, and metal storage—whose physical positions can be permuted, with the PLC control logic automatically adapting to the configuration. Material transport is carried out by a linear drive equipped with a pneumatic gripper, executing a cyclic procedure that includes homing, material ejection, gripping, material-type sensing, transport to the appropriate storage box, and release. The dataset comprises two complementary subsets.

The first contains fully labeled multivariate time series data (`Genesis_StateMachineLabels.csv`, `Genesis_AnomalyLabels.csv`) with 16,220 observations sampled every $50\,\mathrm{ms}$ via an OPC DA server. These files share identical sensor readings but differ in labeling: one annotates the internal PLC state machine (idle, homing, transport, gripping, sensing, and release states), while the other provides manually verified anomaly labels corresponding to linear drive malfunctions (jammed/tilted motion and subsequent correction). The second subset consists of unlabeled runs (`Genesis_normal.csv`, `Genesis_lineardrive.csv`, `Genesis_pressure.csv`) capturing nominal operation and gradually induced faults in the linear drive and pneumatic pressure. For IndustryAssetEQA, we derive short episodic windows from these time series to evaluate descriptive, temporal, and diagnostic question answering under realistic cyber-physical noise and partial supervision, complementing the more failure-centric PdM and C-MAPSS datasets.

Hydraulic Systems Dataset. We additionally evaluate IndustryAssetEQA on the *Condition Monitoring of Hydraulic Systems* dataset (Schneider et al., 2018; Helwig et al., 2015a,c,b), which captures multivariate sensor measurements from a laboratory hydraulic test rig under controlled degradation. The rig consists of a primary working circuit and a secondary cooling–filtration circuit connected via a shared oil tank, and repeatedly executes constant-load operating cycles of 60 seconds. Each cycle is annotated with condition states of four hydraulic components—cooler, valve, pump, and accumulator—across multiple severity levels.

The dataset contains 2,205 operating cycles with no missing values. Each cycle is represented as a matrix of raw time series measurements collected at heterogeneous sampling rates, yielding 43,680 attributes per instance. Sensors include pressures (PS1–PS6, 100 Hz), motor power (EPS1, 100 Hz), volume flow (FS1–FS2, 10 Hz), temperatures (TS1–TS4, 1 Hz), vibration (VS1, 1 Hz), and virtual efficiency indicators (CE, CP, SE, 1 Hz). Component condition labels are provided cycle-wise in a separate profile file and include both categorical degradation states (e.g., leakage levels) and continuous-valued health indicators (e.g., accumu-

lator pressure).

Following prior work, we treat each operating cycle as a maintenance-relevant episode and aggregate sensor traces into fixed-length episodic representations suitable for descriptive, diagnostic, temporal, and counterfactual question answering. This dataset is widely used for benchmarking condition monitoring and fault diagnosis methods and provides realistic, multi-rate telemetry for evaluating evidence-grounded industrial QA.

H Appendix: Evaluation

H.1 Baselines

We compare IndustryAssetEQA against several baselines which we term as system configurations that reflect common design choices in industrial QA systems. While several baseline configurations correspond to partial versions of IndustryAssetEQA, we treat baselines as externally reasonable system designs for comparison, whereas ablations explicitly remove components from the full IndustryAssetEQA system to isolate causal contributions.

LLM-only QA. The LLM is prompted with the question and a natural-language summary of the episode, without structured evidence, provenance constraints, and simulator outputs.

LLM with Episodic Evidence. The model receives a textual rendering of episode features and time windows, but is not required to produce structured outputs or explicit provenance.

LLM with Episodic Evidence + KG Grounding. In addition to episodic facts, the model is provided with structured domain knowledge derived from FMEA-KG. This includes failure mode metadata, equipment categories and classes, unit and subunit types, sensor–failure associations, and recommended maintenance actions. Outputs remain unconstrained free text, and no verification is applied. This variant isolates the effect of symbolic domain knowledge grounding independent of structural and verifier-based constraints.

Provenance-Enforced QA (no simulator). The model is required to return structured JSON outputs containing an explicit `provenance` field that cites episode identifiers and feature names. A deterministic verifier validates these references against the Episodic Store and rejects any output with missing, malformed, or unsupported evidence. This configuration enforces evidence correctness post-generation but does not provide access to the risk simulator for counterfactual reasoning.

Full IndustryAssetEQA. The complete system integrates episodic grounding, KG-based domain knowledge, provenance-enforced structured outputs, and simulator-grounded counterfactual reasoning. This configuration reflects the intended deployable system.

H.2 Evaluation Metrics

We define the following evaluation metrics, all of which are computed automatically using deterministic verifiers over structured model outputs.

Structural Validity (`Struct.OK`). This metric checks whether the model output conforms to the required JSON schema, including the presence and type correctness of mandatory fields such as `direct_answer`, `reasoning_answer`, `provenance`, and `confidence`. Outputs that fail structural validation are considered unusable in deployed systems and are counted as failures.

Provenance Accuracy (`Prov.OK`). Provenance accuracy verifies that the response cites a valid episodic fact identifier, references only features that actually exist in that episode, and matches the recorded source file and row index when provided. This check is implemented by validating model outputs against the Episodic Store and penalizes hallucinated or unsupported evidence.

`Struct.OK` and `Prov.OK` are implemented by the `Verifier module` that validates model outputs against the episodic store.

Label Consistency (`Label Cons.`). For diagnostic QA tasks, `label_consistent` measures agreement between the model's predicted diagnostic label and the SME-validated reference label. To account for terminology variation, labels are normalized using the FMEA knowledge graph, allowing matches via synonyms or hierarchical relations. This metric evaluates final-answer correctness only and does not assess the correctness or faithfulness of the generated explanation. Label consistency is not applicable to non-diagnostic tasks.

Full Pass Rate (`Full_pass`). The full pass rate measures the fraction of QA instances for which all applicable checks succeed simultaneously: structural validity, provenance accuracy, and label consistency. This metric serves as a conservative proxy

for *deployable answer reliability*, as any failure indicates an answer that cannot be safely consumed by downstream systems.

Temporal and Counting Accuracy. For temporal and counting QA tasks, we evaluate whether predicted timestamps, durations, or counts match the episode-derived ground truth. Depending on the task, this is computed as exact match or window-level overlap.

Counterfactual Direction Accuracy (`CF Acc.`). For counterfactual QA tasks, we evaluate whether the predicted direction of risk change (*increase*, *decrease*, or *no change*) matches the direction produced by the parametric risk simulator, which we use as the surrogate intervention model. The simulator's directional predictions were reviewed by domain experts on sampled cases for directional plausibility. We therefore measure agreement with the expert-reviewed surrogate model rather than claiming causal ground truth under real-world interventions. When numeric risk estimates (`risk_before` and `risk_after`) are present in the model output, the direction is computed directly from these values; textual direction labels are used only as a fallback. This metric evaluates action–model consistency under an explicit parametric intervention assumption rather than empirical causal validity.

Entailment Pass Rate (`Entail.Pass`). This metric measures whether generated explanations are supported by the provided evidence. Each reasoning answer is segmented into individual sentences, and a natural language inference (NLI) model, `FacebookAI /roberta-large-mnli` (Liu et al., 2019) is used to test whether each sentence is entailed by the episodic facts and FMEA knowledge supplied in the prompt. A sentence is counted as entailed if its entailment probability exceeds a fixed threshold. We fix the threshold at 0.80 following prior empirical evaluations. The entailment pass rate is computed as the fraction of reasoning sentences that satisfy this criterion, capturing the degree to which explanations avoid unsupported or contradictory claims.

Claim Precision (`Claim Prec.`). Claim precision evaluates the factual correctness of explanations at a symbolic level. Atomic claims are extracted from the reasoning text (e.g., assertions about feature states or failure–symptom relationships) and verified deterministically against the episodic store and the FMEA knowledge graph. Claim precision is defined as the fraction of extracted claims that can be validated. This metric penalizes explanations that introduce incorrect or unverifiable statements, even when the final answer is correct.

H.3 Ablation Design

To understand which components materially affect model behavior, we perform controlled ablations:

- **No episodic memory**: remove the episodic store and explicit episode identifiers, providing only free-form summaries of telemetry without fact-level indexing or verifiable provenance.
- **No FMEA knowledge graph**: remove KG-derived semantic context (failure-mode metadata, associated sensors, and recommended actions), leaving the model to rely solely on episodic telemetry and prompts.
- **No provenance enforcement**: remove the structured output requirement and verifier checks, allowing free-text explanations without machine-verifiable provenance.
- **No risk simulator**: require the model to answer counterfactual questions without simulator guidance.

All models are evaluated on the same QA instances to isolate the effect of architectural constraints rather than dataset differences.