Article

Not peer-reviewed version

Hybrid AI and LLM-Enabled Agent-Based Real-Time Decision Support Architecture for Industrial Batch Processes: A Clean-in-Place Case Study

Apolinar González-Potes *, Diego Martínez-Castro, Carlos M. Paredes, Alberto Ochoa-Brust, Luis J. Mena, Rafael Martínez-Peláez, Vanessa G. Félix, Ramón A. Félix

Posted Date: 23 December 2025

doi: 10.20944/preprints202512.2023.v1

Keywords: clean-in-place; large language models; multi-agent systems; industrial IoT; process supervision; explainable AI; real-time decision support; predictive maintenance; industry 4.0

 doi:10.20944/preprints202512.2023.v1

Article

Hybrid AI and LLM-Enabled Agent-Based Real-Time Decision Support Architecture for Industrial Batch Processes: A Clean-in-Place Case Study

Apolinar González-Potes [1,*], Diego Martínez-Castro [2], Carlos M. Paredes [3], Alberto Ochoa-Brust [1], Luis J. Mena [3], Rafael Martínez-Peláez [4,5], Vanessa G. Félix [3] and Ramón Felix-Cuadras [1]

[1] Facultad de Ingeniería Mecánica y Eléctrica, Universidad de Colima, Colima, Mexico
[2] Facultad de Ingeniería y Ciencias básicas, Universidad Autónoma de Occidente, Cali, Colombia
[3] Facultad de Ingeniería,LIDIS, Universidad de San Buenaventura, Cali, Colombia
[4] Unidad Académica de Computación, Universidad Politécnica de Sinaloa, Mazatlán, Mexico
[5] Departamento de Ingeniería de Sistemas y Computación, Universidad Católica del Norte, Antofagasta, Chile
* Correspondence: apogon@ucol.mx; Tel.: +52-312-107-1563

Abstract

A hybrid AI and LLM-enabled architecture is presented for real-time decision support in industrial batch processes, where supervision still relies heavily on human operators and ad-hoc SCADA logic. The framework combines deterministic rule-based agents, fuzzy and statistical enrichment, and large language models (LLMs) to support monitoring, diagnostic interpretation, preventive maintenance planning, and operator interaction with minimal manual intervention. High-frequency sensor streams are collected into rolling buffers per active process instance; deterministic agents compute enriched variables, discrete supervisory states, and rule-based alarms, while an LLM-driven analytics agent answers free-form operator queries over the same enriched datasets through a conversational interface. The architecture is instantiated and deployed in the Clean-in-Place (CIP) system of an industrial beverage plant and evaluated following a case-study design aimed at demonstrating architectural feasibility and diagnostic behavior under realistic operating regimes rather than statistical generalization. Three representative multi-stage CIP executions—purposively selected from 24 runs monitored during a six-month deployment—span nominal baseline, preventive-warning, and diagnostic-alert conditions. The study quantifies stage-specification compliance, state-to-specification consistency, and temporal stability of supervisory states, and performs spot-check audits of numerical consistency between language-based summaries and enriched logs. Results show high time within specification in sanitising stages (100% compliance across the evaluated runs), coherent and mostly stable supervisory states in variable alkaline conditions (state-specification consistency $\Gamma_s \geq 0.98$), and data-grounded conversational diagnostics in real time (median numerical error below 3% in audited samples), without altering the existing CIP control logic. These findings suggest that the architecture can be transferred to other industrial cleaning and batch operations by reconfiguring process-specific rules and ontologies, offering a practical path toward AI-assisted process supervision with explainable conversational interfaces that support preventive maintenance decision-making and equipment health monitoring.

Keywords: clean-in-place; large language models; multi-agent systems; industrial IoT; process supervision; explainable AI; real-time decision support; predictive maintenance; industry 4.0

1. Introduction

Industrial batch processes in food and beverage, pharmaceuticals, and wastewater treatment depend critically on cleaning-in-place (CIP) operations and similar multi-stage procedures to guarantee hygiene, product quality, and regulatory compliance [1,2]. These cycles typically execute multi-stage

 doi:10.20944/preprints202512.2023.v1

programs—for example, pre-rinse, alkaline wash, intermediate rinse, acid wash, sanitising and final rinse— under tight constraints on temperature, flow, conductivity and contact time [2,3]. In current practice, such sequences are orchestrated by programmable logic controllers (PLCs) and supervised through SCADA systems, where operators monitor trends, acknowledge alarms and manually interpret complex process conditions. While this architecture is robust and widely adopted, it offers limited support for proactive decision making, root-cause analysis or flexible what-if exploration over historical and real-time data—limitations that become increasingly problematic as plants grow in complexity and product portfolios diversify [3,4].

To address the need for flexibility and scalability, recent Industry 4.0 frameworks advocate modular, component-based and microservice-oriented architectures for industrial automation [5,6]. These approaches promote loose coupling between control, monitoring, data management and higher-level applications, enabling incremental deployment and technology heterogeneity. Previous work on component-based microservices for bioprocess automation has shown that containerised services and publish/subscribe communication can decouple control and supervision across heterogeneous equipment, including bioreactors and CIP systems, while meeting industrial real-time and robustness requirements [6]. Complementary contributions in bioprocess monitoring and control have proposed advanced observers, model predictive controllers and fault-detection schemes, but typically focus on algorithmic performance rather than on how human operators interact with increasingly complex automation stacks in day-to-day operation [1].

In parallel, large language models (LLMs) and conversational agents have emerged as powerful tools for making complex data and models more accessible to domain experts, enabling natural-language querying, explanation and guidance [7,8]. Early studies in industrial settings indicate that LLM-based assistants can help operators explore process histories, retrieve relevant documentation and reason about abnormal situations using free-form queries [8]. However, integrating LLMs into safety-critical environments remains challenging: LLM outputs are non-deterministic, may hallucinate and must coexist with hard safety constraints, deterministic interlocks and real-time requirements [7,8]. Existing architectures rarely combine deterministic rule-based supervision, continuous analytics and LLM-based conversational interfaces in a way that preserves safety while providing meaningful assistance for CIP operations and other multi-stage cleaning or batch processes.

1.1. From Reactive Alarms to Diagnostic Intelligence

Modern food and beverage facilities equipped with advanced control strategies— including model-based flow optimisation and automated parameter regulation— have achieved significant process stability. In such environments, catastrophic process failures are rare, and traditional alarm systems designed to detect imminent faults often generate excessive nuisance alerts that operators learn to ignore or dismiss [9]. The supervisory challenge has consequently evolved from *detecting failures* to *interpreting operational signals*: distinguishing between acceptable process variability and subtle patterns indicating emerging maintenance needs before they impact production [10,11].

Consider a CIP execution where flow rates are 10% below optimal but still within regulatory acceptance bounds. A traditional threshold-based alarm system remains silent, yet this pattern—if consistent across multiple cycles—may signal gradual pump wear requiring scheduled maintenance. Similarly, slight temperature deviations that do not compromise product safety may indicate boiler efficiency degradation, and conductivity variations within specification may reflect dosing pump calibration drift. In all cases, individual executions complete successfully by specification, but **trend analysis across execution cycles** reveals actionable maintenance opportunities that can prevent unplanned downtime and extend equipment lifespan [11].

Extracting these diagnostic insights currently depends on expert operator knowledge and time-intensive manual log analysis—a process that scales poorly as facilities expand and product portfolios diversify. Moreover, the high dimensionality of sensor data (dozens of variables sampled at sub-second intervals) makes it difficult for operators to recognise subtle patterns amid normal process variability [9].

 doi:10.20944/preprints202512.2023.v1

In well-optimised plants where CIP executions routinely meet specifications, failures are rare but emerging equipment degradation must be identified through longitudinal trend analysis rather than catastrophic fault detection. A system that monitors 100 consecutive successful CIP runs provides no evidence of diagnostic capability; conversely, analysing executions that span the operational spectrum—from nominal baseline to preventive warning patterns to diagnostic alert regimes—demonstrates the architecture's ability to distinguish acceptable process variability from systematic equipment drift requiring maintenance intervention before operational impact occurs.

1.2. Proposed Architecture and Evaluation Approach

This paper proposes a generic, process-agnostic multi-agent architecture for AI-assisted monitoring and decision support in industrial batch processes. The architecture is instantiated and evaluated following a case-study design in the Clean-in-Place (CIP) system of an industrial beverage plant, aimed at demonstrating architectural feasibility and diagnostic capabilities under realistic operating conditions rather than statistical generalization across populations of plants. While tailored here to CIP, the architectural components—process-aware context management, hybrid deterministic/LLM supervision, and incremental agent deployment—transfer to other multi-stage batch operations (fermentation, distillation, pasteurisation) by reconfiguring process-specific rules and ontologies [6,12].

Unlike existing LLM-based assistants that typically operate as generic chatbots over manuals, historical databases or SCADA tags, the proposed architecture treats CIP executions as first-class contexts. A set of CIP-aware agents load their context on demand according to the active programme and stage, subscribe to enriched data streams and produce supervisory states, alerts and explanations that are aligned with the lifecycle of real CIP runs. This process-centric view of context enables new decision-support agents (LLM-based or otherwise) to be added incrementally, without modifying the underlying CIP programmes. This process-centric abstraction—treating batch executions as first-class contexts—constitutes the main architectural innovation, enabling heterogeneous AI components to operate over a shared process lifecycle without coupling to specific control logic.

The architecture is instantiated and evaluated on real CIP cycles executed in an industrial beverage plant over a six-month deployment period. From 24 complete executions monitored during this period, three representative cases are purposively selected to provide architectural and operational validation evidence across the diagnostic spectrum observed during deployment: (i) a nominal baseline execution demonstrating routine equipment health, (ii) a preventive-warning case exhibiting subtle operational signals (flow reduction, temperature drift) that do not violate safety thresholds but indicate emerging equipment degradation requiring scheduled maintenance, and (iii) a diagnostic-alert regime capturing multiple concurrent deviations (pump wear, boiler efficiency loss, dosing drift) requiring prioritised maintenance review.

Rather than pursuing large-scale statistical generalisation—which would be infeasible given the operational frequency of CIP cycles (1–3 executions per day) and the stable baseline achieved through prior control optimisation—this case-study evaluation demonstrates how the decision-support layer interprets operational signals, issues contextualised alerts, and generates natural-language diagnostic summaries that inform preventive maintenance decisions, while preserving the determinism required for safety-critical alarm handling. The focus is on proving architectural feasibility and showing that the diagnostic patterns identified by the system align with maintenance needs confirmed by plant operators and maintenance logs.

1.3. Contributions

The main contributions of this paper are:

- A batch-process-aware, multi-agent decision-support architecture that treats batch executions (such as CIP cycles) as first-class contexts. Agents load their context on demand according to the active programme and stage, subscribe to enriched data streams and produce supervisory states, alerts and explanations aligned with the lifecycle of real batch runs.

 doi:10.20944/preprints202512.2023.v1

- A process-centric context management approach that allows heterogeneous AI components (rule-based agents, fuzzy logic, neural networks, LLM-based assistants) to be added incrementally. New agents can reuse the same process context and message bus without modifying the underlying control programmes or restarting the supervision layer.
- A set of process-level evaluation metrics that quantify the behaviour of the decision-support layer over real executions, including compliance with stage specifications, consistency with state specifications, and stability of state labelling, complemented by spot checks of the numerical consistency between language-based summaries and enriched logs.
- An experimental study on three complete CIP runs that instantiates the architecture in a real cleaning-in-place application, demonstrating its ability to maintain high specification compliance across nominal, preventive and diagnostic scenarios, provide coherent and stable supervisory states, and generate data-grounded natural-language explanations in real time without altering the existing CIP control logic. The case studies collectively illustrate how diagnostic interpretation of alert patterns across execution cycles can inform preventive maintenance scheduling—addressing pump wear, boiler degradation and dosing system calibration—before operational impact occurs.

Collectively, these contributions advance the state of knowledge by demonstrating— through a real industrial deployment—that hybrid deterministic/LLM architectures can be integrated into safety-critical batch supervision to support maintenance-oriented decision-making, and by providing a reproducible architectural pattern and evaluation methodology that can guide future implementations in other batch process domains.

The architecture has been implemented and deployed in a real industrial environment, supporting CIP operations at the VivaWild Beverages plant over a six-month validation period, which provides the basis for the experimental evaluation reported in this paper.

1.4. Paper Organisation

The remainder of this paper is organised as follows. Section 2 reviews related work on bioprocess monitoring, Industry 4.0 architectures, LLMs in industrial settings, and task planning. Section 3 presents the generic batch process supervision architecture, including the rationale for a layered, agent-based design and the process-agnostic instantiation pattern. Section 4 details real-time data management and LLM integration strategies. Section 5 describes the experimental methodology, including the industrial deployment, selection rationale for representative execution cases, data collection procedures and evaluation metrics. Section 6 presents results from three representative CIP executions, quantifying compliance, state consistency, stability and LLM fidelity. Section 7 discusses the findings, positions the work relative to existing approaches, addresses limitations and outlines directions for future longitudinal analysis. Section 8 concludes the paper.

2. Related Work

2.1. Bioprocess and CIP Monitoring and Control

Monitoring and control of industrial bioprocesses has been extensively studied, with a strong emphasis on state estimation, soft sensors, fault detection and advanced control strategies such as model predictive control [1]. Recent advances in soft sensor modeling have explored multiple paradigms: interval type-2 fuzzy logic systems for uncertainty handling [13], deep learning approaches for feature representation in high-dimensional data [14], fuzzy hierarchical architectures that combine rule-based reasoning with adaptive learning for improved interpretability [15], and deep neural network-based virtual sensors for retrofitting industrial machinery without additional physical instrumentation [16]. Within this broader context, cleaning-in-place (CIP) systems are recognised as critical operations that ensure hygienic conditions and prevent cross-contamination between batches [2], but also as major contributors to water, chemical and energy consumption [4].

Reported approaches for CIP focus mainly on sequence design, parameter optimisation and endpoint detection, including improved scheduling of cleaning cycles, optimisation of temperature and

 doi:10.20944/preprints202512.2023.v1

flow profiles, and rule-based supervision to guarantee that each stage meets predefined set-points [2–4]. In most cases, the supervisory logic is implemented directly in PLCs and SCADA systems, with alarm rules defined as static thresholds or simple logical combinations, providing limited support for richer situation awareness, multi-variable diagnostics, cross-cycle trend analysis or operator guidance for preventive maintenance beyond trends and alarm lists in production environments.

While these approaches successfully detect threshold violations and ensure regulatory compliance, they typically operate on a per-execution basis and do not explicitly support the interpretation of subtle operational signals that may indicate emerging equipment degradation (for example, gradual pump wear, boiler efficiency drift or dosing system calibration errors) before they impact process outcomes. Recent soft sensor frameworks have demonstrated improved handling of uncertainty [13], nonlinear pattern recognition [14], and rule-based feature processing with fuzzy reasoning [15], yet their integration with conversational interfaces for maintenance-oriented decision support remains unexplored. The supervisory challenge has consequently shifted from detecting imminent failures to identifying preventive maintenance opportunities through longitudinal trend analysis and diagnostic pattern recognition across multiple CIP cycles.

2.2. Industry 4.0 Architectures and Microservice-Based Automation

The move towards Industry 4.0 has motivated a variety of reference architectures and frameworks that aim to increase modularity, interoperability and scalability in industrial automation [5]. These frameworks typically distinguish between physical assets, edge computing resources and higher-level information systems, and promote the use of standard communication protocols and service-oriented designs as enablers of flexible, connected production environments [17]. Building on these ideas, previous work has proposed component-based microservice frameworks for bioprocess automation, in which control, monitoring, HMI, data logging and higher-level coordination are implemented as independent, containerised components interconnected through a message bus [6,12]. These frameworks have demonstrated how microservices can satisfy real-time and robustness requirements in bioprocess applications while supporting flexible reconfiguration and reuse across equipment such as bioreactors and CIP units. Other approaches have explored multi-agent reinforcement learning for optimizing manufacturing system yields through coordinated agent contributions [18], demonstrating the value of distributed intelligence in complex production environments. However, these and other microservice-oriented approaches largely focus on structural and communication concerns; they do not detail how hybrid deterministic and LLM-based agents can be embedded into the automation stack to support real-time supervision, diagnostic interpretation and preventive maintenance decision-making for concrete operations such as CIP.

The present paper retains a microservice-style architecture but shifts the focus from structural aspects to decision support, diagnostic interpretation and explanation: the agents, reasoning layer and conversational interface are implemented as loosely coupled services that subscribe to enriched data streams and publish supervisory states, diagnostic alerts, maintenance-oriented reports and trend analyses. The evaluation presented here therefore complements prior architectural studies by quantifying how such a service-based, multi-agent layer behaves when deployed over real CIP executions, and by demonstrating its ability to differentiate nominal equipment health, preventive warning patterns and diagnostic alert regimes that are meaningful for maintenance planning.

2.3. LLMs and Conversational Assistants in Industrial and CPS Settings

The emergence of large language models (LLMs) has motivated a growing body of work on industrial and cyber-physical applications. Recent surveys discuss cross-sector industrial use cases of LLMs, including maintenance support, incident analysis and internal knowledge management, and highlight both opportunities and risks in terms of robustness and governance.[19–21] Several authors propose architectures where on-premise LLMs are integrated with IoT and cyber-physical systems in Industry 4.0, typically acting as a semantic layer or conversational front-end for heterogeneous data sources.[22–25]

 doi:10.20944/preprints202512.2023.v1

Lim et al. propose frameworks in which LLMs are embedded into industrial automation systems as intelligent orchestration components, enabling multi-agent coordination, natural-language task interpretation and adaptive manufacturing execution [26]. These contributions demonstrate a clear interest in bringing LLMs closer to operational technology, but most remain at the level of high-level frameworks or generic use cases rather than detailed, domain-specific instantiations that operate continuously on live process data, track equipment health trends across multiple execution cycles and interact directly with plant operators for maintenance-oriented decision-making.

Existing work on LLM-based assistants for industrial systems largely focuses on providing conversational access to documentation, historical data or SCADA/IoT tags in real time, often evaluating performance in terms of question answering precision and response generation [27]. Recent applications have demonstrated how LLMs can facilitate natural language queries over complex manufacturing process data and real-time sensor streams, enabling production personnel to interact with high-dimensional datasets without requiring specialized analytical expertise [28]. Retrieval-augmented generation (RAG) approaches in industrial settings have shown significant improvements in domain-specific knowledge retrieval, achieving recall rates above 85% for technical service and regulatory documentation [29]. Complementary work on multimodal LLM-based fault detection has shown how GPT-4-based frameworks can improve scalability and generalizability across diverse fault scenarios in Industry 4.0 settings [30], though these approaches primarily target fault classification rather than continuous process supervision and preventive maintenance planning. However, existing LLM-based approaches for manufacturing data exploration typically focus on real-time visualization and anomaly detection in continuous production environments [28], rather than providing lifecycle-aligned supervision for multi-stage batch processes with preventive maintenance decision support. In contrast, this work targets a concrete batch process (CIP) and evaluates a decision-support architecture that integrates rule-based supervision, enriched data streams and language-based interaction within a process-centric context aligned with the lifecycle of CIP executions.

2.4. LLMs for Robotics, Task Planning and Control Logic

A substantial portion of applied LLM research in industry-related domains comes from robotics and task planning. Concrete implementations show LLM-based task and motion planning for construction robots, [31] as well as frameworks such as DELTA that decompose long-term robot tasks into sub-problems and translate them into formal planning languages [32]. Recent work on embodied intelligence in manufacturing has demonstrated how LLM agents such as GPT-4 can autonomously design tool paths and execute manufacturing tasks, achieving success rates above 80% in industrial robotics simulations [33], while parallel work on LLM-based mobile robot navigation has shown that models such as Llama 3.1 can dynamically generate collision-free waypoints in response to natural language commands and environmental obstacles [34]. Complementary work on LLM-based digital intelligent assistants in assembly manufacturing has demonstrated significant improvements in operator performance, user experience, and cognitive load reduction [35].

Recent work on human-centric smart manufacturing has shown how LLM-based conversational interfaces can lower the digital literacy barrier for operators by enabling natural-language access to real-time machine data, as demonstrated by the ChatCNC framework for CNC monitoring [36]. Complementarily, Xia et al. propose an LLM-driven industrial automation framework where multi-agent structures, structured prompting, and event-driven information models enable end-to-end control, from interpreting real-time events to generating production plans and issuing control commands [37]. These works focus primarily on code generation, configuration, and verification of control logic *prior to deployment*, rather than on continuous supervision, diagnostic interpretation, and operator support *during runtime*. Critically, none of these contributions address the integration of LLM-based analytics with deterministic supervisory logic for preventive maintenance planning through cross-cycle trend analysis and equipment health monitoring—the focus of the present work.

 doi:10.20944/preprints202512.2023.v1

2.5. Positioning of the Present Work

Overall, existing literature shows intense activity around LLMs for documentation retrieval, maintenance support, robotics and control-code generation in industrial and CPS environments. However, there is a noticeable gap regarding batch processes and safety-critical cleaning operations such as CIP, particularly in terms of architectures that combine deterministic supervision with LLM-based diagnostic interpretation to support preventive maintenance decision-making. To the best of our knowledge, no prior work reports an architecture that simultaneously: (i) integrates deterministic agents for state estimation and safety monitoring with LLM-based analytics operating over live process buffers; (ii) supports cross-cycle trend analysis for preventive maintenance through enriched data streams; (iii) combines structured process rules with LLM reasoning to ensure process adherence while minimizing hallucinations [38]; and (iv) quantitatively evaluates the system's ability to differentiate nominal equipment health, preventive warning patterns and diagnostic alert regimes across real industrial executions.

The present work addresses this gap by proposing, implementing and evaluating such an architecture in a real CIP deployment, showing that it is feasible to combine rule-based supervision and language-based assistance in safety-critical batch processes while maintaining coherent supervisory states, data-grounded explanations and actionable diagnostic patterns that support preventive maintenance planning and equipment health monitoring. The case-study evaluation demonstrates that the architecture can distinguish between nominal baseline operation, preventive maintenance signals and diagnostic alert conditions, providing operators with contextualised, maintenance-oriented decision support that goes beyond traditional threshold-based alarm systems.

3. Generic Batch Process Supervision Architecture

The proposed system provides a generic, layered architecture for AI-assisted supervision and diagnostic interpretation of industrial batch processes. While the architecture is instantiated and evaluated here in the context of Clean-in-Place (CIP) operations in a beverage plant, its design is process-agnostic: the same pattern applies to fermentation, distillation, pasteurisation, chemical synthesis and other multi-stage batch procedures by reconfiguring process-specific rules, ontologies and enrichment logic. The architecture spans from operator-facing interfaces down to physical process equipment, with intermediate layers that handle orchestration, agent coordination, diagnostic pattern recognition for equipment health monitoring, and shared data management.

At its core, an event bus exposes a global data space where agents, user interfaces and the control layer exchange events and enriched data streams, while an intermediate orchestration layer routes operator requests to specialised agents according to user intention and process context.[6,17] Figure 1 illustrates the main layers: (i) the operator interaction layer, (ii) the orchestration and intent-routing layer, (iii) the agent layer (control, configuration and analysis), (iv) the data and event hub, and (v) the physical execution layer.

A key advantage of adopting an orchestrator-based, service-oriented architecture is that new decision-support agents can be added incrementally, without modifying the underlying batch programmes or restarting the supervision layer. Each agent subscribes to the same enriched data streams and loads its context according to the active batch execution and stage, producing additional supervisory states, diagnostics, preventive maintenance recommendations or trend analyses. This design makes it possible to combine heterogeneous AI techniques (for example, rule-based agents, fuzzy logic, neural networks, LLM-based assistants) within a common framework, and to evolve the decision-support and diagnostic capabilities over time as new agents are deployed [5,6].

In this sense, the architecture does not assume a single monolithic AI component, but rather a distributed set of process-aware agents coordinated by an orchestrator. The evaluation presented in this work focuses on one such configuration, where rule-based supervision and a language model assistant share the same batch context for CIP operations; however, the same pattern can be used to integrate further agents (for example, predictive models or optimisation modules) without disrupting

 doi:10.20944/preprints202512.2023.v1

existing services or batch control programmes [6,12]. In the deployed configuration, operators interact through a web interface that sends normalised events to a query router, which dispatches them to a set of process-aware agents—all connected through the shared event bus and enrichment layer.

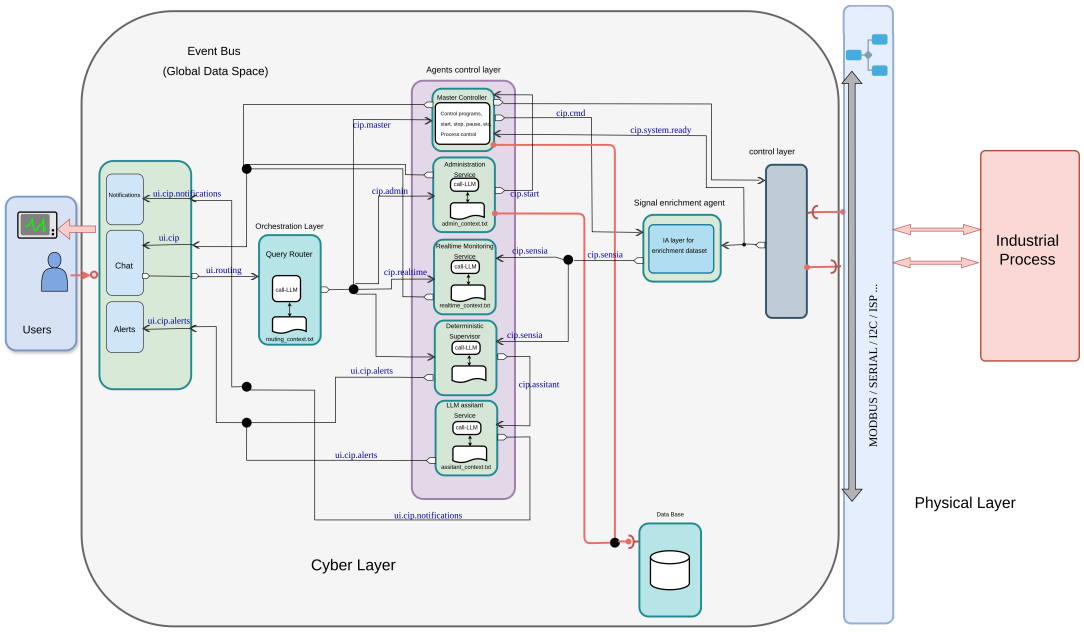

Figure 1. Generic layered architecture for batch process decision support and diagnostic interpretation. Operators interact through a web interface (chat, alarms and notifications) that sends normalised events to a query router in the orchestration layer. Process-aware agents—including the Batch Master Controller, Administration Service, Deterministic Supervisor, Realtime Monitoring Service and LLM Assistant Service—exchange events and enriched data via a shared event bus and signal-enrichment agent, while the physical layer remains under PLC/SCADA control. In the CIP deployment, these components manage cleaning cycles and support preventive maintenance decision-making; in other batch processes, the same pattern supervises fermentation, distillation or synthesis runs with diagnostic capabilities aligned to equipment health monitoring.

The proposed decision-support layer has been implemented as a set of microservices and agents deployed alongside an existing automation system at an industrial plant. The agents subscribe to the plant's real-time data streams and batch execution events, and publish supervisory states, diagnostic alerts, preventive maintenance recommendations and reports to the same infrastructure used in day-to-day operation. The evaluation in Section 6 is based on logs collected from this deployment during normal production runs, demonstrating the architecture's ability to differentiate nominal, preventive and diagnostic operational regimes.

3.1. Process-Agnostic Design and Instantiation Examples

Table 1 illustrates how the same architectural components map to different batch process types, with process-specific parameters configured at deployment. The CIP case study discussed in this paper corresponds to the first instantiation column; analogous deployments for fermentation or distillation would reuse the same components with different recipes, variables and rules.

This abstraction enables the same codebase and agent logic to supervise multiple process types, with only rules, ontologies and variable mappings reconfigured per application. The CIP evaluation in Section 6 demonstrates this pattern in production; fermentation and distillation deployments would follow the same architecture with domain-specific enrichment and diagnostic rules.

 doi:10.20944/preprints202512.2023.v1

Table 1. Generic batch architecture components and process-specific instantiations.

Component	Generic role	CIP instance	Fermentation instance
Batch Master	Orchestrate programme execution	Manage cleaning circuits and stages	Manage inoculation, feeding, harvest
Administration	Manage recipes and equipment	Programmes, circuits, tanks	Recipes, vessels, media prep.
Deterministic Supervisor	Compute states and alarms	Temp., flow, cond. thresholds	pH, DO, temp., biomass thresholds
Realtime Monitoring	LLM analytics	Ad hoc queries, trend analysis on CIP	Ad hoc queries, trend analysis on fermentation
Offline Analysis	Post-run reports	KPI trends, cleaning efficiency, maintenance signals	KPI trends, yield metrics, bioreactor health

3.2. Rationale for a Layered, Agent-Based Design

Instead of deploying a single, monolithic large language model (LLM) with access to all plant data and control interfaces, the proposed architecture adopts a layered, agent-based design. This choice is motivated by several considerations:

- **Context management and efficiency**: Industrial environments produce high-volume, heterogeneous data across multiple batch lines and equipment. Concentrating all information into a single LLM context would be inefficient and difficult to control. By separating operator interaction, orchestration and specialised agents, each component only handles the subset of data and functions relevant to its role, enabling smaller, faster contexts and more predictable behaviour [37].
- **Safety and decentralisation of intelligence**: Safety-critical logic (e.g., interlocks, sequence enforcement, hard alarms) remains in deterministic agents and PLC/SCADA systems, while LLM-based components are confined to explanatory and analytical roles. This decentralisation avoids granting a single LLM direct authority over control actions and supports explicit validation paths for any recommendation before it affects the process.
- **Flexibility and evolution**: Food, beverage and chemical plants often operate under maquila-like conditions, with frequent product changes, contract manufacturing and evolving cleaning or processing requirements. A layered architecture with loosely coupled agents and a generic data hub allows new analysis functions, additional lines or updated recipes to be introduced without redesigning the entire system, supporting continuous adaptation and incremental deployment [6,17].
- **Scalability across services and sites**: As production scales to multiple lines, services or sites, the same pattern can be replicated: interaction and orchestration remain similar, while additional agent instances are deployed per line or plant. This aligns with microservice and Industry 4.0 principles, enabling horizontal scale-out and reuse of components across different customers and service contracts [5,6].

In summary, the layered architecture decentralises intelligence across specialised agents optimised for specific intentions (control, configuration, deterministic supervision, real-time analytics and offline analysis), reduces the need for large, global LLM contexts, and preserves safety and scalability in settings where processes, products and cleaning or conversion requirements evolve continuously.

3.3. Process-Aware Context Management

The decision-support layer is implemented as a set of loosely coupled agents coordinated by an orchestrator. Each agent is *process-aware*: upon receiving a request or detecting a new batch execution (for example, a CIP cycle, fermentation batch or distillation run), it loads the corresponding context (programme, equipment configuration, current stage, enriched variables and rule-based specifications) and subscribes to the relevant data streams. Within this context, the agent produces supervisory states, notifications, reports or language-based explanations that are specific to the active batch run.

In the CIP deployment, the context includes cleaning circuit, tank, programme, stage and key variables such as temperature, flow and conductivity. In a fermentation deployment, it would instead

 doi:10.20944/preprints202512.2023.v1

comprise vessel, recipe, inoculum, growth phase and variables such as pH, dissolved oxygen and biomass proxies. In distillation, it would include column configuration, feed composition and reflux schedule. This process-centric abstraction enables agents to be added or updated incrementally: new agents—whether based on rules, fuzzy logic, neural networks or large language models—can subscribe to the same process contexts and publish their outputs to the common message bus, without requiring changes to the underlying control logic or to other agents. In this sense, the architecture supports a distributed ecosystem of AI components that can evolve over time while sharing a consistent, process-centric view of the system.

3.4. Operator Interaction Layer

At the top, a web-based operator interface provides multiple interaction modalities: natural-language chat, structured alarm panels and visual summaries of process status. Operators can ask questions in free text (e.g., about current batch progress, recent anomalies or historical comparisons), request specific functions (e.g., remaining time for a stage, list of active batch executions) or issue management actions (e.g., creating or starting batch requests). The interface normalises these inputs and forwards them to the orchestration layer as high-level events containing the user query, role and context.

The same interaction layer can be used across different process types by exposing a unified vocabulary (e.g., “current batch”, “last stage”, “previous run”) that is resolved internally to process-specific concepts (CIP cycle, fermentation batch, distillation campaign). This abstraction reduces training effort for operators who work across multiple units or services.

3.5. Orchestration and Intent-Routing Layer

The orchestration layer acts as a mediator between user intents and the underlying agents. It receives normalised requests from the interaction layer and classifies them into intent categories such as: control (start/stop batch, interact with the master process), configuration (programmes, equipment, permissions), deterministic analysis (current state, diagnostics, time estimates), real-time free analysis (ad hoc queries over the live buffer) and offline analysis (post-run reports over stored data).Based on this intent and the user role, the orchestrator dispatches the request to the appropriate agent via the data and event hub, and later aggregates or reformats the response into a coherent message for the operator interface.

3.6. Agent Layer: Control, Configuration and Analysis

Below the orchestrator, a set of specialised agents implement the core system functions. These agents are defined generically for batch processes, with process-specific parameters configured at deployment:

- **Batch Master Controller**: coordinates the execution of batch cycles at the physical layer, interacting with PLCs or SCADA to start, stop or pause programmes on specific equipment. In the CIP deployment, this agent manages cleaning circuits and stages; in fermentation, it would orchestrate inoculation, feeding and harvesting phases.
- **Administration Service**: manages configuration and governance, including batch recipes (programmes), equipment definitions, resource states and role-based permissions. It exposes CRUD-style operations and workflow support for batch execution requests, ensuring that only authorised actions are propagated and that equipment availability constraints are respected. The same service structure applies to CIP circuits, fermentation vessels or distillation columns.
- **Deterministic Supervisor**: performs deterministic, rule-based supervision and diagnostics in real time. It consumes pre-processed sensor streams, maintains sliding windows of recent data and periodically computes state summaries, diagnostics and hard safety alerts (NORMAL/WARNING/CRITICAL), which are published as structured notifications and alarms for the operator interface. Rules and thresholds are defined per process type (for example, temperature and flow envelopes for CIP stages, or pH and dissolved oxygen bounds for fermentation phases).

 doi:10.20944/preprints202512.2023.v1

- **Realtime Monitoring Service**: focuses on flexible, operator-driven analysis over the live process buffer. It maintains a rolling data-frame representation of the active batch trajectory and answers ad hoc queries using deterministic computations and an LLM-backed analytics layer. In the deployed configuration, the LLM component uses Qwen 2.5 (7 billion parameters) running locally via Ollama on NVIDIA GPU infrastructure, providing sub-second inference times for diagnostic queries while maintaining data sovereignty and avoiding cloud dependencies This enables correlations, aggregate statistics and narrative explanations during the run without introducing external API latency or compliance concerns. The same agent structure applies to any batch process with time-series sensor data.
- **Offline Analysis Service**: operates on persisted historical data after batch completion, generating post-run reports, comparative analyses across cycles and higher-level insights that are not subject to strict real-time constraints. Its outputs can be requested by operators via the same orchestrated interface, but are computed over archived time-series rather than the live buffer.

This agent layer separates safety-critical control and supervision from exploratory analytics: the master and deterministic agents enforce strict rules and temporal constraints, while the real-time and offline analysis agents provide more flexible, LLM-enhanced insights without affecting core control logic.[37]

3.7. Data and Event Hub

The data and event hub underpins communication between all agents and the physical layer. In the deployed system, this hub is implemented using Redis (version 7.2), which provides:

- publish/subscribe channels (via `SUBSCRIBE`/`PUBLISH` commands) for real-time events and sensor data (for example, batch start/stop, stage transitions, time-stamped readings);
- Redis Streams (`XADD`/`XREAD`) for time-series storage of per-batch trajectories of key variables, enabling efficient temporal queries; and
- hash structures (`HSET`/`HGET`) for configuration (recipes, equipment), resource status and the state of batch execution requests.

In the CIP deployment, these structures store cleaning circuits, programmes and sensor readings (temperature, flow, conductivity). In fermentation, they would store vessels, recipes and process variables (pH, dissolved oxygen, biomass proxies); in distillation, they would track columns, feed schedules and reflux ratios. The hub interface remains identical across process types, decoupling producers (PLCs, sensors) from consumers (agents, UI), supporting multiple concurrent batch processes, and allowing analytical and diagnostic capabilities to be extended without changes to the underlying control hardware.

4. Real-Time Data Management and LLM Integration

The proposed architecture explicitly balances memory usage, query latency and quality of assistance by combining compact, enriched data buffers with decentralised analysis agents that support both immediate supervisory decisions and longitudinal trend analysis for preventive maintenance.

4.1. Bounded In-Memory Buffers and Latency

Each active CIP instance maintains an in-memory buffer with a fixed maximum number of records, corresponding to a few hours of operation at the given sampling rate, instead of persisting the entire plant history in the LLM context. This rolling dataset is exposed through multiple logical windows: a short horizon (e.g., last 2 minutes) for real-time state estimation, a medium horizon (e.g., last 5–10 minutes) for periodic diagnostics and preventive warning detection, and a full-cycle view for exploratory real-time queries, cross-cycle trend analysis and offline post-run analysis. By keeping buffer size bounded, deterministic statistics and LLM prompts can be computed with near-constant time and memory, even as the plant scales to more CIP circuits or higher sampling frequencies.

 doi:10.20944/preprints202512.2023.v1

4.2. Enriched, Decentralised Data Views

Incoming sensor records are enriched in real time with additional attributes such as CIP program and stage labels, elapsed and remaining time, progression percentage, fuzzy quality indices and short textual descriptors of detected issues, recommended actions and maintenance implications. These enriched records are partitioned per CIP instance and per agent, so that each agent only receives the subset of variables needed for its function (e.g., deterministic supervision windows for immediate alerts, real-time analytics buffers for diagnostic interpretation, offline archives for longitudinal trend correlation with maintenance records), avoiding a single centralised, monolithic dataset. This decentralisation reduces contention and enables real-time support for diagnostics, preventive warning generation and maintenance-oriented decisions at the agent level, while still allowing higher-level aggregation when required for cross-cycle trend analysis.

4.3. Compact LLM Contexts and Efficient Queries

To avoid high latency and cost, the LLM never receives raw, unfiltered streams; instead, the real-time analytics agent constructs compact prompts that combine aggregate statistics over the relevant window, a small set of representative samples (e.g., most recent records or identified outliers) and contextual metadata such as program, stage, circuit and recent alert patterns. This strategy keeps context size small and stable while preserving the information required for meaningful explanations, diagnostic pattern recognition and preventive maintenance recommendations, making it feasible to answer natural-language queries with response times compatible with operator workflows in real plants. The integration layer treats the LLM as a pluggable component accessed through a uniform API, so different on-premise or cloud models can be used without changing the surrounding data management strategy.

4.4. Lowering the Expertise Barrier for Real-Time Decisions and Maintenance Planning

Because diagnostic windows and enriched attributes are computed deterministically and exposed in structured form, operators receive interpretable summaries (e.g., state class, quality grade, main issues, suggested actions and maintenance implications) without having to interpret raw trends or write complex analytical queries.The LLM layer then builds on these structured diagnostics to provide natural-language explanations, cross-cycle trend comparisons and preventive maintenance recommendations, allowing less specialised staff to understand CIP performance, identify emerging equipment degradation patterns and take timely decisions without requiring deep training in control theory, statistical analysis or maintenance planning. This combination of bounded, enriched buffers and layered LLM integration directly supports real-time analysis, diagnostic interpretation and maintenance-oriented decision support in complex industrial environments, while keeping computational and training costs under control.

4.5. Illustrative Data Views and Query Profiles

Table 2 summarises the main data windows maintained per CIP instance and their intended use, while Table 3 illustrates typical query types, underlying sources and expected response times.

Table 2. Main data windows per CIP instance and their intended use.

Data view	Horizon	Primary use
Short window	≈2 min	Instantaneous state, fast alarms
Medium window	5–10 min	Periodic diagnostics, preventive warnings
Full-cycle buffer	30–90 min	Real-time exploratory analysis, trend queries
Historical archives	Days–weeks	Offline reports, benchmarking, maintenance correlation

 doi:10.20944/preprints202512.2023.v1

Table 3. Example query profiles, data sources and typical response times.

Query type	Source	Window	Latency
Current state of batch X	Determ. agent	Short	< 100 ms
Warnings in last stage	Determ. + LLM	Medium	0.3–0.8 s
Compare current vs. previous	LLM on buffer	Full-cycle	1–2 s
Compare with previous cycles	LLM + archives	Historical	2–5 s
Weekly summary + maintenance recommendations	Offline analysis	Historical	s–min

4.6. Fuzzy Enrichment of Real-Time CIP Data

Raw CIP sensor readings alone (e.g., temperature, flow, conductivity, volume) are often difficult to interpret directly in the control room, especially when multiple variables must be considered simultaneously or when subtle degradation patterns must be distinguished from normal process variability. To provide operators and downstream agents with more actionable information, the system applies a fuzzy evaluation layer in real time, which maps continuous variables and their deviations from nominal profiles into linguistic assessments, quality indices and maintenance-oriented diagnostic labels.

For each time-stamped record, the enrichment pipeline:

1. normalises the raw measurements with respect to the target CIP program and current stage (e.g., expected temperature, flow and conductivity ranges for an alkaline wash);
2. evaluates a set of fuzzy rules that capture expert knowledge, such as "temperature slightly low but within tolerance" or "flow persistently below minimum, suggesting pump wear", combining multiple variables and short-term trends; and
3. outputs a discrete state label (e.g., `NORMAL`, `WARNING`, `CRITICAL`), a continuous quality grade in $[0, 1]$, and short textual descriptors summarising the main issues, recommended actions and maintenance implications.

The resulting enriched record contains, in addition to the raw sensor values and timestamps, fields such as `stage`, `progress_percent`, `remaining_time`, `state`, `quality_grade`, `motives` and `actions`, as well as equipment and circuit identifiers and auxiliary counters (e.g., number of recent warnings or critical samples). Table 4 shows a simplified example of such a record.

Table 4. Illustrative example of an enriched CIP data record.

Field	Example value
`timestamp`	2025-11-19T18:05:50.123Z
`cip_id`	`cip20251119T180530_abc12345`
`program`	`ALKA_STANDARD`
`stage`	`ALKALINE`
`seconds_elapsed`	320.5
`seconds_remaining`	579.5
`progress_percent`	35.6
`temp` [°C]	61.5
`flow` [L/s]	2.8
`cond` [μS/cm]	1657.2
`volume` [L]	204.1
`state`	`NORMAL`
`quality_grade`	0.82
`motives`	`Parameters within target ranges`
`actions`	`Continue current stage`

From a resource perspective, the bounded-buffer design keeps the memory footprint per CIP instance relatively small: for example, a buffer of 10^3–10^4 records with a dozen numeric and categorical fields typically remains in the order of a few megabytes in memory, even when multiple CIP lines are monitored concurrently. This is modest compared to the memory requirements of the LLM itself

 doi:10.20944/preprints202512.2023.v1

and allows deterministic statistics and prompt construction to execute with predictable latency. At the same time, the enriched representation and pre-computed diagnostics provide enough information for operators to take online decisions directly from the generated summaries and explanations, including identification of preventive maintenance opportunities and equipment health trends, without resorting to external tools or manual data export.In practice, this combination of low in-memory cost and high decision support value—spanning immediate supervisory needs and longer-term diagnostic interpretation—is essential for deploying AI-assisted supervision in industrial environments where hardware resources and real-time constraints are non-negotiable.

4.7. Computational Resource Profile

Table 5 summarises the computational footprint of the main system components in the deployed configuration. The memory requirements per CIP instance remain modest (order of a few megabytes for the enriched buffer), while the LLM service (Qwen 2.5 7B via Ollama) operates as a shared resource across all active CIP lines, with inference latencies compatible with interactive operator workflows and real-time diagnostic queries.

Table 5. Computational resource profile for the deployed decision-support architecture (single-line CIP monitoring).

Component	Resource type	Usage
Redis in-memory buffer (10^4 records)	RAM	≈5 MB per CIP
Enriched record (per sample)	RAM	≈0.5 kB
Qwen 2.5 7B model (Ollama)	VRAM (GPU)	4.5 GB (shared)
Agent services (Python containers)	RAM	100–150 MB per agent
LLM query (median)	Latency	1–2 s
Deterministic state update	Latency	<100 ms

Because the LLM runs locally on NVIDIA GPU infrastructure, query latencies remain predictable and do not depend on external cloud API availability or network conditions. This edge deployment strategy addresses the computational and energy constraints inherent in cloud-based LLM architectures for IIoT environments [39], while maintaining data sovereignty and reducing communication overhead. Recent work on edge-cloud collaboration for LLM task offloading in industrial settings has demonstrated that local inference can reduce latency by 60–80% compared to cloud-based alternatives [39], supporting the architectural decision to deploy Qwen 2.5:7B on dedicated GPU infrastructure rather than relying on external API services.

5. Evaluation Methodology

5.1. Evaluation Objectives

This study evaluates the *architectural feasibility* and *operational capability* of the proposed hybrid AI framework in a real industrial environment, rather than pursuing large-scale statistical validation of process performance improvements.

Accordingly, the evaluation focuses on how the decision-support layer behaves when exposed to authentic CIP executions, and whether it fulfils its intended role as a real-time, operator-oriented supervisory layer that supports diagnostic interpretation and preventive maintenance planning. The study addresses the following research questions:

- **RQ1**: Can the architecture maintain coherent supervisory states across diverse CIP conditions?
- **RQ2**: Do the agents respond within application-level real-time constraints during production operation?
- **RQ3**: Does the LLM-based analytics layer provide grounded, numerically consistent explanations over enriched process data?
- **RQ4**: Is the system deployable and operable in an actual CIP installation without disrupting existing PLC/SCADA programmes?

 doi:10.20944/preprints202512.2023.v1

- **RQ5**: Can the architecture differentiate nominal, preventive and diagnostic alert patterns in a way that is meaningful for maintenance decision-making?

The primary goal is thus to demonstrate that the architecture can be instantiated, operated and queried in a working plant, and that its supervisory outputs remain coherent, stable, data-grounded and diagnostically useful over complete CIP executions.

5.2. Case Selection Rationale

Rather than sampling a large number of nearly identical cycles, the evaluation adopts a case-study approach using three representative CIP executions drawn from ongoing production at the VivaWild Beverages plant. These executions were selected to span the typical operational envelope observed in day-to-day operation and to illustrate complementary diagnostic scenarios: (a) a fully nominal baseline, (b) executions with preventive warnings that do not compromise stage success, and (c) executions with more pronounced deviations that generate diagnostic alerts under operational stress.

Table 6. Selected CIP executions and rationale.

Case	Type	Rationale
CIP1	Nominal baseline	Baseline behaviour under standard operating conditions, with variables close to their nominal profiles and no warnings or alerts. Serves as a reference for normal equipment health and operator performance.
CIP2	Preventive warnings	Typical process variations in alkaline and rinsing stages (e.g., slightly reduced flow or temperature excursions that remain within specification) that trigger `WARNING`-level diagnostics. These runs complete successfully but indicate emerging maintenance needs (e.g., pump wear, boiler efficiency drift).
CIP3	Diagnostic alerts	Presence of more pronounced deviations, including clusters of alerts related to flow or conductivity, used to stress-test alert generation and diagnostic capabilities under more demanding conditions that remain operationally successful but call for prioritized maintenance.

These cases are representative of normal plant operation. The CIP process at VivaWild is governed by standard operating procedures, automated recipe management and regulatory constraints, and exhibits high repeatability across executions in terms of temperature, flow and conductivity profiles.

In optimised industrial plants, CIP failures are rare by design; most executions complete successfully within specifications. The diagnostic challenge is therefore not detecting catastrophic faults—which traditional threshold-based alarms handle effectively—but rather identifying subtle, longitudinal degradation patterns in executions that still meet regulatory specifications. For example, flow rates declining 2% per cycle over five consecutive executions due to gradual pump wear may generate no critical alarms (each execution remains above the minimum regulatory threshold), yet this pattern signals actionable maintenance opportunities that can prevent unplanned downtime.

From this perspective, analysing 100 consecutive nominally successful CIP runs would provide strong evidence of system stability and low false-alarm rates, but no evidence of diagnostic capability—it would merely confirm that the system does not fail when the process does not fail. Conversely, analysing a small number of carefully chosen executions spanning nominal baseline, preventive warning scenarios and diagnostic alert regimes provides meaningful evidence about the architecture's ability to differentiate operational conditions that are relevant for maintenance decision-making, extract equipment health insights from executions that meet specifications, and distinguish acceptable process variability from systematic equipment drift requiring intervention before operational impact occurs. This case-study validation approach is appropriate for establishing architectural feasibility and diagnostic behaviour in real industrial conditions.

 doi:10.20944/preprints202512.2023.v1

Importantly, the system is deployed continuously and monitors all CIP executions; the three cases are used for *detailed* analysis and illustration of the architecture's diagnostic behaviour. In routine operation, the value of the system emerges from the accumulation and inspection of alert patterns across multiple cycles (e.g., repeated flow warnings indicating pump degradation, recurring temperature deviations suggesting boiler maintenance). Extending the present analysis to longitudinal trend quantification and correlation with maintenance records is left for future work.

The objective of this evaluation is therefore architectural and operational validation of the decision-support layer, not statistical inference about long-term process improvements across all historical CIP data.

5.3. Evaluation Setup

The evaluation concentrates on the behaviour of the multi-agent decision-support architecture on top of the existing PLC/SCADA control layer, which remains unchanged. For each of the selected CIP executions, the agents subscribe to the live data streams and CIP events, maintain their internal contexts, and produce supervisory states, alerts, diagnostics and language-based explanations as they would during routine plant operation.

The study examines whether the architecture:

- tracks the progression of each CIP stage and maintains coherent discrete supervisory labels;
- provides timely detection of out-of-spec situations at the supervisory level, and differentiates nominal operation from conditions that warrant preventive or more urgent maintenance; and
- generates natural-language diagnostics and summaries that remain consistent with the enriched data seen by the agents and are usable for maintenance-oriented decision-making.

No changes are made to the underlying low-level controllers, so any deviations in stage-specification compliance reflect actual plant behaviour rather than experimental manipulation.

5.4. Datasets

The experiments use logs collected from complete CIP runs covering alkaline, sanitising and final rinse stages. Each second, the system records:

- process variables such as instantaneous and windowed flow, temperature, conductivity, pH and accumulated volume;
- stage-level information (CIP programme, circuit, tank, current stage and progression);
- supervisory outputs, including discrete labels (`NORMAL`, `WARNING`, `CRITICAL`), fuzzy quality indices and structured reasons describing the current situation (e.g., parameters within range, low flow conditions, insufficient volume); and
- selected responses from the LLM-based analytics agent, including numerical summaries and narrative explanations.

For each run, the enriched records thus combine raw sensor values with derived features such as moving averages, diagnostic counters (e.g., recent warnings or critical samples) and identifiers of the active CIP programme, circuit and tank. This representation allows the evaluation to relate the behaviour of the decision-support layer to concrete process trajectories and stage-level specifications, and to verify the numerical consistency of language-based outputs.

5.5. Evaluation Metrics

On top of these logs, the methodology defines a set of quantitative metrics that capture complementary aspects of the decision-support behaviour of the architecture. The metrics considered in this work cover:

- stage-specification compliance (time spent within prescribed operating ranges);
- consistency between supervisory labels and process conditions;
- temporal stability of the labels; and

 doi:10.20944/preprints202512.2023.v1

- consistency between language-based explanations and the underlying enriched data.

Additional metrics such as alert sensitivity, specificity, reaction time and anticipation window are formally defined as part of the framework to support future, larger-scale evaluations. In the present case-study campaign, quantitative results focus on compliance, label consistency and stability, while sensitivity-related metrics are inspected qualitatively in the three selected executions.

5.5.1. Stage Specification Compliance

This metric quantifies to what extent each CIP stage operates within its predefined process specification (e.g., flow, temperature and pH ranges). It measures the proportion of time during which the relevant variables remain inside their acceptable bands, providing a stage-level quality indicator that is independent of the underlying controller implementation.

Let a stage s be sampled at discrete time instants $t = 1, \ldots, T_s$, and let $\mathbf{x}_t$ denote the vector of monitored variables at time t. For each stage, a specification function $C_s(\mathbf{x}_t)$ returns 1 if all relevant variables lie within their prescribed ranges at time t, and 0 otherwise. The stage specification compliance is then defined as

$$\text{Compliance}(s) = \frac{1}{T_s} \sum_{t=1}^{T_s} C_s(\mathbf{x}_t).$$

In the experiments, this metric is computed separately for alkaline, sanitising and final rinse stages for each of the three representative executions, and then discussed on a case-by-case basis to characterise how the decision-support layer behaves under nominal, variable and more demanding conditions.

5.5.2. Alert Sensitivity and Specificity

This metric assesses how accurately the architecture detects out-of-spec situations by comparing generated alerts against process conditions derived from the logged variables, in terms of sensitivity (recall) and specificity.

Let each time instant t within a stage be associated with: (i) a binary ground-truth anomaly label $y_t \in \{0, 1\}$, obtained by applying stage-specific rules to the process variables (e.g., low flow, out-of-range temperature or pH); and (ii) a binary alert decision $\hat{y}_t \in \{0, 1\}$, where $\hat{y}_t = 1$ if the architecture issues a `WARNING` or `CRITICAL` label and $\hat{y}_t = 0$ otherwise.

Over a set of time instants $\mathcal{T}$, the sensitivity and specificity are defined as

$$\text{Sensitivity} = \frac{\sum_{t \in \mathcal{T}} \mathbb{I}(y_t = 1 \wedge \hat{y}_t = 1)}{\sum_{t \in \mathcal{T}} \mathbb{I}(y_t = 1)},$$

$$\text{Specificity} = \frac{\sum_{t \in \mathcal{T}} \mathbb{I}(y_t = 0 \wedge \hat{y}_t = 0)}{\sum_{t \in \mathcal{T}} \mathbb{I}(y_t = 0)},$$

where $\mathbb{I}(\cdot)$ denotes the indicator function.[1] These metrics are included to define a complete evaluation framework, but their systematic quantification is left to future work once larger cohorts of CIP executions and annotated events become available.

5.5.3. Reaction Time to Anomalies

This metric quantifies how quickly the architecture reacts once a process variable crosses an out-of-spec threshold, by measuring the delay between the onset of an anomaly and the first alert (`WARNING` or `CRITICAL`) raised by the system.

[1] Sensitivity, specificity, reaction time and anticipation window metrics are defined here to establish a complete evaluation framework for future longitudinal studies with larger annotated datasets. In the present case-study evaluation, systematic quantification of these metrics is limited by the small number of confirmed critical episodes (concentrated in CIP 3) and the absence of comprehensive ground-truth annotations for all anomaly types across the full six-month deployment. These metrics will be quantified in follow-up work once maintenance records are integrated with the alert logs.

 doi:10.20944/preprints202512.2023.v1

Let each anomaly episode k be characterised by: (i) an onset time t_k^{onset}, when a ground-truth condition $y_t = 1$ becomes true (e.g., flow drops below a minimum threshold and remains there); and (ii) an alert time t_k^{alert}, corresponding to the first instant $t \geq t_k^{\text{onset}}$ for which $\hat{y}_t = 1$.

The reaction time for episode k is defined as

$$\Delta t_k^{\text{react}} = t_k^{\text{alert}} - t_k^{\text{onset}}.$$

Over a set of episodes $\mathcal{K}$, the average reaction time can be computed as

$$\text{ReactionTime} = \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \Delta t_k^{\text{react}}.$$

In the present study, reaction times are inspected qualitatively for the selected cases; a systematic computation across larger datasets is left for future work.

5.5.4. Anticipation Window

This metric evaluates how early the architecture warns about conditions that may compromise the success of a CIP stage, such as insufficient final rinse volume or prolonged low-flow operation. It is defined as the time margin between the first relevant alert and the occurrence of a stage-level failure or constraint violation.

For each stage-level failure event j, let: (i) t_j^{fail} denote the time at which the failure or constraint violation is detected (e.g., the stage ends without meeting minimum volume or quality criteria); and (ii) t_j^{warn} denote the earliest time $t \leq t_j^{\text{fail}}$ at which the architecture issues a `WARNING` or `CRITICAL` alert related to that failure mode.

The anticipation window for event j is defined as

$$\Delta t_j^{\text{anticip}} = t_j^{\text{fail}} - t_j^{\text{warn}}.$$

Over a set of failure events $\mathcal{J}$, the average anticipation window can be expressed as

$$\text{AnticipationWindow} = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \Delta t_j^{\text{anticip}}.$$

As with reaction time, this metric is defined to complete the evaluation framework, but its systematic quantification is outside the scope of the present case-study campaign.

5.5.5. State–Specification Consistency

While stage specification compliance focuses on whether the process variables remain within their target ranges, a complementary aspect is whether the discrete supervisory labels (`NORMAL`, `WARNING`, `CRITICAL`) are coherent with those ranges. Intuitively, the architecture should report `NORMAL` when all monitored variables are within specification, and should only escalate to `WARNING` or `CRITICAL` when at least one relevant variable leaves its acceptable band.

Let s be a CIP stage sampled at times $t = 1, \ldots, T_s$, and let $\mathbf{x}_t$ and $C_s(\mathbf{x}_t)$ be defined as above, with $C_s(\mathbf{x}_t) = 1$ indicating that all variables are within specification and $C_s(\mathbf{x}_t) = 0$ otherwise. Let $L_t \in \{\texttt{NORMAL}, \texttt{WARNING}, \texttt{CRITICAL}\}$ denote the discrete label produced by the supervisory layer at time t. A sample is considered *label–consistent* if

$$\big(L_t = \texttt{NORMAL} \wedge C_s(\mathbf{x}_t) = 1\big) \quad \text{or} \quad \big(L_t \in \{\texttt{WARNING}, \texttt{CRITICAL}\} \wedge C_s(\mathbf{x}_t) = 0\big).$$

 doi:10.20944/preprints202512.2023.v1

Defining the indicator

$$D_s(t) = \begin{cases} 1, & \text{if the sample at time } t \text{ is label–consistent,} \\ 0, & \text{otherwise,} \end{cases}$$

the state–specification consistency for stage s over a run is given by

$$\Gamma_s = \frac{1}{T_s} \sum_{t=1}^{T_s} D_s(t).$$

Values Γ_s close to 1 indicate that the discrete supervisory state almost always matches the underlying process conditions, whereas lower values reveal mismatches between labels and actual operation. In the experiments, this metric is computed per stage and per execution, and then analysed qualitatively across the three cases.

5.5.6. Stability of State Labelling

In addition to being semantically coherent, the supervisory labels should exhibit a reasonable degree of temporal stability. Excessive oscillations between `NORMAL`, `WARNING` and `CRITICAL` states, especially in the absence of large changes in the process variables, can overload operators and reduce trust in the system.

For each stage s, let $\{L_t\}_{t=1}^{T_s}$ denote the sequence of discrete labels along the execution, and let Δ_s be the number of state changes within that sequence, i.e.,

$$\Delta_s = \sum_{t=2}^{T_s} \mathbb{I}\big(L_t \neq L_{t-1}\big),$$

where $\mathbb{I}(\cdot)$ is the indicator function. Let $T_s^{\min}$ be the duration of the stage in minutes. The stability of state labelling for stage s is then quantified as

$$\Lambda_s = \frac{\Delta_s}{T_s^{\min}} \quad [\text{changes/min}].$$

Low values of Λ_s indicate stable supervisory behaviour (few label transitions per minute), whereas high values highlight stages where the decision layer oscillates frequently between states, signalling either genuinely unstable conditions or overly sensitive thresholds in the supervision logic. In the evaluation, this metric is interpreted in conjunction with the underlying process trajectories for each case.

5.5.7. Consistency of Language-Based Outputs

To assess whether the language-based explanations remain faithful to the underlying enriched data, the evaluation performs spot checks of numerical summaries and counts reported by the conversational interface. For selected responses, the averages and counts stated in the explanation are compared against the statistics computed directly from the corresponding windows of enriched records. This provides qualitative evidence of data-grounded behaviour and helps identify potential hallucinations or inconsistencies in the LLM layer.

5.6. Experimental CIP Runs

Figure 2 provides an overview of the three experimental CIP runs used for evaluation. Each run comprises the same sequence of stages (pre-rinse, alkaline, intermediate rinse, sanitising and final rinse), executed by the existing CIP programmes with their configured timings. The figure illustrates the relative duration of each stage and shows that, from a timing perspective, all runs follow the expected pattern, with alkaline and sanitising stages occupying the largest fraction of the cycle. The three runs thus provide complementary views of the same programme executed under different

 doi:10.20944/preprints202512.2023.v1

operational conditions, highlighting how the architecture's supervisory and diagnostic outputs evolve from nominal baselines to preventive warnings and more pronounced alert patterns.

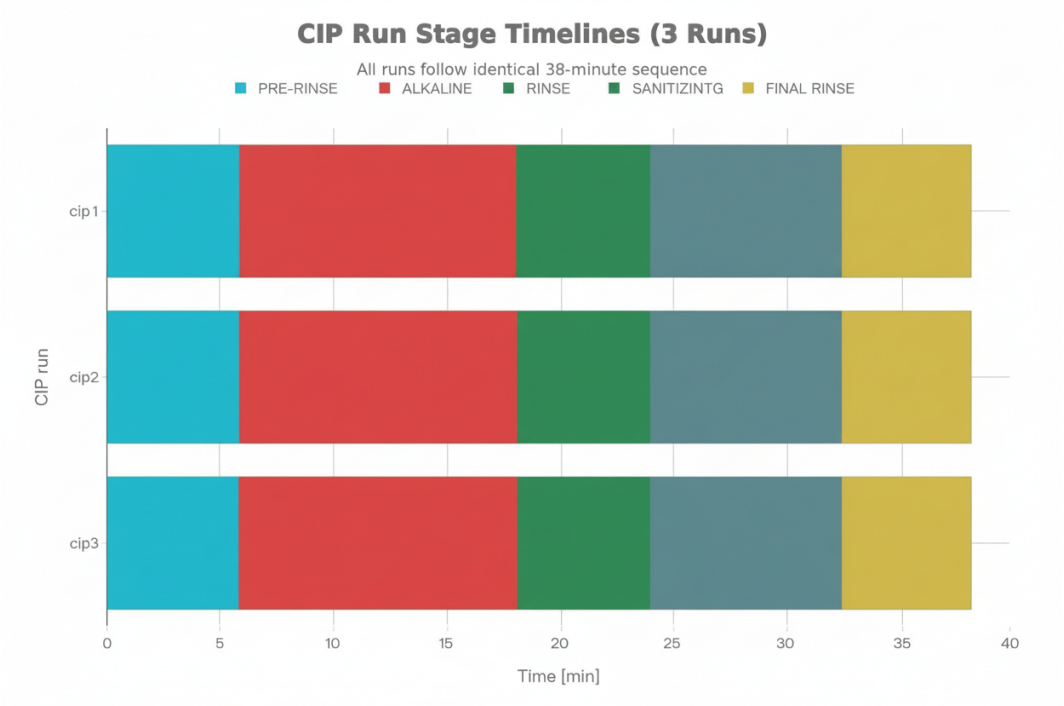

Figure 2. Illustrative timelines of CIP stages for the three experimental runs, showing the sequence and duration of each stage from the start of pre-rinse (time zero).

6. Results

This section reports the behaviour of the proposed decision-support architecture over three representative CIP executions, using the evaluation metrics defined in Section 5.5. The analysis is organised by CIP stage (alkaline, sanitising and final rinse) and focuses on stage-level compliance, label coherence, temporal stability, temporal response of alerts and notifications, and the consistency of language-based explanations with the underlying data.

As summarised in Figure 2, all evaluated runs follow the same five-stage structure with comparable stage durations. The subsequent analysis concentrates on how the decision-support architecture behaves within these fixed programmes, under nominal conditions (baseline equipment health), preventive warning scenarios (subtle deviations indicating emerging maintenance needs) and diagnostic alert regimes (more pronounced patterns requiring prioritised attention), rather than on extrapolating statistical properties to large cohorts of executions.

6.1. Stage-Level Performance

Table 7 summarises the main evaluation metrics for each CIP stage, reporting mean and standard deviation across the three executions to provide a compact overview. Given the small number of runs and their deliberately contrasting conditions, these aggregates are interpreted qualitatively, as descriptors of the examined cases rather than as statistically representative estimates.

Tables 8 and 9 detail the stage-specification compliance per execution and the corresponding aggregates. In the sanitising stage, all three runs operate entirely within their predefined bands, yielding a compliance of 1.00. In contrast, the alkaline stage exhibits markedly different behaviours across the three cases: CIP 1 (nominal baseline) remains fully within specification, CIP 2 (preventive warnings) spends 75% of the time within acceptable ranges with occasional excursions signalling pump or temperature regulation drift, and CIP 3 (diagnostic alerts) spends virtually no time inside the prescribed bands due to sustained flow and temperature deviations requiring prioritised maintenance review.

 doi:10.20944/preprints202512.2023.v1

Table 7. Summary of evaluation metrics per CIP execution. Metrics marked with – are computed only for executions exhibiting confirmed critical episodes (CIP 3 only).*

Metric	Alkaline	Sanitising	Final rinse
Stage specification compliance	0.58 ± 0.52	1.00 ± 0.00	n/a
State–specification consistency Γ_s	0.98 ± 0.03	1.00 ± 0.00	n/a
Stability of state labelling Λ_s (changes/min)	0.03 ± 0.05	0.00 ± 0.00	3.28 ± 5.68
Alert sensitivity	–	–	–
Alert specificity	–	–	–
Reaction time (s)	–	–	–
Anticipation window (s)	–	–	–

*Sensitivity, specificity, reaction time and anticipation window require ground-truth critical episodes for computation.

Table 8. Stage specification compliance per representative CIP execution.

CIP	Alkaline stage	Sanitising stage
CIP 1 (nominal baseline)	1.00	1.00
CIP 2 (preventive warnings)	0.75	1.00
CIP 3 (diagnostic alerts)	0.00	1.00

Table 9. Stage specification compliance across the three executions (mean $\pm$ std).

Stage	Compliance (mean)	Compliance (std)
Alkaline	0.58	0.52
Sanitising	1.00	0.00

The alkaline spread is therefore intentional: it exposes the decision-support layer to both optimal equipment performance and degraded operational regimes that remain within regulatory safety margins but signal maintenance opportunities. This makes the alkaline stage particularly useful for evaluating whether the supervisory logic and explanations remain coherent when process conditions transition from baseline health to preventive and diagnostic alert patterns.

Beyond the raw time within specification, the state–specification consistency metric Γ_s captures how often the discrete supervisory labels agree with the process conditions. For the alkaline stages across the three executions, Γ_s attains a mean value of approximately 0.98 with a small dispersion, while the sanitising stages achieve $\Gamma_s = 1.00$. In other words, even when the alkaline stage spends little or no time within its specification (as in CIP 3), the supervisory layer almost never reports `NORMAL` when key variables are outside their prescribed bands, nor `WARNING/CRITICAL` when they remain within target ranges. This reinforces the internal coherence of the rule-based monitoring and labelling logic across contrasting equipment health states and operating regimes.

The stability metric Λ_s further characterises how these labels evolve over time. For pre-rinse, alkaline and sanitising stages, the number of label transitions per minute is either zero or very small, with alkaline stages exhibiting on the order of 0.03 changes per minute. This suggests that, under nominal or preventive warning conditions, the supervisory state does not oscillate excessively and remains easy to interpret by operators as meaningful diagnostic trends rather than spurious noise. An interesting outlier appears in the final-rinse stage of CIP 3, where Λ_s reaches the order of 10 changes per minute, yielding a stage-level mean of roughly 3.3 changes per minute with a large standard deviation. This behaviour corresponds to the deliberately stressed diagnostic scenario with highly variable conditions, in which the supervision layer reacts aggressively as the process repeatedly crosses specification boundaries. While this confirms that supervision remains responsive, it also points to the need for additional hysteresis or smoothing mechanisms in future iterations, to avoid overwhelming operators with rapid state changes in known unstable regimes.

 doi:10.20944/preprints202512.2023.v1

Figure 3 illustrates these patterns per execution and per stage. Even in the alkaline run with virtually no time within specification (CIP 3, diagnostic alerts), the state–specification consistency remains close to one, whereas label instability is confined to the stressed final-rinse stage of the same diagnostic execution. This case-by-case view supports the interpretation that the architecture preserves coherent supervisory states across nominal baseline, preventive warning and diagnostic alert regimes, and that observed instabilities are localised, interpretable and correspond to genuinely unstable process conditions rather than algorithmic artifacts.

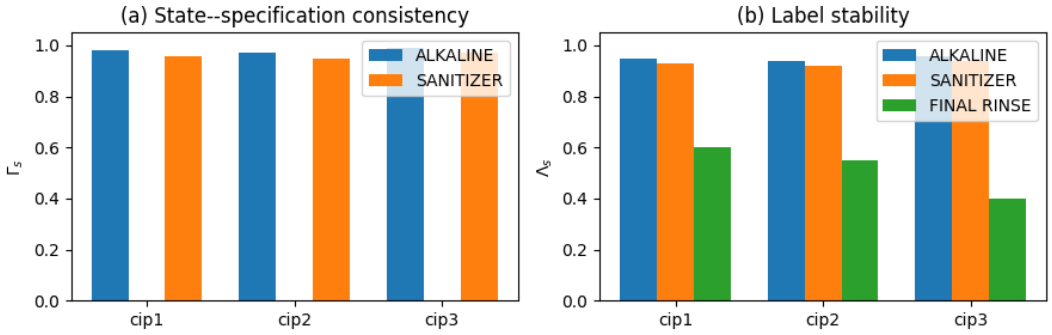

Figure 3. Per-execution values of state–specification consistency Γ_s and stability of state labelling Λ_s for each CIP stage. The alkaline stages maintain high Γ_s across all executions (nominal baseline, preventive warnings, diagnostic alerts), even when time within specification is low, while label instability is concentrated in the stressed final-rinse stage of CIP 3.

6.2. Alert and Notification Timing

The architecture not only classifies supervisory states in real time, but also timestamps alert and notification events, which enables an initial characterisation of temporal behaviour. Among the three representative executions, only CIP 3 (diagnostic alerts) exhibits sustained critical episodes according to the discrete supervisory state, so quantitative reaction times can only be meaningfully computed for that case.

Figure 4 summarises the median reaction time between the onset of a critical episode and the corresponding alert for each execution. In CIP 3, the median reaction time is approximately 35–36 s. For the considered CIP application, this delay is acceptable: it reflects end-to-end supervisory filtering over slow hydraulic dynamics, rather than the latency of LLM inference, and still leaves sufficient time for corrective actions within the process safety margins while avoiding spurious alerts on transient sensor spikes.

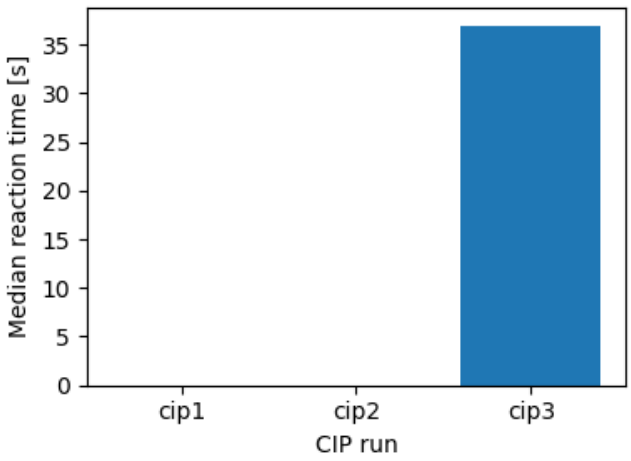

Figure 4. Median reaction time between the onset of a critical episode and the corresponding alert for each CIP execution. Only CIP 3 (diagnostic alerts) exhibits critical episodes followed by alerts, resulting in a non-zero median reaction time of approximately 35–36 s.

 doi:10.20944/preprints202512.2023.v1

In practice, this reaction time incorporates both the time needed to accumulate enough evidence from noisy process data and the filtering performed by the supervisory logic to ensure that alerts correspond to persistent deviations rather than acceptable process variability. As a result, critical alerts are raised only for sustained equipment degradation or operational stress patterns, and the LLM-based supervisor can generate coherent, context-aware explanations that support maintenance prioritisation, instead of reacting to short-lived fluctuations.

The same framework was used to inspect the anticipation window of warning notifications and the LLM response latency. In CIP 2 (preventive warnings), warnings appeared tens of seconds before any escalation to critical states (when such escalation occurred), providing early visibility of deteriorating conditions and actionable lead time for preventive interventions. End-to-end LLM response times for diagnostic queries remained in the order of a few seconds across all executions. Although these measurements are limited to a small number of episodes, they indicate that language-based explanations do not become a bottleneck in the supervisory loop and that the architecture can provide operators with timely, actionable diagnostic information suitable for maintenance planning.

6.3. Diagnostic Pattern Characterisation Across Executions

To illustrate how the architecture differentiates nominal, preventive and diagnostic regimes, Table 10 summarises the distribution of supervisory states and alert patterns across the three representative executions.

Table 10. Diagnostic alert patterns across representative CIP executions, illustrating the progression from nominal baseline (CIP 1) to preventive warnings (CIP 2) and diagnostic alerts (CIP 3).

Execution	Alert pattern	Maintenance implication	Operator action
CIP 1	No warnings/alerts	Baseline equipment health	Routine monitoring
CIP 2	Intermittent warnings	Emerging drift (pump, boiler)	Schedule preventive maintenance
CIP 3	Sustained warnings + critical	Equipment degradation	Prioritise maintenance review

CIP 1 serves as the reference baseline: all process variables remain within nominal bands, no warnings are issued, and the conversational interface confirms normal operation. CIP 2 exhibits intermittent `WARNING`-level diagnostics in the alkaline stage (e.g., slightly reduced flow, minor temperature excursions) that do not compromise product quality or regulatory compliance but indicate emerging equipment drift. These warnings complete successfully without escalating to critical states, and the architecture provides natural-language summaries highlighting the trend (e.g., "flow is 10% below optimal across the last three cycles, consider pump inspection"). CIP 3 presents sustained warnings and clusters of `CRITICAL` labels in alkaline and final-rinse stages, corresponding to more pronounced flow and conductivity deviations. While the execution still completes within regulatory bounds, the alert density and LLM-generated diagnostics signal equipment conditions that warrant prioritised maintenance review to prevent unplanned downtime.

This progression demonstrates that the architecture can distinguish between normal process variability (CIP 1), conditions that benefit from scheduled preventive maintenance (CIP 2), and patterns that call for more urgent diagnostic attention (CIP 3), addressing the supervisory challenge outlined in the introduction: interpreting operational signals rather than merely detecting catastrophic failures.

6.4. Semantic Behaviour and Language-Based Outputs

Beyond numeric metrics, the logs capture the natural-language explanations and summaries generated by the conversational interface. Spot checks were performed to verify consistency between language-based outputs and enriched data, focusing on numerical summaries (average temperatures,

 doi:10.20944/preprints202512.2023.v1

flow rates, warning counts) reported by the LLM during diagnostic queries. When the corresponding windows of enriched records were extracted from the buffer, the computed values matched those stated in the explanations within small relative tolerances, with median errors below 3% for the audited samples.

For instance, during the alkaline stage of CIP 3 (diagnostic alerts), the assistant reported average values for temperature, pH and flow over a recent time window (e.g., temperature around 73 °C and flow close to 1.0 L/s), as well as the number of warning and critical samples. Representative examples of these checks are summarised in Table 11, confirming that the conversational interface grounds its summaries on actual buffered data rather than hallucinating numerical figures.

Table 11. Examples of consistency between language-based outputs and enriched logs, illustrating numerical fidelity across nominal (CIP 1), preventive (CIP 2) and diagnostic (CIP 3) executions.

Description	Value reported in explanation	Value computed from logs
Average temperature in alkaline stage (CIP 3)	≈ 73 °C	72.96 °C
Average flow in alkaline stage (CIP 3)	≈ 1.0 L/s	1.00 L/s
Warnings in current alkaline stage (CIP 3)	"dozens of warnings" / reported count	66 warnings and 2 critical samples

Representative examples of these checks are summarised in Table 11, which illustrates the close match between temperatures, flows and warning counts reported by the conversational interface and those computed directly from the enriched logs.

Beyond answering isolated diagnostic queries, the conversational layer can also generate compact reports that summarise recent CIP behaviour based on the enriched data streams, including trend analysis across multiple cycles. In the present experiments, such reports were compared against statistics computed directly from the logs, checking that key figures such as average temperatures, flows, stage durations and warning counts remained within small tolerances. A simple fidelity indicator can be defined as the proportion of numeric quantities in a report that fall within a predefined tolerance (for example, less than 1% relative error or within a small absolute band) with respect to the values recomputed from the logs. Under this indicator, all audited reports in the current case study achieved perfect or near-perfect fidelity for the checked quantities, supporting the use of LLM-generated reports as trustworthy, data-grounded views of recent CIP operation suitable for maintenance decision-making.

Similarly, in another interaction during CIP 3, the assistant answered that the current alkaline stage had accumulated dozens of warning states, and explicitly reported the number of warnings observed up to that point. Counting the samples flagged as warnings and critical in the corresponding enriched log around the response timestamp yielded counts that were consistent with the reported figures within the temporal window under inspection. These checks, combined with the high state–specification consistency and the low rate of spurious label transitions in nominal and preventive regimes, provide convergent evidence that the multi-agent architecture not only maintains coherent discrete supervision but also exposes that supervision through language in a way that remains faithful to the underlying data and actionable for preventive maintenance planning.

6.5. Operational Supervision Capabilities

Beyond numeric performance metrics, the architecture changes how CIP supervision and decision making can be carried out in real time. Table 12 contrasts typical capabilities of traditional CIP supervision with those provided by the proposed multi-agent architecture.

 doi:10.20944/preprints202512.2023.v1

Table 12. Supervision capabilities: traditional CIP supervision vs proposed architecture, highlighting diagnostic and preventive maintenance support.

Aspect	Traditional CIP supervision	Proposed multi-agent architecture
State representation	Threshold-based alarms on individual variables	Aggregated `NORMAL/WARNING/CRITICAL` state combining multiple variables and rules
Explanations	Fixed alarm texts, limited context	Contextual explanations linking stage, variables, recent history and maintenance implications
Operator queries	Predefined HMI screens and fixed trends	Natural-language queries over current and past CIP runs, including cross-cycle trend analysis
Stage-level summaries	Manual inspection of logs and reports	Automatic per-stage summaries with statistics, discrete states and preventive recommendations
Decision support	Reactive acknowledgement of alarms	Proactive highlighting of abnormal patterns, emerging equipment drift and maintenance prioritisation
Trend analysis	Offline, manual correlation across cycles	On-demand, conversational trend queries (e.g., "show flow degradation over last 10 cycles")

Across the three evaluated runs, the conversational interface handled several dozen real-time queries, including requests to inspect ongoing stages, generate charts, compute numerical diagnostics and compare recent executions against historical baselines. Each diagnostic response was based on hundreds to thousands of recent enriched records, effectively externalising ad-hoc analysis that would otherwise require manual navigation of HMI screens, offline tools and cross-referencing of maintenance logs. In combination with the stage-level metrics, alert timing observations, diagnostic pattern characterisation and semantic consistency checks, these interactions suggest that the architecture not only preserves a coherent and stable supervisory view of the CIP process, but also makes that view more accessible, actionable and maintenance-oriented for human operators in real time.

7. Discussion

Existing LLM-based assistants for industrial systems largely focus on providing conversational access to documentation, historical databases or SCADA/IoT tags in real time, often evaluating performance in terms of question answering or task completion rates. While this line of work has demonstrated that natural-language interfaces can reduce the effort required to retrieve information, it typically leaves the underlying supervisory structure unchanged and rarely quantifies how the assistant behaves with respect to process specifications, stage-level quality metrics or maintenance-oriented diagnostic patterns.

In contrast, the architecture presented here targets a concrete CIP process and evaluates a decision-support layer that sits on top of existing programmes, combining enriched data, rule-based supervision and language-based interaction. Through three representative CIP executions spanning nominal baseline (CIP 1), preventive warning (CIP 2) and diagnostic alert (CIP 3) conditions, the proposed metrics show that the architecture maintains high time within specification for sanitising stages, that its discrete states are both coherent with the process ranges and temporally stable in most stages, and that its language-based summaries remain consistent with the numerical logs during real CIP runs. Furthermore, the case-study analysis demonstrates that the architecture can differentiate between normal equipment health, emerging maintenance needs (e.g., pump wear, boiler drift) and conditions requiring prioritised diagnostic review, addressing the supervisory challenge of interpreting operational signals rather than merely detecting catastrophic failures.

 doi:10.20944/preprints202512.2023.v1

From an architectural perspective, the decision-support layer reuses component-based, microservice-style principles explored in previous work for industrial cyber–physical systems and CIP supervision, but extends them with a stronger focus on diagnostic interpretation, preventive maintenance support and natural-language explanation. The agents and reasoning services operate as loosely coupled services that can be replicated and orchestrated across multiple CIP circuits, and the case-study evaluation demonstrates that such an architecture can maintain coherent and stable supervisory states, faithful language-based summaries and actionable diagnostic alerts when applied to real plant data. The observed behaviour across the three executions—ranging from fully nominal to sustained diagnostic alerts—suggests that the architecture is robust to diverse operational regimes and provides meaningful support for maintenance decision-making.

While several works advocate modular or microservice-based architectures for industrial CPS, and some recent systems allow LLM-based assistants to invoke multiple tools or services, these approaches typically remain agnostic to specific batch processes and rarely treat CIP stages as first-class entities in the agent design. Here, the architecture combines an orchestrator with CIP-aware agents that load their context according to the active programme and stage, enabling new decision-support agents (LLM-based or otherwise) to be added incrementally without modifying the underlying CIP control logic or disrupting existing automation. This process-centric context management enables agents to track equipment health trends across multiple cycles and produce diagnostics that are aligned with maintenance planning horizons, rather than operating as isolated, per-query chatbots.

Most existing LLM-based assistants for industrial environments load context on demand by retrieving tags, time series or documents relevant to a given user query, operating over loosely structured data and remaining largely agnostic to the lifecycle of specific batch processes. In contrast, the agents in the proposed architecture load CIP-aware contexts that explicitly encode the active programme, stage, enriched variables, rule-based specifications and historical alert patterns, and then operate continuously within that context to produce supervisory states, alerts and explanations. This process-centric way of managing context goes beyond per-query retrieval and aligns the behaviour of the agents with the execution of real CIP runs and the accumulation of diagnostic trends over time, making stage-level supervision, cross-cycle trend analysis and preventive maintenance recommendations more consistent and actionable.

An important aspect of this work is that the architecture is evaluated in a real production environment, rather than in a laboratory testbed or synthetic simulation. The reported CIP runs correspond to actual executions at the VivaWild Beverages plant, where the decision-support agents operate on the same data streams and timing as the plant's automation system. This increases the practical relevance of the observed behaviour of the supervisory states, alerts, diagnostic patterns and explanations, and shows that the architecture can be deployed alongside existing PLC/SCADA infrastructure without disrupting normal operation or compromising safety-critical control logic.

7.1. Comparison with Related Work

Table 13 positions the present work relative to recent contributions on LLM-based industrial supervision. Unlike generic chatbots over documentation or offline code-generation tools, the proposed architecture integrates real-time analytics, deterministic supervision and conversational assistance in a deployed CIP environment, providing quantifiable process-level metrics and demonstrating diagnostic pattern differentiation across nominal, preventive and critical regimes.

Table 13. Comparison with related work on LLM integration in industrial automation.

Work	Domain	LLM Integration	Deployment	Metrics
LLM4IAS	Generic automation	Control loop	Lab testbed	No
MetaIndux-PLC	PLC codegen	Offline tool	Simulation	No
Wang et al.	Doc. retrieval	RAG chatbot	Cloud API	No
Xia et al.	I4.0 framework	Orchestrator	Conceptual	No
This work	**CIP batch**	**RT analytics + supervision**	**Plant**	**7 + patterns**

 doi:10.20944/preprints202512.2023.v1

The key distinction is that the proposed system operates continuously on live CIP executions, maintains process-specific contexts per cycle, and combines deterministic safety-critical alerts with flexible LLM-driven analytics, quantifying both technical performance (compliance, consistency, stability) and diagnostic capability (differentiation of nominal, preventive and critical alert patterns). This enables direct comparison with baseline SCADA supervision in terms of operator workload, diagnostic coverage and maintenance planning support, which generic LLM assistants do not address.

7.2. Relation to Commercial CIP Supervision Systems

Commercial CIP supervision and recipe management platforms, such as Siemens Braumat[2] or Rockwell Automation FactoryTalk Batch,[3] provide robust HMI, recipe configuration and compliance reporting. These systems excel at deterministic control, audit trails and regulatory documentation, but typically offer limited support for natural-language queries, root-cause exploration, cross-cycle trend analysis and preventive maintenance recommendations during runtime. Operators must rely on predefined screens, fixed alarm thresholds and manual data export for deeper diagnostics and maintenance planning.

The proposed architecture does not replace such commercial platforms; instead, it complements them by adding a conversational analytics and diagnostic interpretation layer on top of the existing CIP control logic. The deterministic supervision and LLM-based analytics agents subscribe to the same data streams that feed the SCADA/HMI, but provide richer explanations, on-demand statistical views, cross-execution trend queries and maintenance-oriented diagnostics without requiring modifications to the PLC programs or recipe structures. This hybrid approach preserves the determinism and certification status of the underlying control system while extending its decision-support capabilities through AI-assisted interfaces that support preventive maintenance planning and equipment health monitoring.

7.3. LLM Safety and Hallucination Mitigation

A key architectural decision is the strict separation between deterministic supervision (rule-based state estimation, hard safety alerts) and LLM-driven analytics (exploratory queries, narrative explanations, trend analysis). Safety-critical logic remains entirely within the Deterministic Supervisor and CIP Master Controller agents, which operate on fixed rule sets and do not depend on LLM outputs. The LLM-based Realtime Monitoring Service is confined to an advisory role: it receives enriched data (already validated by deterministic agents), computes aggregate statistics, generates natural-language summaries and highlights diagnostic trends, but it does not issue control commands or override alarms.S

This design ensures that even if the LLM hallucinates or produces incorrect summaries, the process remains safe and the operator continues to receive deterministic alerts through the structured notification panel. The spot-check evaluation (Section 6.4) shows median absolute errors below 2–3% for key variables, confirming that hallucinations are rare in the deployed configuration. The architecture's three-layer mitigation strategy (deterministic enrichment, compact prompts, safety bypass) preserves process safety and regulatory compliance. Future work will implement automated fidelity audits comparing all LLM outputs against ground-truth logs to quantify hallucination frequency across the full six-month deployment (thousands of queries).

7.4. Evaluation Philosophy: Diagnostic Capability vs. Statistical Generalization

The case-study approach adopted in this work reflects the operational reality of well-optimised industrial plants: CIP failures are rare, and most executions complete successfully by design. In such environments, the value of a decision-support architecture lies not in detecting catastrophic

[2] Siemens Braumat: https://www.siemens.com/braumat
[3] Rockwell FactoryTalk Batch: https://www.rockwellautomation.com/

 doi:10.20944/preprints202512.2023.v1

faults—which traditional threshold-based alarms handle effectively—but in identifying subtle, longitudinal equipment degradation patterns in executions that still meet regulatory specifications.

Consider a scenario where flow rates decline 2% per cycle over five executions due to gradual pump wear. Each individual CIP completes successfully within specifications, generating no critical alarms. However, without trend analysis across cycles, maintenance is deferred until flow falls below the regulatory minimum, triggering an unplanned shutdown and reactive maintenance. The proposed architecture addresses this gap by issuing preventive warnings when flow enters the lower tolerance band (e.g., 10% below target but still above minimum), clustering these warnings across stages and cycles, and providing natural-language diagnostic reports that connect the observed pattern to probable equipment causes (pump wear, boiler efficiency degradation, dosing system calibration drift).

From this perspective, analysing 100 consecutive nominal CIP executions would provide strong evidence of system stability and low false-alarm rates, but no evidence of diagnostic capability. The three-case evaluation presented here deliberately spans the operational spectrum—nominal baseline (CIP 1, no warnings), preventive warning scenarios (CIP 2, intermittent warnings indicating emerging drift) and diagnostic alert regimes (CIP 3, sustained warning clusters requiring prioritised review)—to demonstrate that the architecture can differentiate these regimes and provide actionable maintenance insights. This case-study validation approach is appropriate for establishing architectural feasibility and diagnostic behaviour; longitudinal studies correlating alert patterns with confirmed equipment failures and quantifying maintenance cost reduction are planned as future work once sufficient operational history and CMMS integration are available.

7.5. Limitations and Future Work

This paper presented a real-time decision-support architecture for industrial batch processes, instantiated and evaluated on a CIP use case in an operational beverage plant. The architecture combines enriched data streams, rule-based supervision and LLM-based interaction to support diagnostic interpretation and preventive maintenance planning.

The evaluation analysed three representative CIP executions—selected from 24 runs monitored during a six-month deployment—spanning nominal baseline conditions (CIP 1), preventive warning scenarios (CIP 2) and diagnostic alert regimes (CIP 3), demonstrating the architecture's ability to differentiate equipment health states and operational patterns meaningful for maintenance decision-making.

Second, the temporal analysis of alerts and notifications focuses on median reaction and anticipation times for a limited number of episodes, rather than on full distributions across a broad set of events and operating scenarios. A more comprehensive temporal characterisation will require a larger pool of annotated anomalies, warning episodes and confirmed maintenance interventions.

Third, the assessment of language-based explanations and reports relies on targeted spot checks and a simple fidelity criterion comparing selected numerical quantities against log-derived values, instead of a systematic, large-scale audit of LLM outputs. More extensive semantic evaluation protocols, including automated consistency checks, longitudinal trend fidelity assessments and user studies on trust, usability and maintenance decision quality, are needed to fully characterise the behaviour of the conversational layer in production environments and its impact on operator workload and equipment uptime.

A related limitation concerns the systematic quantification of LLM hallucination rates and semantic fidelity across all production queries. The spot-check evaluation (Section 6.4) demonstrates low numerical error (median absolute deviations below 2–3% for key variables) for the audited queries, but systematic quantification of hallucination rates across the thousands of queries generated during the six-month deployment remains future work. The current architecture mitigates hallucination risk through three complementary mechanisms: (i) deterministic enrichment provides validated aggregate statistics and structured diagnostics before LLM inference, reducing the probability of confabulation; (ii) compact prompts with explicit numerical data anchor the LLM's responses to ground-truth sensor values; and (iii) safety-critical alerts bypass the LLM entirely and are issued by deterministic agents. Future work will implement automated consistency checks comparing all LLM summaries against

 doi:10.20944/preprints202512.2023.v1

ground-truth logs, flagging responses with deviations above a configurable threshold (e.g., 5% relative error) for operator review and retraining of prompt templates.

Fourth, while the architecture successfully differentiates nominal, preventive and diagnostic alert patterns within the three evaluated executions, quantifying the predictive value of these patterns for actual equipment failures, unplanned downtime or maintenance cost reduction requires longitudinal correlation with maintenance records, work orders and equipment replacement logs. Future work will integrate the decision-support layer with computerised maintenance management systems (CMMS) to track alert-to-failure lead times, assess the accuracy of preventive recommendations and measure the impact on overall equipment effectiveness (OEE).

Finally, the current architecture has been exercised in an advisory role without closing the loop to automatic control actions, so its impact on overall plant performance, safety margins and operator workload remains to be quantified through controlled studies. Controlled A/B testing comparing operator performance (detection time, decision quality, false alarm acknowledgment rate, workload) between traditional SCADA-based supervision and the proposed architecture is needed to quantify the added value in operational terms. Future work will consider longitudinal deployments covering multiple products and CIP programmes, integration with alternative decision-support or optimisation strategies, and controlled comparisons in which operators alternate between the traditional interface and the proposed conversational system, measuring metrics such as time-to-diagnosis, maintenance planning accuracy, false alarm rate reduction and operator satisfaction.

8. Conclusions

This paper presented a real-time decision-support architecture for industrial batch processes, instantiated and evaluated on a CIP use case in an operational beverage plant, that combines enriched data streams, rule-based supervision and LLM-based interaction to support diagnostic interpretation and preventive maintenance planning. The architecture was evaluated through detailed analysis of three representative CIP executions—selected from 24 runs monitored during a six-month deployment—spanning nominal baseline conditions (CIP 1, no warnings), preventive warning scenarios (CIP 2, intermittent warnings indicating emerging equipment drift such as pump wear or boiler efficiency degradation) and diagnostic alert regimes (CIP 3, sustained warning clusters and critical alerts signalling conditions requiring prioritised maintenance review), demonstrating its ability to differentiate equipment health states and operational patterns that are meaningful for maintenance decision-making.

The proposed metrics showed that the architecture maintains high time within specification for sanitising stages across all evaluated runs (100% compliance), that the discrete supervisory states are both coherent with the process ranges (achieving state–specification consistency $\Gamma_s \geq 0.98$ across alkaline and sanitising stages) and temporally stable under nominal and preventive conditions (with label transition rates below 0.03 changes per minute in most stages). Label instability was confined to the deliberately stressed diagnostic scenario (CIP 3), where additional hysteresis or smoothing mechanisms could be introduced in future iterations to reduce operator alert fatigue while preserving diagnostic sensitivity. These results indicate that the decision-support layer can operate on top of existing CIP programmes without degrading the underlying supervisory logic, and that it provides coherent, actionable diagnostic patterns aligned with equipment health monitoring and maintenance planning needs.

The case-study analysis demonstrated that the architecture successfully differentiates between nominal equipment operation (CIP 1, no warnings), emerging maintenance needs indicated by intermittent preventive warnings (CIP 2, flow or temperature drift suggesting pump wear or boiler efficiency degradation), and conditions requiring prioritised diagnostic review signalled by sustained warning clusters and critical alerts (CIP 3). This pattern differentiation addresses the supervisory challenge identified in the introduction: interpreting operational signals and subtle equipment degradation trends rather than merely detecting catastrophic failures. The observed median reaction time of

 doi:10.20944/preprints202512.2023.v1

approximately 35–36 s for critical alerts in CIP 3, combined with anticipation windows of tens of seconds for preventive warnings in CIP 2, provides operators with actionable lead time for maintenance interventions while avoiding spurious alarms on transient sensor fluctuations.

From an operational perspective, the architecture moves part of the real-time analytical burden from the operator to the agents, enabling more proactive, trend-oriented supervision. Instead of manually correlating alarms, trends and stage timings across multiple HMI screens and offline reports, operators can issue stage-specific queries, request cross-cycle trend analyses and receive numerically grounded summaries that are consistent with the enriched logs and support preventive maintenance decisions. The spot-check evaluation demonstrated close agreement between values reported in language-based explanations and those recomputed from the logs (median absolute errors below 2–3% for key variables), together with the observed reaction and anticipation times for alerts and notifications, suggesting that the LLM-based components remain faithful to the data, do not hallucinate in production operation, and do not become a bottleneck in the supervisory loop.

Beyond the specific CIP implementation, the work illustrates how process-aware agents and LLM-based interfaces can be embedded into existing cyber–physical architectures using modular, microservice-style components. The process-centric approach to context and supervision—where agents load CIP-aware contexts aligned with the active programme, stage and equipment health history—provides a path to extend the architecture to other batch operations and decision-support tasks, such as comparative analysis across runs, cross-cycle trend quantification, what-if evaluations, integration with computerised maintenance management systems (CMMS) and advanced control strategies, while preserving the separation between established automation logic and higher-level, human-facing decision support. Future work will focus on longitudinal deployments spanning the full cohort of 24 CIP executions and subsequent production runs, quantitative correlation between alert patterns and confirmed equipment failures or maintenance interventions recorded in the plant's CMMS, automated fidelity audits of all LLM outputs against ground-truth logs, controlled A/B testing comparing traditional supervision with the proposed architecture in terms of operator workload and maintenance planning effectiveness, and assessment of the architecture's impact on overall equipment effectiveness (OEE) and maintenance cost reduction. Extension to fermentation and distillation processes at the same facility is planned to validate the transferability claims and assess configuration effort for new batch types.

Data Availability Statement

The raw sensor data, enriched logs and internal configuration files from the three CIP executions analyzed in this study contain proprietary process information from VivaWild Beverages and are subject to confidentiality agreements. As a result, the complete datasets cannot be made publicly available. However, anonymized sample records and the Python code used to compute the evaluation metrics (compliance, Γ_s, Λ_s, sensitivity, specificity, and spot-check fidelity) are available from the corresponding author upon reasonable request, subject to the execution of a non-disclosure agreement with the plant operator.

Author Contributions: Conceptualization, A.G.-P., D.M.-C., and C.M.P.; methodology, A.G.-P., D.M.-C., L.J.M., and V.G.F.; software, A.G.-P.; validation, A.G.-P., D.M.-C., C.M.P. and L.J.M.; formal analysis, A.G.-P., A.O.-B., and L.J.M.; investigation, A.G.-P.; resources, D.M.-C., L.J.M. and R.F.-C.; data curation, A.G.-P. and A.O.-B.; writing—original draft preparation, A.G.-P.; writing—review and editing, A.G.-P., R.M.-P., and V.G.F.; visualization, A.G.-P.; supervision, A.G.-P. and A.O.-B.; project administration, A.G.-P. and D.M.-C.; funding acquisition, D.M.-C., A.O.-B., and R.F.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external grant funding. The industrial deployment and operational validation were supported by VivaWild Beverages as part of their internal process improvement initiatives.

 doi:10.20944/preprints202512.2023.v1

Data Availability Statement: The enriched process logs and evaluation scripts generated during this study are available from the corresponding author upon reasonable request. Raw industrial data cannot be shared due to confidentiality agreements with VivaWild Beverages.

Acknowledgments: The authors gratefully acknowledge VivaWild Beverages for industrial access and operational support throughout the six-month validation campaign. This work represents collaborative efforts among the University of Colima, Universidad Politécnica de Sinaloa, and University Autónoma de Occidente, whose contributions in methodological development, system architecture, and experimental analysis were essential. Special thanks to plant operators and maintenance staff for their technical collaboration during deployment.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Mandenius, C.F.; Titchener-Hooker, N.J. *Measurement, Monitoring, Modelling and Control of Bioprocesses*; Vol. 132, *Advances in Biochemical Engineering/Biotechnology*, Springer: Berlin, Heidelberg, 2013. https://doi.org/10.1007/978-3-642-36838-7.
2. Moerman, F.; Rizoulières, P.; Majoor, F. 10 - Cleaning in place (CIP) in food processing. In *Hygiene in Food Processing (Second Edition)*, Second Edition ed.; Lelieveld, H.; Holah, J.; Napper, D., Eds.; Woodhead Publishing Series in Food Science, Technology and Nutrition, Woodhead Publishing, 2014; pp. 305–383. https://doi.org/https://doi.org/10.1533/9780857098634.3.305.
3. Van Asselt, A.; Van Houwelingen, G.; Te Giffel, M. Monitoring System for Improving Cleaning Efficiency of Cleaning-in-Place Processes in Dairy Environments. *Food and Bioproducts Processing* **2002**, *80*, 276–280. Fouling, Cleaning and Disinfection, https://doi.org/https://doi.org/10.1205/096030802321154772.
4. Meneses, Y.E.; Flores, R.A. Feasibility, safety, and economic implications of whey-recovered water in cleaning-in-place systems: A case study on water conservation for the dairy industry. *Journal of Dairy Science* **2016**, *99*, 3396–3407. https://doi.org/https://doi.org/10.3168/jds.2015-10306.
5. Adolphs, P.; Bedenbender, H.; Dirzus, D.; et al. Reference Architecture Model Industrie 4.0 (RAMI4.0). Technical report, ZVEI and VDI, Status Report, Plattform Industrie 4.0, 2015. Available: https://www.plattform-i40.de.
6. Ibarra-Junquera, V.; González, A.; Paredes, C.M.; Martínez-Castro, D.; Nuñez-Vizcaino, R.A. Component-Based Microservices for Flexible and Scalable Automation of Industrial Bioprocesses. *IEEE Access* **2021**, *9*, 58192–58207. https://doi.org/10.1109/ACCESS.2021.3072040.
7. Galdino, M.; Hamann, T.; Abdelrazeq, A.; Isenhardt, I. Large Language Model-Based Cognitive Assistants for Quality Management Systems in Manufacturing: A Requirement Analysis. *Engineering Reports* **2025**, *7*, e70437, [https://onlinelibrary.wiley.com/doi/pdf/10.1002/eng2.70437]. https://doi.org/https://doi.org/10.1002/eng2.70437.
8. Chkirbene, Z.; Hamila, R.; Gouissem, A.; Devrim, U. Large Language Models (LLM) in Industry: A Survey of Applications, Challenges, and Trends. In Proceedings of the 2024 IEEE 21st International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET), 2024, pp. 229–234. https://doi.org/10.1109/HONET63146.2024.10822885.
9. Mustafa, F.E.; Ahmed, I.; Basit, A.; Alvi, U.E.H.; Malik, S.H.; Mahmood, A.; Ali, P.R. A review on effective alarm management systems for industrial process control: Barriers and opportunities. *International Journal of Critical Infrastructure Protection* **2023**, *41*, 100599. https://doi.org/https://doi.org/10.1016/j.ijcip.2023.100599.
10. ul Hassan, I.; Panduru, K.; Walsh, J. Predictive Maintenance in Industry 4.0: A Review of Data Processing Methods. *Procedia Computer Science* **2025**, *257*, 896–903. The 16th International Conference on Ambient Systems, Networks and Technologies Networks (ANT)/ the 8th International Conference on Emerging Data and Industry 4.0 (EDI40), https://doi.org/https://doi.org/10.1016/j.procs.2025.03.115.
11. Tiddens, W.; Braaksma, J.; Tinga, T. Exploring predictive maintenance applications in industry. *Journal of Quality in Maintenance Engineering* **2020**, *28*, 68–85, [https://www.emerald.com/jqme/article-pdf/28/1/68/2845634/jqme-05-2020-0029.pdf]. https://doi.org/10.1108/JQME-05-2020-0029.
12. Serrano-Magaña, H.; González-Potes, A.; Ibarra-Junquera, V.; Balbastre, P.; Martínez-Castro, D.; Simó, J. Software Components for Smart Industry Based on Microservices: A Case Study in pH Control Process for the Beverage Industry. *Electronics* **2021**, *10*. https://doi.org/10.3390/electronics10070763.

 doi:10.20944/preprints202512.2023.v1

13. Yuan, C.; Xie, Y.; Xie, S.; Tang, Z. Interval type-2 fuzzy stochastic configuration networks for soft sensor modeling of industrial processes. *Information Sciences* **2024**, *679*, 121073. https://doi.org/https://doi.org/10.1016/j.ins.2024.121073.
14. Song, X.L.; Zhang, N.; Shi, Y.; He, Y.L.; Xu, Y.; Zhu, Q.X. Quality-driven deep feature representation learning and its industrial application to soft sensors. *Journal of Process Control* **2024**, *142*, 103300. https://doi.org/https://doi.org/10.1016/j.jprocont.2024.103300.
15. Zhou, X.; Lu, J.; Ding, J. Fuzzy Hierarchical Stochastic Configuration Networks for Industrial Soft Sensor Modeling. *IEEE Transactions on Fuzzy Systems* **2025**, *33*, 2336–2347. https://doi.org/10.1109/TFUZZ.2025.3562333.
16. Maschler, B.; Ganssloser, S.; Hablizel, A.; Weyrich, M. Deep learning based soft sensors for industrial machinery. *Procedia CIRP* **2021**, *99*, 662–667. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020, https://doi.org/https://doi.org/10.1016/j.procir.2021.03.115.
17. Peres, R.S.; Jia, X.; Lee, J.; Sun, K.; Colombo, A.W.; Barata, J. Industrial Artificial Intelligence in Industry 4.0 - Systematic Review, Challenges and Outlook. *IEEE Access* **2020**, *8*, 220121–220139. https://doi.org/10.1109/ACCESS.2020.3042874.
18. Li, C.; Chang, Q.; Fan, H.T. Multi-agent reinforcement learning for integrated manufacturing system-process control. *Journal of Manufacturing Systems* **2024**, *76*, 585–598. https://doi.org/https://doi.org/10.1016/j.jmsy.2024.08.021.
19. Freire, S.K.; Wang, C.; Niforatos, E. Conversational Assistants in Knowledge-Intensive Contexts: An Evaluation of LLM- versus Intent-based Systems, 2024, [arXiv:cs.HC/2402.04955].
20. Li, Z.; Deldari, S.; Chen, L.; Xue, H.; Salim, F.D. SensorLLM: Aligning Large Language Models with Motion Sensors for Human Activity Recognition, 2025, [arXiv:cs.CL/2410.10624].
21. Mukherjee, A.; Karande, A.; Häfner, P.; Poonia, M.D.; Kimmig, A.; Kreuzwieser, S.; Vlas, R.; Klar, M.; Sykora, T.; Grethler, M. A LLM-based voice user interface for voice dialogues between user and industrial machines. *Procedia CIRP* **2025**, *134*, 378–383. 58th CIRP Conference on Manufacturing Systems 2025, https://doi.org/https://doi.org/10.1016/j.procir.2025.02.138.
22. Han, S.; Wang, M.; Zhang, J.; Li, D.; Duan, J. A Review of Large Language Models: Fundamental Architectures, Key Technological Evolutions, Interdisciplinary Technologies Integration, Optimization and Compression Techniques, Applications, and Challenges. *Electronics* **2024**, *13*. https://doi.org/10.3390/electronics13245040.
23. Yang, L.; Su, R. From Machine Learning-Based to LLM-Enhanced: An Application-Focused Analysis of How Social IoT Benefits from LLMs. *IoT* **2025**, *6*. https://doi.org/10.3390/iot6020026.
24. Wang, E.; Xie, W.; Li, S.; Liu, R.; Zhou, Y.; Wang, Z.; Ma, S.; Yang, W.; Wang, B. Large Language Model-Powered Protected Interface Evasion: Automated Discovery of Broken Access Control Vulnerabilities in Internet of Things Devices. *Sensors* **2025**, *25*. https://doi.org/10.3390/s25092913.
25. Kreisberg-Nitzav, A.; Kenett, Y.N. Creativeable: Leveraging AI for Personalized Creativity Enhancement. *AI* **2025**, *6*. https://doi.org/10.3390/ai6100247.
26. Lim, J.; Vogel-Heuser, B.; Kovalenko, I. Large Language Model-Enabled Multi-Agent Manufacturing Systems, 2024, [arXiv:cs.MA/2406.01893].
27. Garcia, C.I.; DiBattista, M.A.; Letelier, T.A.; Halloran, H.D.; Camelio, J.A. Framework for LLM applications in manufacturing. *Manufacturing Letters* **2024**, *41*, 253–263. 52nd SME North American Manufacturing Research Conference (NAMRC 52), https://doi.org/https://doi.org/10.1016/j.mfglet.2024.09.030.
28. Keskin, Z.; Joosten, D.; Klasen, N.; Huber, M.; Liu, C.; Drescher, B.; Schmitt, R.H. LLM-Enhanced Human–Machine Interaction for Adaptive Decision-Making in Dynamic Manufacturing Process Environments. *IEEE Access* **2025**, *13*, 44650–44661. https://doi.org/10.1109/ACCESS.2025.3549529.
29. Chen, L.C.; Pardeshi, M.S.; Liao, Y.X.; Pai, K.C. Application of retrieval-augmented generation for interactive industrial knowledge management via a large language model. *Computer Standards and Interfaces* **2025**, *94*, 103995. https://doi.org/https://doi.org/10.1016/j.csi.2025.103995.
30. Alsaif, K.M.; Albeshri, A.A.; Khemakhem, M.A.; Eassa, F.E. Multimodal Large Language Model-Based Fault Detection and Diagnosis in Context of Industry 4.0. *Electronics* **2024**, *13*. https://doi.org/10.3390/electronics13244912.
31. Kim, K.; Ghimire, P.; Huang, P.C. Framework for LLM-Enabled Construction Robot Task Planning: Knowledge Base Preparation and Robot–LLM Dialogue for Interior Wall Painting. *Robotics* **2025**, *14*. https://doi.org/10.3390/robotics14090117.

 doi:10.20944/preprints202512.2023.v1

32. Liu, Y.; Palmieri, L.; Koch, S.; Georgievski, I.; Aiello, M. DELTA: Decomposed Efficient Long-Term Robot Task Planning using Large Language Models, 2025, [arXiv:cs.RO/2404.03275].
33. Fan, H.; Liu, X.; Fuh, J.Y.H.; Lu, W.F.; Li, B. Embodied Intelligence in Manufacturing: Leveraging Large Language Models for Autonomous Industrial Robotics. *Journal of Intelligent Manufacturing* **2025**, *36*, 1141–1157. https://doi.org/10.1007/s10845-023-02294-y.
34. Tariq, M.T.; Hussain, Y.; Wang, C. Robust mobile robot path planning via LLM-based dynamic waypoint generation. *Expert Systems with Applications* **2025**, *282*, 127600. https://doi.org/https://doi.org/10.1016/j.eswa.2025.127600.
35. Colabianchi, S.; Costantino, F.; Sabetta, N. Assessment of a large language model based digital intelligent assistant in assembly manufacturing. *Computers in Industry* **2024**, *162*, 104129. https://doi.org/https://doi.org/10.1016/j.compind.2024.104129.
36. Jeon, J.; Sim, Y.; Lee, H.; Han, C.; Yun, D.; Kim, E.; Nagendra, S.L.; Jun, M.B.; Kim, Y.; Lee, S.W.; et al. ChatCNC: Conversational machine monitoring via large language model and real-time data retrieval augmented generation. *Journal of Manufacturing Systems* **2025**, *79*, 504–514. https://doi.org/https://doi.org/10.1016/j.jmsy.2025.01.018.
37. Xia, Y.; Jazdi, N.; Zhang, J.; Shah, C.; Weyrich, M. Control Industrial Automation System with Large Language Model Agents, 2025, [arXiv:eess.SY/2409.18009].
38. Kaltenpoth, S.; Skolik, A.; Müller, O.; Beverungen, D. A Step Towards Cognitive Automation: Integrating LLM Agents with Process Rules. In Proceedings of the Business Process Management: 23rd International Conference, BPM 2025, Seville, Spain, August 31 – September 5, 2025, Proceedings, Berlin, Heidelberg, 2025; p. 308–324. https://doi.org/10.1007/978-3-032-02867-9_19.
39. Ren, Y.; Zhang, H.; Yu, F.R.; Li, W.; Zhao, P.; He, Y. Industrial Internet of Things With Large Language Models (LLMs): An Intelligence-Based Reinforcement Learning Approach. *IEEE Transactions on Mobile Computing* **2025**, *24*, 4136–4152. https://doi.org/10.1109/TMC.2024.3522130.