Agentic AI Framework for Building Management Systems: Towards Intelligent and Autonomous Building Operations

Dissertation Course ID: DSECLZG628T

by

Prithviraj Acharya

(BITS ID: 2023DC04009)

Dissertation Work carried out at:

Honeywell Technology Solutions

Lab Pvt. Ltd Bangalore, India

Submitted in partial fulfillment of Data Science & Engineering degree program

Under the Supervision of

Eswara Manikya Lalitha

Honeywell Technology Solutions

Lab Pvt. Ltd, Bangalore

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE,
PILANI VIDYA VIHAR, PILANI, RAJASTHAN - 333031.

December, 2025

Contents

Abstract

Building Management Systems (BMS) are crucial for the efficient and reliable operation of Heating, Ventilation, and Air Conditioning (HVAC) systems in large commercial buildings. Modern HVAC setups produce a lot of operational data. However, finding and diagnosing faults in these systems still relies heavily on fixed rules and manual expert input. These approaches often lack flexibility, provide limited diagnostic insights, and can result in longer system downtimes and operational inefficiencies.

This dissertation introduces a Retrieval-Augmented Generation (RAG) framework for diagnosing faults in Building Management Systems. The framework offers a reasoning-driven, hierarchical retrieval method. It supports adaptive fault identification and explanation in complex HVAC settings and tackles significant weaknesses of traditional diagnostic techniques.

A simulation-based environment is used to generate consistent Air Handling Unit (AHU) telemetry under both normal and faulty conditions. Fault scenarios are shown as time-limited events and appear as observable patterns in telemetry data, without including explicit fault labels in individual data records. This setup allows for controlled tests while closely matching real-world diagnostic situations.

The framework combines structured telemetry analysis with contextual knowledge from equipment manuals and previous fault cases. By merging telemetry data with gathered domain knowledge, the system creates diagnostic explanations and suggests corrective actions. Performance is measured using metrics like diagnostic accuracy, clarity of the explanations generated, and time needed for diagnosis.

Overall, this work illustrates the use of a RAG-based approach to improve fault detection and resolution in Building Management Systems. This enables diagnostics that are more adaptable, clear, and grounded in knowledge, which helps enhance operational reliability and decrease system downtime.

Student :

Name: Prithviraj Acharya

Date: 19/12/2025

Place: Bangalore

Supervisor :

Name: Eswara Manikya Lalitha

Date: 19/12/2025

Place: Bangalore

Proposed System Architecture and Functional Modules

The proposed system uses a layered and modular design to support predictable experimentation, scalable stream processing, and understandable diagnostic reasoning for Building Management Systems. Each layer has a specific responsibility, which allows for gradual development and a clear distinction between data generation, processing, reasoning, and user interaction. This modular approach fits well with the phased implementation plan outlined in this dissertation.

The main functional modules of the system are described below, organized by architectural layer. The implementation status of each module shows the current progress at the mid-semester stage.

A. Simulation Layer

Behavioural AHU Telemetry Simulator
This module generates consistent telemetry for Air Handling Units (AHUs) operating in both normal and faulty conditions with predefined behavioural profiles. It allows for scenario configuration and specific fault episode injection, which enables controlled and repeatable testing over multiple runs. By producing consistent outputs for a given configuration, the simulator provides a reliable baseline for evaluating and comparing all downstream diagnostic layers.
Status: Completed

Telemetry Producer and Replay Module
This module replays generated telemetry as a time-ordered data stream. You can adjust the replay speeds to simulate real-time or faster operation. By simplifying the streaming process, the same replay logic works for both local testing and future use with production-level streaming systems. This guarantees consistent data delivery and reproducible input at all development stages.
Status: Completed

B. Stream Processing Layer

Telemetry Consumer Module
This module takes in telemetry events from the streaming layer and ensures reliable, ordered data consumption for downstream processing. It serves as a stable entry point into the processing pipeline, keeping later components separate from transport-level issues. This separation makes sure that the feature extraction and reasoning layers get a consistent data stream, no matter the underlying messaging system.
Status: Completed

Windowing and Feature Extraction Module

This module changes the continuous flow of raw telemetry into structured time windows. Instead of looking at individual data points, it computes statistical summaries and flags basic anomalies, such as sensors going beyond expected operating ranges, based on set threshold and pattern rules. Importantly, the module does not try to find the root fault. It identifies suspicious patterns and packages them into clear window summaries. This way, the reasoning layer gets a clear snapshot of system behaviour rather than being flooded with raw telemetry noise.

Status: Completed

C. Knowledge and Memory Layer

Static Knowledge Base Preparation Module

This module prepares manuals for AHU and HVAC equipment, along with summaries of past cases. It supports retrieval-augmented reasoning. The focus is on changing unstructured technical documents into a well-organized collection. This collection provides the specific background needed to understand how the system behaves. It ensures that the reasoning layer can access equipment standards and standard operating procedures (SOPs) during diagnostics.

Status: Completed

Vector Store and Semantic Retrieval Module

This module keeps vectorized representations of technical manuals and historical cases in a high-performance vector database. It provides semantic retrieval interfaces that enable the system to perform similarity-based lookups. During diagnostic reasoning, these functions help bring up relevant contextual information and equipment-specific guidance based on the symptoms observed in the telemetry windows.

Status: In Progress

Case-History Indexer (Dynamic Memory) Module

This module saves processed window summaries as searchable historical records. It creates a dynamic case history store. By allowing access to similar past situations, it helps improve and contextualize diagnostic reasoning over time. This function enables the system to identify recurring fault patterns and use past diagnostic results to enhance the accuracy of the current analysis.

Status: In Progress

D. Reasoning and Orchestration Layer

Controller and Retrieval Routing Module

This module provides the logic for the diagnostic workflow. It sets the order for retrieval, manages refinement policies, and coordinates interactions among telemetry summaries, knowledge retrieval, and the reasoning components. By serving as the system's logic hub, it ensures that retrieving domain knowledge is focused and efficient. This way, it gives the LLM the most relevant context for each specific fault scenario.

Status: Planned

LLM Reasoner (Hierarchical RAG) Module

This module performs structured, multi-stage diagnostic reasoning by combining evidence from technical manuals, past cases, and historical summaries. It produces clear diagnostic outputs, including likely root causes, confidence estimates, and detailed reasoning traces. In addition, it generates suggested corrective actions from the knowledge base, making sure that each diagnosis is practical and based on established engineering standards.

Status: Planned

E. Ticketing and User Interface Layer

Ticket Generator and Persistence Module

This module turns complex diagnostic outputs into structured tickets stored in a relational database. It supports a human-in-the-loop workflow, which lets facility managers review, approve, or reject suggested diagnoses and maintenance actions. By keeping these records, the module ensures a permanent, searchable audit trail of all interventions and system states.

Status: Planned

User Interface and Visualization Module

This module offers an interactive interface for looking at processed telemetry summaries, reviewing diagnostic analyses, and managing maintenance tickets. It aims to support decision-making by showing system behaviour, identified issues, and supporting evidence clearly and accessibly. By highlighting the reasoning and recommended actions, the module helps facility managers confirm the agent's insights before starting maintenance workflows.

Status: Planned

Functional Block Diagram and Data Flow

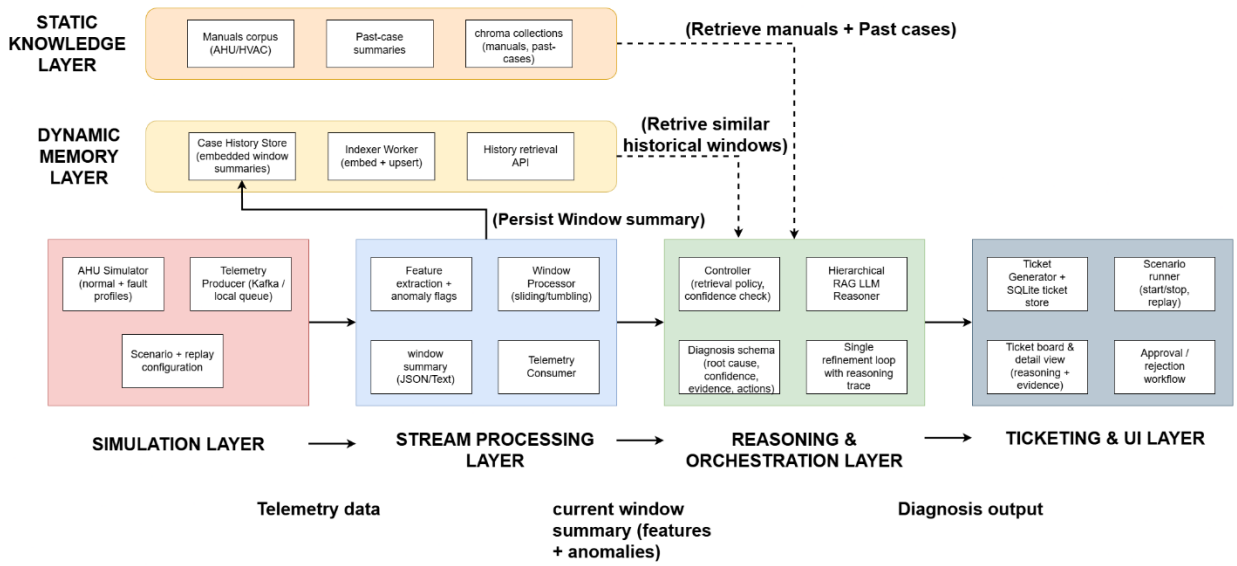

Figure 1

Figure 1 shows the functional block diagram of the proposed system. It illustrates the lifecycle of telemetry data as it goes through the simulation, processing, reasoning, and ticketing layers. A key design choice is converting raw telemetry into structured, window-level summaries before any reasoning happens. This choice ensures that the agentic layer works with clear, evidence-rich information instead of high-frequency raw noise.

End-to-End Data Flow - The system processes data through a sequence of eight distinct stages:

- **Scenario Configuration and Telemetry Generation:** A scenario definition sets the simulation parameters, including duration, sampling rate, and the schedule for fault episode injection. The AHU simulator produces time-stamped telemetry, like temperatures, valve positions, and fan speeds, that show both normal and faulty behaviors.

- **Streaming and Replay:** The telemetry producer replays generated telemetry as a time-ordered sequence of events. Replay speed can be configured to accelerate experiments while preserving timestamp order.

- **Telemetry Ingestion:** The consumer takes in telemetry events and sends them to the processing pipeline. This stage hides the transport method, keeping the downstream components separate from the specific messaging system used. This separation helps the system stay flexible as the project grows from local development to production-level streaming.

- **Windowing and Feature Summarization:** Incoming events are grouped into fixed time windows. For each window, we compute statistical features, such as means, variances, and trends, from selected telemetry signals like temperatures, valve positions, and fan speeds. This summarizes the system's behavior over the interval and reduces high-frequency noise into a manageable data structure.

- **Symptom-Level Anomaly Checks:** Rule-based checks are done at the window level to identify symptom indicators, like ongoing setpoint deviations or unusual actuator behavior. These checks point out abnormal patterns without trying to diagnose the root cause. This approach makes sure that raw evidence is kept for the reasoning layer.

- **Window Summary Generation:** Each processed window turns into a standardized "window summary" object. This summary includes structured data fields and a short textual description. It acts as the main input for all reasoning components that come after. This creates a consistent interface between the processing and AI layers.

- **Knowledge Retrieval and Reasoning:** The controller uses the identified symptom indicators and window context to manage the retrieval of relevant information from the knowledge base, such as equipment manuals and past case studies. The LLM-based reasoner then combines this evidence with the window summary to determine likely root causes, create detailed reasoning traces, and suggest corrective actions along with confidence estimates.

- **Ticketing and Review:** Diagnostic outputs are changed into structured maintenance tickets and saved in a relational database for traceability. The user interface allows users to review the analysis, the supporting evidence, and the suggested actions. This improves the review process and decision-making for human operators.

Separation of Responsibilities

The architecture separates symptom detection from fault diagnosis. The stream processing layer converts raw telemetry into summarized indicators in a clear and predictable way. Meanwhile, the agentic layer carries out the uncertain task of root-cause analysis. This division makes sure that diagnostic reasoning relies on evidence-based summaries and domain knowledge instead of raw data noise.

Technical Specifications and Data Schema

The following specifications define the architectural standards and data structures required for the system to function effectively. By decoupling simulation logic from diagnostic reasoning, the architecture ensures that raw telemetry is converted into a canonical, high-integrity format. This structured approach supports both modular development and the reproducible evaluation of complex fault behaviors.

System-Level Technical Specifications

The core technical specifications of the system are summarized in Table 1. These specifications define how telemetry is generated, processed, and consumed by downstream diagnostic components.

Parameter	Specification
Telemetry Source	Simulation-based AHU telemetry
Telemetry Mode	Replay-driven event sequence
Timestamp Semantics	Event-time based, strictly ordered
Determinism	Window-level summarization
Processing Granularity	Window-level summarization
Windowing Strategy	Fixed-duration tumbling/sliding windows
Anomaly Detection	Rule-based symptom identification
Diagnostic Method	Retrieval-augmented diagnostic reasoning
Knowledge Sources	Curated technical manuals and historical case summaries

Table 1

Telemetry Event Schema

A telemetry event shows the operational state of an AHU at a specific moment. The behavioral simulator generates telemetry events, which serve as the raw input for the stream processing layer. Each event has a timestamp marking the event time, an AHU identifier, and a consistent set of operational signals.

Importantly, telemetry events do not include fault labels. Fault conditions are indicated only by noticeable changes in signal behavior. This setup reflects real-world Building Management Systems (BMS), where faults must be deduced from operational data instead of being reported directly.

Telemetry Signal Categories

The telemetry generated by the simulator is organized into semantic signal categories reflecting standard HVAC control logic.

- Temperature signals represent thermal conditions across the AHU. These include outside air temperature (***oa_temp***), return air temperature (***ra_temp***), mixed air temperature (***ma_temp***), supply air temperature (**sa_temp**), and aggregated zone temperature (**avg_zone_temp**).
- Actuator and control signals represent control outputs and targets issued by the BMS. These include damper positions for outside air and return air (***oa_damper, ra_damper***), chilled water valve commands (***cc_valve***), and the supply air temperature setpoint (***sa_temp_sp***).
- Operational state signals describe equipment operation and context, such as supply air fan speed (***sa_fan_speed***), which provides insight into airflow and load conditions.

Fault Modeling and Injection

Faults are modeled as time-limited behavior episodes. This method allows us to evaluate diagnostic performance under realistic degradation patterns without needing a complete physics-based HVAC simulation. The system supports several fault types:

- **Cooling coil faults** that lessen heat exchange effectiveness.
- **Stuck damper faults** caused by mechanical failure of outside air or return air dampers.
- **zone temperature sensor drift,** which shows a gradual loss of measurement accuracy over time.

Each fault episode is defined by a start time, an end time, and a magnitude parameter ranging from 0.0 to 1.0 that controls fault severity. Ground truth fault labels are stored only in scenario metadata and are never exposed to downstream processing or reasoning components.

Window Summary as the Diagnostic Interface

The window summary serves as the main data contract between the telemetry pipeline and the reasoning engine. By converting high-frequency sensor streams into stable, interpretable snapshots, the system gives a structured "evidence packet" for downstream reasoning components to analyze:

- **Statistical Compression:** Instead of raw values, the summary provides statistical aggregates, such as mean, variance, and slope, that highlight trends and stability over a fixed interval.
- **Evidence-Based Symptoms:** The summary includes specific triggers, such as sustained control-response mismatches, along with the numerical evidence needed for the system to support its diagnosis.
- **Traceable Contracts:** Once emitted, these summaries cannot be changed, ensuring that every diagnostic output connects back to a consistent and auditable set of observations.

Determinism and Reproducibility

To ensure experimental rigor, the architecture is designed to produce identical diagnostic outcomes across repeated executions of the same scenario.

- **Data Integrity Tracking:** Each summary records sample_count and missing_count. This helps the reasoning layer consider incomplete or degraded input data when assessing confidence.
- **Deterministic Identity:** Window identifiers come from a combination of the AHU ID, timestamp, and window size. This ensures a stable identity for every event throughout the experiment's lifecycle.

Design Considerations

This section describes the main design ideas that influenced the system architecture. It focuses on experimental control, clear diagnostics, and the suitability of data for reasoning by agents.

Behavioral Simulation for Diagnostic Research

We chose a behavioral approach instead of detailed physics-based modeling. This choice helps us control the process and reliably introduce specific fault signs. Since the research centers on understanding and explaining faults instead of optimizing thermodynamics, behavior-driven patterns offer enough complexity while keeping the simulation lightweight and predictable.

Decoupling Symptom Detection from Root-Cause Diagnosis

The architecture maintains a clear separation between recognizing abnormal behavior and figuring out its cause. Deterministic rules at the stream-processing layer identify symptoms, such as stuck dampers or sensor noise, without labeling faults. This approach avoids early bias and ensures the agentic reasoner combines various symptoms with domain context before forming a conclusion.

Temporal Abstraction and Noise Reduction

To provide stable inputs, the system summarizes telemetry over fixed time periods instead of processing individual data points. This method filters out short-lived sensor noise while capturing important trends and statistical features. It guarantees that the reasoning engine works with high-integrity evidence packets rather than raw, noisy data streams.

Grounding Reasoning in Domain Knowledge

A key design goal is to reduce "hallucinations" in diagnostic results. Diagnostic reasoning relies on a structured knowledge base that comes from equipment manuals and historical cases. By using Retrieval-Augmented Generation (RAG), the system ensures that reasoning is based on verified domain information, not the internal biases of a language model.

Design for Scientific Reproducibility

We see reproducibility as a basic requirement. Every step, from scenario setup to feature summarization, is designed to yield the same outputs for a given seed. This consistency allows for thorough comparison between experiments and establishes a stable baseline for assessing the accuracy and reliability of the diagnostic agent.

Project Roadmap and Future Milestones

Phase	Timeline	Key Activities and Deliverables	Status
Phase 1: Problem Framing and Design	Nov 2025	Literature review on HVAC Fault Detection and Diagnostics (FDD), Agentic AI, and RAG methods. Definition of system objectives, scope, architecture, and end-to-end data flow.	**Completed**
Phase 2: Environment Setup and Simulation Development	Nov-25 – Dec -25	Development of a deterministic AHU telemetry simulator with fault episode injection. Implementation of telemetry producer, consumer, windowing, feature extraction, and anomaly rule pipeline. Preparation of mid-semester report and demo artifacts.	**Completed**
Mid-Semester Review Milestones	Dec-25	Submission of mid-semester report, plagiarism clearance, supervisor sign-off, and mid-semester viva demonstrating simulation and stream processing pipeline.	**Completed**
Phase 3: Knowledge Base and Retrieval Setup	Dec -25 – Jan-26	Preparation of HVAC manuals and synthetic past fault cases. Initialization of vector database and semantic retrieval interfaces. Refinement of anomaly thresholds and feature logic.	**In Progress**
Phase 4: Agentic Reasoning Framework	Jan-26	Implementation of controller logic, hierarchical retrieval routing, and LLM-based diagnostic reasoning with evidence provenance and confidence scoring.	**Planned**
Phase 5: Ticketing, Evaluation, and Finalization	Jan-26 – Feb-26	Generation of diagnostic tickets, system evaluation across multiple scenarios, documentation of results, dissertation writing, and viva preparation.	**Planned**

Abbreviations

Abbreviation	Description
AHU	Air Handling Unit
BMS	Building Management System
CC	Cooling Coil
FDD	Fault Detection and Diagnostics
HVAC	Heating, Ventilation, and Air Conditioning
LLM	Large Language Model
MA	Mixed Air
OA	Outside Air
RA	Return Air
RAG	Retrieval-Augmented Generation
SA	Supply Air
SOP	Standard Operating Procedure
SP	Setpoint
UI	User Interface