Generative AI for Anomaly Diagnosis in 5G NR Scheduling Logs

Hongyan Li

Department of Electrical and Information Technology
Lund University

Supervisors:
Johan Thunberg (LTH)
Ali Moradian (Ericsson)

Examiner: Michael Lentmaier

June 8, 2026

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Fifth-Generation New Radio (5G NR) base stations produce detailed scheduling logs that record per-slot decisions, channel measurements, and Hybrid Automatic Repeat Request (HARQ) feedback at sub-millisecond granularity. Diagnosing anomalies in these logs currently requires domain experts to spend approximately one hour of manual inspection per file, and no standardised tooling exists for automated Root Cause Analysis (RCA).

This thesis presents an end-to-end automated diagnostic system whose main contribution is the integration of statistical anomaly detection, correlation-based root-cause differentiation, and Generative Artificial Intelligence (GenAI) explanation into a single pipeline for 5G NR scheduling logs. Raw binary traces are parsed into a relational database, where statistical and Machine Learning (ML) methods detect anomalies at both the individual-record and temporal levels. A correlation-based module differentiates root causes that exhibit similar signatures, a rule engine maps findings to verified diagnostic patterns, and a GenAI agent powered by a Large Language Model (LLM) synthesises the collected evidence into readable diagnostic reports. A feedback mechanism allows engineers to confirm or correct diagnoses, accumulating a case store for future reference.

The system was evaluated as a proof-of-concept on 11 scheduling logs covering multiple distinct anomaly scenarios in a controlled laboratory environment. Compared with unassisted manual diagnosis, the system correctly classified all observed anomaly patterns, identified two engineer-confirmed failures, and uncovered subtle events that conventional threshold-based alarms missed. Diagnosis time dropped from approximately one hour to a few minutes per log. Although the evaluation is confined to a laboratory setting with a small sample size, the results confirm that combining ML-based detection with LLM-driven explanation is viable for operational 5G log analysis.

Popular Science Summary

Every time you stream a video, make a call, or scroll through social media on your phone, you are connected to a mobile base station, a tower with antennas that sends and receives radio signals. Inside the base station, a computer makes thousands of decisions every second: which users to serve, how fast to send their data, and how to react when signal conditions change. A new decision is made roughly every half-millisecond.

These decisions are all recorded in log files. A single file covering a few minutes of operation can contain hundreds of thousands of entries. When something goes wrong, such as a user's video freezing or speeds dropping without explanation, an engineer opens the log and manually traces the chain of events that led to the failure. In practice this means searching hundreds of thousands of lines for a handful of relevant ones, guided mostly by experience and intuition. It typically takes about an hour per file.

This thesis develops a system that automates that process in three stages.

First, the raw log data is parsed and organised into a structured format that can be queried efficiently, turning dense binary records into clean, searchable tables.

Second, statistical methods and machine learning are applied to detect unusual patterns. Some methods flag individual values far outside the normal range. Others detect clusters of small anomalies occurring together within a short time window, events that look harmless alone but collectively indicate a real problem. A Bayesian Network checks whether relationships between measurements still hold: for instance, if signal quality is reported as excellent but data transmissions keep failing, something is wrong even though no single number looks abnormal by itself.

Third, once the system knows what is unusual and when it happened, a large language model, the same type of technology behind tools like ChatGPT, writes a diagnosis in plain language. It combines the statistical evidence with knowledge of how 5G systems behave and produces an explanation of what went wrong, why, and what the engineer might do about it.

The system was tested on eleven base-station logs from a controlled lab environment. It correctly identified all known problems, including subtle ones such as a brief one-second reconfiguration event or a momentary connection loss with a single user, that standard monitoring tools would never flag because they fall below alarm thresholds. In two cases where experienced engineers had already

confirmed the root cause, the system reached the same conclusion and provided more specific quantitative detail than the engineers had noted.

The full analysis, from raw data to written diagnosis, completed in a few minutes, compared to roughly one hour of manual work. Engineers can then spend their time verifying and fixing rather than searching.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACK	Acknowledgement
AI	Artificial Intelligence
AIOps	Artificial Intelligence for IT Operations
API	Application Programming Interface
BLER	Block Error Rate
BN	Bayesian Network
CB	Codebook-Based
CC	Component Carrier
CDF	Cumulative Distribution Function
CLI	Command-Line Interface
CPT	Conditional Probability Table
CQI	Channel Quality Indicator
CSI	Channel State Information
DAG	Directed Acyclic Graph
DL	Downlink
GenAI	Generative Artificial Intelligence
gNB	Next Generation NodeB
HARQ	Hybrid Automatic Repeat Request
IF	Isolation Forest
IpN	Interference plus Noise
KNN	K-Nearest Neighbours
KPI	Key Performance Indicator
L1	Layer 1 (Physical Layer)
L2	Layer 2 (MAC Layer)
LLM	Large Language Model
LOF	Local Outlier Factor
LSTM	Long Short-Term Memory
MAC	Medium Access Control
MAD	Median Absolute Deviation
MCS	Modulation and Coding Scheme
MIMO	Multiple-Input Multiple-Output
ML	Machine Learning
MU	Multi-User
NACK	Negative Acknowledgement

NLP	Natural Language Processing
NLOS	Non-Line-of-Sight
NR	New Radio
OLLA	Outer Loop Link Adaptation
PDSCH	Physical Downlink Shared Channel
PHY	Physical Layer
PMI	Precoding Matrix Indicator
PRB	Physical Resource Block
PUSCH	Physical Uplink Shared Channel
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAG	Retrieval-Augmented Generation
RAN	Radio Access Network
RAT	Reciprocity-Based Antenna Transmission
RCA	Root-Cause Analysis
RI	Rank Indicator
SINR	Signal-to-Interference-plus-Noise Ratio
SQL	Structured Query Language
SRS	Sounding Reference Signal
SU	Single-User
TBS	Transport Block Size
TDD	Time Division Duplex
TTIs	Transmission Time Intervals
UE	User Equipment
UL	Uplink

Chapter 1
Introduction

Recent advances in Generative Artificial Intelligence (GenAI), particularly Large Language Models (LLMs) built on the Transformer architecture, have opened new possibilities for automating network operations and troubleshooting in telecommunications [1, 2]. This thesis presents a log analysis framework for 5G New Radio (NR) base stations that combines statistical and machine learning anomaly detection with LLM-based reasoning to automate Root-Cause Analysis (RCA) [3]. The system parses raw logs, detects scheduling anomalies, and generates causal diagnoses through an interactive chat interface. By comparing this approach against a traditional manual workflow, the study evaluates improvements in root-cause accuracy and the reduction of engineer cognitive load, demonstrating the practical value of GenAI-assisted Artificial Intelligence for IT Operations (AIOps) [4] in mobile network operations.

1.1 Background

In Ericsson's product verification and network operations workflows, a large number of tests and monitoring tasks are executed daily. Some run automatically in continuous integration pipelines, others are initiated by engineers during field trials or live network monitoring. Every execution produces hundreds of megabytes of log data, capturing system states and decisions at millisecond resolution. When a failure or performance anomaly is observed, engineers must identify its root cause by cross-referencing the logs against specifications, configuration files, and domain knowledge. The process is time-consuming and relies heavily on individual experience.

This thesis focuses on one specific instance of this problem: analysing 5G NR base-station scheduling logs to detect anomalies and diagnose their root causes automatically.

1.2 Motivation

RCA in 5G base-station logs is still largely manual. In a 5G NR base station, the Medium Access Control (MAC) scheduler decides every slot (0.5–1 ms) which users to serve, what modulation and coding scheme to use, and how many resource

blocks to allocate. These decisions, together with the underlying Physical(PHY) layer measurements such as Signal-to-Interference-plus-Noise Ratio (SINR) and interference levels, are recorded in scheduling logs. A single file covers a few minutes of operation, producing hundreds of thousands of entries. Most reflect normal behaviour; an anomaly may occupy only a few hundred milliseconds of that window. Identifying the anomnaly requires the engineer to track relationships between multiple parameter (channel quality measurements, modulation settings, retransmission feedback, link adaptation adjustments) and cross-reference them against protocol specifications and system configuration. This process typically takes hours per log and the evaluation depends heavily on individual experience.

Some logs are unusable due to corrupted records, incomplete sessions, or misconfigured setups. Without automated screening, engineers only discover this after spending considerable time on the analysis.

The motivation for this work is to automate the repetitive parts of this workflow: parsing, statistical screening, and pattern matching. The system should deliver a pre-filtered set of anomalies with a suggested root cause, or confirm that the log contains nothing actionable. The goal is not to replace the engineer but to eliminate wasted effort and reduce the time from log collection to diagnosis.

1.3 Goals

The goal of this thesis is to build and evaluate a system that automates the most time-consuming parts of log-based root-cause analysis for 5G NR base stations. Specifically, the work addresses four objectives:

1. Detect anomalies in scheduling logs automatically, including subtle deviations that do not trigger conventional threshold alarms.
2. Identify the temporal location of failures within logs that may contain hundreds of thousands of routine records.
3. Generate root-cause diagnoses by combining statistical evidence with domain knowledge, and present them through an interactive chat interface.
4. Evaluate whether the system reduces diagnosis time and improves root-cause accuracy compared to manual analysis.

1.4 Thesis Structure

The remainder of this thesis is organised as follows. Chapter 2 introduces related wireless domain knowledge, describes the operational scenarios and potential anomalies that arise in practice, and reviews methods for log-based anomaly detection including LLM-based diagnosis. Chapter 3 details the research methodology, including log parsing, parameter family structure, statistical and machine learning detection methods, and LLM-based diagnosis. Chapter 4 describes the system implementation, covering the automated pipeline, rule engine, ReAct agent architecture, and human feedback mechanism. Chapter 5 presents the evaluation

results, comparing diagnosis time and root-cause accuracy against a manual workflow. Chapter 6 concludes the thesis, summarising the contributions and suggesting future research directions.

Chapter 2
Background

2.1 5G NR Downlink Scheduling and Adaptation

5G NR adapts its transmission parameters to match the current channel conditions [5, 6]. The following subsections introduce the mechanisms and terminology used throughout this work.

2.1.1 Base Station Protocol Stack

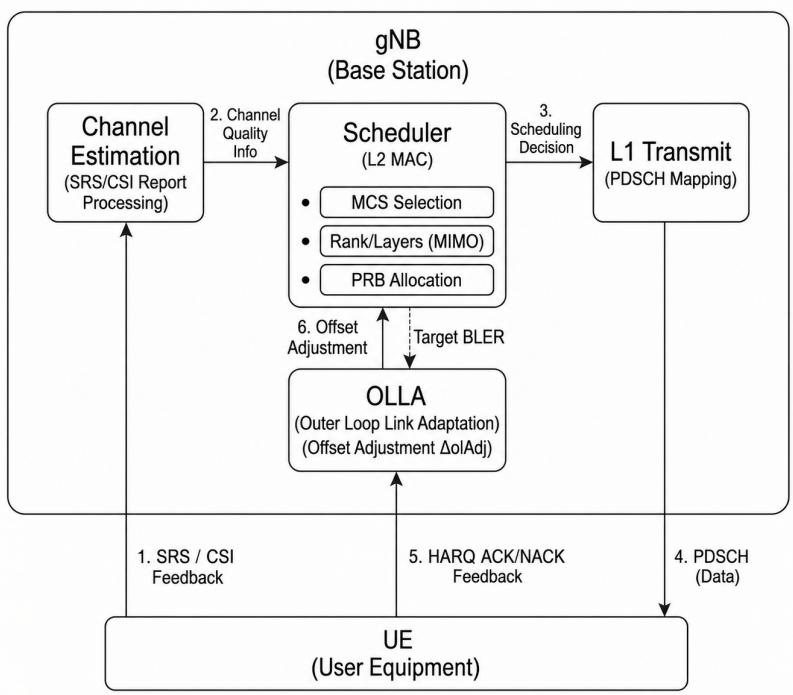

Figure 2.1: Closed-loop scheduling and link adaptation in 5G NR downlink.

As the central node in the 5G Radio Access Network (RAN), the Next Generation NodeB (gNB) serves one or more User Equipment (UE) devices within a cell and manages radio resource allocation and scheduling. Internally, it implements a layered protocol stack, of which the PHY layer and the MAC layer produce the highest log volume and are most directly involved in radio resource scheduling, and are therefore most relevant to this work. The PHY layer is responsible for signal transmission and reception, while the MAC layer handles resource allocation and scheduling. The interaction between these two layers proceeds as follows:

The gNB obtains channel information from two sources: Sounding Reference Signals (SRSs) transmitted by the UE for uplink (UL) channel estimation, and Channel State Information (CSI) feedback reported by the UE for downlink (DL) channel quality assessment. From these inputs, the channel estimation module derives quality indicators and passes them to the MAC scheduler, which selects the Modulation and Coding Scheme (MCS), Multiple-Input Multiple-Output (MIMO) rank, and Physical Resource Block (PRB) allocation each slot. The PHY layer then executes the MAC scheduling decisions by transmitting data on the hysical Downlink Shared Channel(PDSCH).

After receiving and decoding the datathe UE responds to gNB with a HARQ Acknowledgement (ACK) or Negative Acknowledgement (NACK). The Outer Loop Link Adaptation (OLLA) module tracks the resulting Block Error Rate (BLER) based on the ACK and NACK against a target (typically 10%) and computes a SINR adjustment offset, denoted as Δ_{olAdj}, which is fed back to the scheduler, correcting subsequent MCS selections. This prevents the scheduler from being persistently too aggressive or too conservative.

These components form a closed-loop control system as illustrated in Figure 2.1: channel measurements drive scheduling decisions, scheduling decisions determine transmissions, and transmission outcomes feed back through HARQ and OLLA to refine future decisions.

2.1.2 RAN Software Architecture and Trace Points

The protocol stack defines the communication rules; the RAN software architecture is where they are executed. In Ericsson's implementation, functional software modules contain trace points that expose internal states of the protocol layers during operation.

Figure 2.2 illustrates the gNB software architecture. The log traces analysed in this work originate from the Layer 1 (L1, Physical Layer) and Layer 2(L2, MAC layer) modules shown in the figure. Each module is further divided into numbered trace points (e.g. DRADLMDBFNR.73, UPCUEDLNR.172), each recording a specific sub -process at sub-millisecond granularity. These trace point classifications determine which event types appear in the logs and which parameter families are relevant for anomaly detection.

2.1.3 Cross-Layer Causal Dependencies

The parameters recorded by these trace points are not independent but form a causal chain across layers [6]:

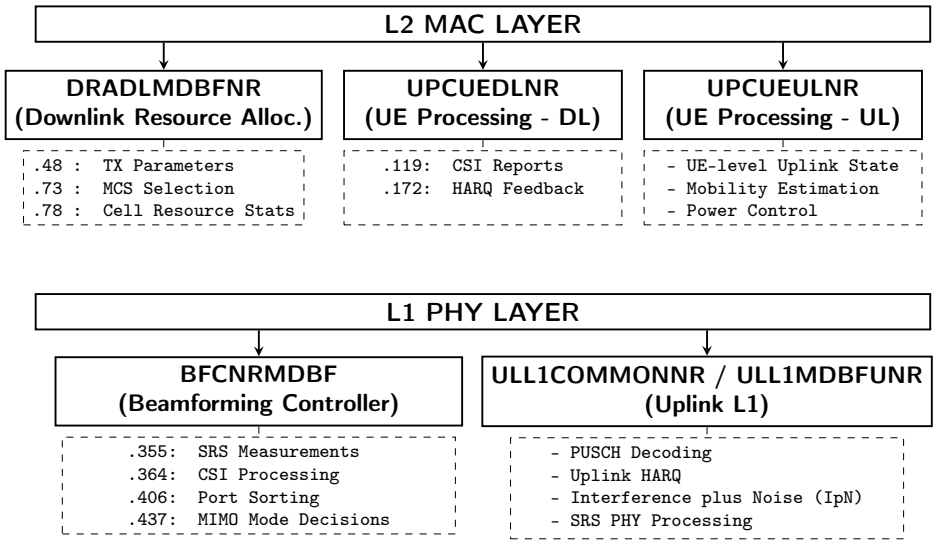

Figure 2.2: L1/L2 Module Hierarchy and Trace Point Mapping.

1. **Channel measurement.** SRS measurements (BFCNRMDBF.355) yield per-port SINR estimates that determine how many antenna ports are usable.
2. **Spatial configuration.** The number of usable ports bounds the achievable MIMO rank (BFCNRMDBF.437).
3. **Feedback correction.** HARQ ACK/NACK (UPCUEDLNR.172) drives OLLA, adjusting the SINR offset Δ_{olAdj} for MCS selection in subsequent TTIs (Transmission Time Intervals).
4. **Scheduling decision.** The scheduler (DRADLMDBFNR.73) combines the corrected SINR, rank constraint, and CSI to select the final MCS, layer count, and Transport Block Size (TBS).

A fault at any stage propagates downstream. An incorrect SRS measurement, for instance, may lead to a wrong rank estimate and an overly aggressive MCS, eventually causing repeated HARQ NACKs. The root cause and the visible symptom then sit in different trace points, separated by several TTIs. The remainder of this section details the parameters monitored at each stage.

Uplink Channel Estimation via SRS

The gNB estimates channel conditions from SRS transmitted by the UE on the uplink [7]. From SRS measurements, the gNB derives:

- **Per-port SINR**: channel quality estimate on each spatial dimension.
- **Port readiness count**: how many antenna ports have sufficient quality for spatial multiplexing.

- **Spatial channel coefficients**: used for precoder selection and rank determination.

In Time Division Duplex (TDD) systems, channel reciprocity allows the gNB to infer downlink channel properties from uplink SRS, enabling beamforming without explicit downlink CSI feedback [5].

Downlink Scheduling and MCS Selection

Based on channel estimates, the gNB scheduler determines three parameters for each transmission [6]:

- **Rank**: the number of spatial streams (1–4 in typical configurations), bounded by port availability and channel conditions.
- **MCS**: an index from a predefined table that specifies modulation order (QPSK, 16QAM, 64QAM, 256QAM) and code rate. Higher MCS gives better spectral efficiency but requires a stronger channel.
- **Resource allocation**: the number of PRBs assigned in the frequency domain.

The resulting TBS which is the number of information bits carried in a single transmission is determined by the combination of these three parameters:

$$\text{Throughput} \propto N_{\text{PRB}} \times N_{\text{layers}} \times \text{SE(MCS)} \tag{2.1}$$

where N_{PRB} is the number of allocated PRBs, N_{layers} is the number of MIMO spatial layers, and SE(MCS) is the spectral efficiency corresponding to the selected MCS index.

HARQ Feedback and Outer Loop Link Adaptation

The transport block is transmitted on the PDSCH [7]. The UE attempts to decode it and reports the outcome via HARQ feedback:

- **ACK**: decoding succeeded.
- **NACK**: decoding failed; the gNB retransmits with incremental redundancy.

If the maximum number of retransmissions is reached without success, the transport block is lost.

OLLA keeps the BLER near a configured target [8]. It maintains an offset olAdj applied to the SINR estimate used for MCS selection:

$$\text{olAdj}_{n+1} = \text{olAdj}_n + \begin{cases} \Delta_{\text{up}} & \text{if ACK} \\ -\Delta_{\text{down}} & \text{if NACK} \end{cases} \tag{2.2}$$

where $\Delta_{\text{down}} = \Delta_{\text{up}} \cdot \frac{1-\text{BLER}_{\text{target}}}{\text{BLER}_{\text{target}}}$ ensures convergence to the configured target BLER under stationary conditions.

CSI Reporting

The UE also periodically reports its own view of the downlink channel through CSI reports [6]:

- **CQI** (Channel Quality Indicator): recommended MCS to achieve the target BLER.
- **RI** (Rank Indicator): recommended number of spatial layers.
- **PMI** (Precoding Matrix Indicator): recommended precoding matrix.

The scheduler may follow or override these recommendations based on its own measurements.

2.2 Operational Scenarios in 5G NR Scheduling

The scheduling mechanisms above apply to all NR deployments, but parameter dynamics and anomaly patterns differ by scenario. The scenarios used in this work are summarised in Table 2.1.

Table 2.1: Operational scenario classification.

Dimension	Category	Description
User density	Single-UE	One UE with all cell resources
	Multi-UE SU-MIMO	Multiple UEs via time/frequency multiplexing
	Multi-UE MU-MIMO	Multiple UEs on same resources via spatial separation
Beamforming	Codebook-based (CB)	UE reports PMI from predefined codebook
	Reciprocity-based (RAT)	gNB derives precoder from uplink SRS

2.3 Potential Anomalies in Link Adaptation

As described in Section 2.1.3, a disruption at any stage of the scheduling chain propagates downstream. This section categorises the anomalies that can arise along this chain.

2.3.1 event-triggered anomalies

Some anomalies are identifiable from a single record without temporal context:

- **Event-level failures**: discarded CSI reports or missing scheduling indices due to UE context loss.
- **Resource exhaustion**: all PRBs consumed or scheduling entity limit reached within a slot.
- **SRS scheduling failures**: uplink SRS resources not allocated, forcing reliance on stale channel estimates.

2.3.2 Time-Windowed Anomalies

These anomalies are defined not by individual values but by their concentration within a time window:

- **Interference bursts**: simultaneous IpN spikes across all antenna ports, lasting 200 ms to 1 s, causing abrupt SINR drops.
- **Retransmission exhaustion**: clusters of transport block losses after all redundancy versions are exhausted, indicating sustained channel degradation beyond OLLA compensation.
- **Dense NACK bursts**: elevated NACK rate within a sliding window, indicating rapid channel change or OLLA convergence failure.

2.3.3 Adaptation Mismatch Anomalies

These anomalies arise when the adaptation mechanism responds incorrectly to the actual channel state:

- **OLLA lag**: channel conditions change faster than OLLA can track, leaving the SINR offset anchored to prior conditions and resulting in overly conservative MCS selection.
- **Rank over-allocation**: more layers assigned than the channel supports, forcing MCS reduction to maintain BLER, yielding worse spectral efficiency than fewer layers at higher MCS.
- **OLLA drift**: sustained monotonic shift in `olAdj` without convergence, indicating a mismatch between the configured BLER target and actual operating conditions.

2.4 Methods for Log Anomaly Detection

2.4.1 Log Parsing Methods

Log parsing converts raw text into structured data that can be processed automatically. This section reviews existing approaches and their applicability to telecommunication trace logs.

Log parsing methods fall into three broad categories. Template mining approaches such as Drain [9], Spell [10], and LenMa [11] separate fixed tokens from variable tokens in free-text logs where the schema is unknown. When the log format is known and consistent, regex-based parsing extracts fields directly using regular expressions [12], requiring no training data but depending on prior format knowledge. More recently, learning-based methods use neural models or LLMs to infer log structure [13, 14], offering flexibility at the cost of non-determinism and the need for labelled examples.

2.4.2 Anomaly Detection Methods

This section reviews anomaly detection techniques applied to structured operational data.

Statistical Anomaly Detection

Statistical methods detect anomalies by comparing observed values against expected distributional properties. Common approaches include identifying individual outliers in continuous parameters and detecting abnormal concentration of discrete events over time.

Modified Z-score (MAD). The Z-score measures how many standard deviations an observation lies from the mean. However, it is sensitive to outliers since extreme values inflate the standard deviation. The modified Z-score addresses this by replacing the mean and standard deviation with the median and Median Absolute Deviation (MAD) [15]:

$$M_i = \frac{0.6745 \cdot (x_i - \tilde{x})}{\text{MAD}} \tag{2.3}$$

where $\tilde{x}$ is the median of the sequence, $\text{MAD} = \text{median}(|x_i - \tilde{x}|)$, and the constant 0.6745 is the 75th percentile of the standard normal distribution, making M_i comparable to a standard z-score. Values with $|M_i| > 3.5$ are typically considered anomalous.

Sliding Window Frequency Analysis. For discrete parameters, a single anomalous value may be normal (e.g. an occasional retransmission), but a high concentration within a short window is not. The method computes the local frequency of a value within a fixed window and compares it to the global baseline using a Z-test [16]. When the local Z-score exceeds a threshold, the window is flagged.

Isolation Forest

Isolation Forest [17] is a tree-based unsupervised anomaly detector. It assumes that anomalous points, being few and distinct, can be separated from normal data with fewer random partitions.

The algorithm builds an ensemble of isolation trees, each recursively splitting the feature space on a random feature and random split value. Anomalies occupy sparse regions and are isolated near the root, producing short path lengths. The anomaly score for a sample x in a dataset of n points is:

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}} \tag{2.4}$$

where $E[h(x)]$ is the average path length of x and $c(n) = 2\,H_{n-1} - \frac{2(n-1)}{n}$ normalises by the expected path length of an unsuccessful search in a binary search tree of size n. $H(\cdot)$ is the k-th harmonic number. Scores near 1 indicate anomalies; scores around 0.5 indicate normal points.

Several properties are worth noting: it is unsupervised and requires no labelled anomaly samples; it operates in high- dimensional spaces and captures

joint multi-parameter deviations; it scales as $O(n \log n)$; and it tolerates moderate contamination in the training set.

The limitation is that Isolation Forest identifies which records are anomalous but not why. Explaining the violated relationships requires a complementary method (Section 3.5.2).

Bayesian Networks

A Bayesian Network [18] is a directed acyclic graph (DAG) where nodes represent random variables and edges represent conditional dependencies. Each node has a Conditional Probability Table (CPT) describing its distribution given its parents. The joint probability factorises as:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P\big(X_i \mid \text{Parents}(X_i)\big) \tag{2.5}$$

CPTs are learned from data. Records with low joint probability represent parameter combinations that rarely occur during normal operation.

For anomaly detection, Bayesian Networks offer several useful properties. Their edges encode known domain relationships, so any violation is directly explainable. If the observed value of a node has very low probability given its parents, the corresponding relationship is considered violated. The model also captures the joint distribution, allowing it to detect cases where each parameter appears normal individually but the combination is implausible.

The network structure can be learned from data or specified manually. For systems with known parameter relationships such as link adaptation, expert-defined structure is more reliable: anomalous samples are rare, and data-driven structure learning may miss infrequent dependencies.

Rule Engines and Hybrid Approaches

Rule-based methods encode expert knowledge as deterministic conditions ("if condition then conclusion"). They are fully interpretable but limited to known patterns and struggle with complex multi-parameter interactions.

Hybrid approaches combine data-driven detection with rule engines [16]: statistical or ML methods flag potential anomalies (high recall), while rules and probabilistic models explain and validate them (high precision). This layered design is common in industrial systems where both discovery of unknown anomalies and interpretable conclusions are required.

Other Approaches

Other methods exist but are less suited to this problem:

- **Distance-based** (KNN, LOF [19]): computational cost scales poorly with the high-dimensional, heterogeneous features in scheduling logs.
- **Deep learning** (Autoencoders, LSTMs [20]): require large homogeneous training sets, offer limited interpretability, and add significant overhead.

- **Supervised classifiers** (Random Forest, XGBoost [16]): need labelled samples, which are rare and diverse in operational logs.

Limitations of Existing Methods

When applied to base station logs, these methods face several limitations. The same parameter shift can be a correct adaptation or a fault depending on context, but traditional methods cannot distinguish the two. Causal delays between root cause and symptom often exceed typical sliding window scope. Many parameters have multi-modal distributions under normal operation, invalidating single-baseline assumptions. Most critically, these methods flag anomalies but cannot explain them in terms that engineers can act on. These gaps motivate combining conventional detection with large language models to cover the full path from anomaly detection to interpretable root-cause diagnosis.

2.5 LLM-Based Diagnosis

This section reviews the use of LLMs for operational log analysis, covering retrieval-augmented generation, agentic reasoning, and human feedback mechanisms.

From Detection to Diagnosis

Anomaly detection methods produce structured output (scores, flagged records, violated conditions) but not explanations [16]. Turning these outputs into a diagnosis requires interpreting them against domain knowledge, correlating multiple findings, and suggesting next steps.

Before LLMs, this was done manually or through rigid expert systems. LLMs can automate the reasoning step by synthesising heterogeneous evidence into natural-language explanations [21].

Retrieval-Augmented Generation

General-purpose LLMs lack domain-specific knowledge needed for accurate diagnosis. Retrieval-Augmented Generation (RAG) [22] addresses this by appending relevant documents from an external knowledge base to the prompt at inference time.

For log diagnosis, the knowledge base may contain parameter semantics and causal relationships, previously confirmed diagnoses for similar patterns, as well as vendor documentation on known issues.

Grounding outputs in retrieved knowledge reduces LLM hallucination. Vector databases (e.g. ChromaDB, FAISS) store and retrieve documents by semantic similarity [23].

Agentic Reasoning

Rather than generating a single response, agentic frameworks let LLMs reason iteratively by calling external tools. The ReAct paradigm [24] interleaves reasoning with tool invocations in a loop:

1. Observe current analysis state.
2. Reason about what information is missing.
3. Call a tool to obtain it.
4. Incorporate the result and repeat.

Log diagnosis is exploratory: an initial finding may prompt further investigation of specific parameters or time windows. The agent decides which analyses to run based on intermediate results. Frameworks such as LangChain [25] provide tool registration, prompt management, and execution orchestration.

Human-in-the-Loop Feedback

LLM diagnoses are not always correct. Feedback mechanisms allow engineers to:

- Confirm or reject a diagnosis.
- Edit with corrections or additional context.
- Record solutions for confirmed patterns.

Confirmed cases can be embedded and stored for retrieval when similar anomalies appear later. The system improves as feedback accumulates, without retraining the model [26].

Limitations

Although LLM-based diagnosis shows considerable promise, deploying it in operational settings still poses several challenges. The most prominent is hallucination: the model may produce plausible yet factually incorrect explanations, particularly when confronted with rare patterns. The multiple tool calls required by agentic workflows introduce non- trivial latency, making the approach difficult to apply under strict real-time monitoring constraints. The model's inherent non-determinism also means that identical inputs can yield different outputs across runs, weakening reproducibility. On the knowledge side, general-purpose models inevitably lack vendor-specific domain expertise; retrieval-augmented generation can narrow this gap but cannot fully close it. Furthermore, because diagnoses take the form of free text, assessing their correctness resists automation and ultimately requires expert judgement. These limitations collectively motivate a hybrid architecture in which deterministic methods handle the detection stage to ensure reliability, while the LLM serves as an upper diagnostic layer providing the interpretability that rule-based systems cannot offer.

Chapter 3
Methodology

This chapter presents the design and implementation of the proposed anomaly detection and diagnosis framework. The system follows a layered architecture: log parsing produces structured data, which is then analysed by progressively more sophisticated detection methods, culminating in LLM-based diagnostic summarisation.

3.1 System Architecture Overview

The framework illustrated in Figure 3.1 comprises four functional layers that operate on a shared database. Later layers build on the findings of earlier stages while retaining direct access to the underlying data:

1. **Log Parsing Layer**: Transforms raw base station trace files into a relational database. Each trace point (e.g. DRADLMDBFNR.78) becomes a separate table with its own fixed schema.

2. **Statistical Detection Layer**: Applies event-triggered checks, spike detection, and sliding window frequency analysis to identify point and temporal anomalies.

3. **Machine Learning Detection Layer**: Employs Isolation Forest for multi-dimensional anomaly scoring, Bayesian Networks for conditional probability analysis, and cross-family correlation to trace causal dependencies across the scheduling chain.

4. **LLM Diagnosis Layer**: Uses a ReAct agent with retrieval-augmented generation to synthesise findings into causal diagnoses, supported by human feedback.

The layers are designed to be complementary: statistical methods provide high recall with low computational cost, machine learning methods capture complex multi-parameter interactions, and the LLM layer provides interpretability and actionable recommendations.

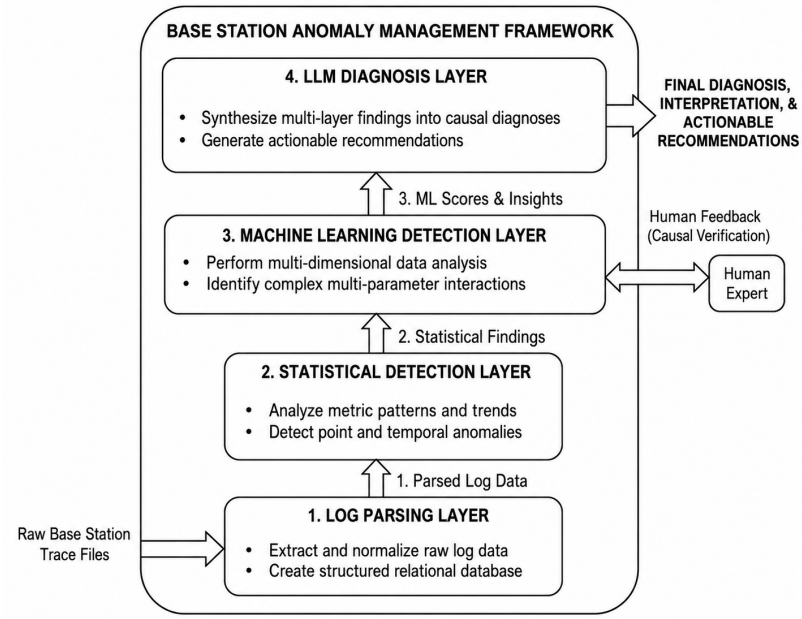

Figure 3.1: System overview.

3.2 Log Parsing and Data Preparation

3.2.1 Data Acquisition and Preparation

The logs utilized in this study originate from high-fidelity and real time Radio RAN scheduling logs captured from Ericsson NR baseband units. With the assistance of the Trace macro instruction embedded in the base station software, scheduling events are written to a circular binary trace buffer without interfering with system's real time performance. Then, acquisition tasks are performed for specific UEs and modules via the AMOS/CLI interface, exporting raw binary files. Finally, decoded by the internal LTNG decoder with platform-specific XML schemas (translation file), the raw binary traces are converted into a structured and semi-structured text format with microsecond-level wall-clock precision and synchronized radio timing states.

These logs mainly capture DL scheduling performance across different operational scenarios, including beamforming strategies, sector carrier configurations, user-density settings (Single-UE, SU-MIMO, and MU-MIMO), and varying channel conditions. Each record within the raw logs conforms to a standardized, semi-structured syntax characterized by the following four-part schema:

1. A timestamp in ISO format, enclosed in square brackets.
2. A hexadecimal unique identifier assigned to the specific software process.
3. A delimited event classification marker, enclosed in angle brackets, that maps the record to its originating software module and its corresponding

performance counter schema.

4. A serialized sequence of attribute-value pairs, utilizing diverse syntactical delimiters a to represent granular performance metrics or computed state variables.

Listing 3.1 shows a sanitized example of a decoded log record exhibiting this structure.

Listing 3.1: Sanitized log example

```
[2026-04-14 14:38:37.078810] 0xXXXXXXXX=(bfn:NNNN, sfn:
    NNN, sf:N.NN, bf:NNN) duId:N MODULE_A/TraceID 2
    COMPONENT_X.001 module_handler.c:1284: <!COMPONENT_X
    .001!> TRACE3
cellId=N, bbUeRef=0x******** sfn=NNN slot=N crnti=NNNNN:
     DL UE feedback. harqKey=NNNNN ccId=N
    harqFeedbackList=N harqSubprocess[0]={feedbackState=N
     txCount=N mcs=N rv=N tbsInBytes=NNNNN
    lastTxTimestamp=NNNNN ...}
```

All records sharing the same event type follow an invariant structural template: field names and their sequence are fixed, only the values change between entries.

3.2.2 Design principles for Parsing Framework

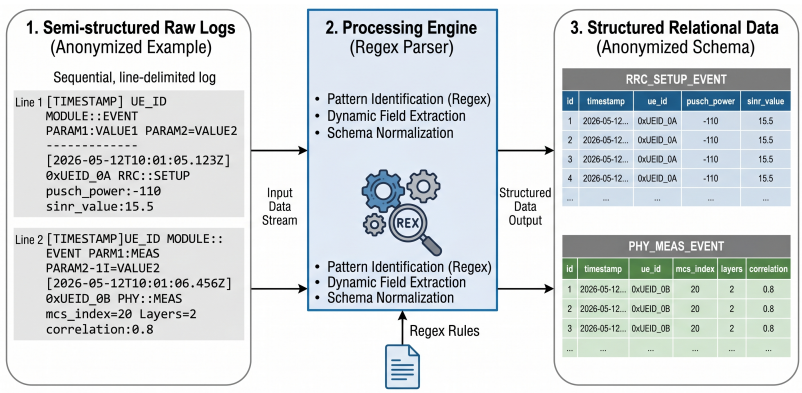

Figure 3.2: From raw text to structured text.

These characteristics make template mining approaches (e.g. LogAnomaly [27]) unnecessary, since the log structure is already fully specified by the vendor's format. Instead, regex-based extraction is sufficient to parse each record reliably [12]. Hence, this study proposes a regex-based parser for Ericsson base station logs that extracts all fields from each event type into relational database tables, as shown in Figure 3.2. This architecture has four practical advantages:

- **Deterministic correctness**: Each event type is identified by a unique marker and parsed with a dedicated regex, ensuring fields are never misassigned across event types.

- **Automatic table creation**: Database tables and columns are created on first encounter of each event type, requiring no manual schema definition.

- **Multi-rate alignment**: Retains the original microsecond timestamps so that records from tables with different sampling rates can be aligned in time.

- **Scalability**: Parses files in a single pass with batch database writes, keeping memory usage low for files of several hundred megabytes.

The output of the parsing stage is a relational database where each table corresponds to one event type, each row to one logged event instance, and each column to a named parameter. This structured representation serves as an input to all subsequent stages of anomaly detection and diagnostic analysis.

3.2.3 Parsing Algorithm

Algorithm 1 formalises the parsing procedure.

Algorithm 1 Single-pass log parsing

Require: Log file F, header regex R_h, event regex map $\mathcal{M}$
Ensure: Relational database $\mathcal{D}$

```
for each line l in F do
    (ts, marker) ← R_h.match(l)
    if marker ∉ M then
        skip l {Unknown event type}
    end if
    fields ← M[marker].match(l)
    if table for marker does not exist in D then
        Create table with columns from fields
    end if
    for each field f in fields do
        if column f does not exist then
            Add column f to table
        end if
    end for
    Insert (ts, fields) into table for marker
end for=0
```

3.3 Parameter Family Structure

A single log contains hundreds of parameters across multiple tables. Analysing them individually is impractical; grouping them by function allows the system to focus detection on the right subset and trace anomalies along causal paths.

We organise the key parameters into six families corresponding to functional stages of the scheduling pipeline described in Section 2.1.3: SRS SINR (channel quality), MIMO Mode (rank), BLER (HARQ/OLLA), CSI (UE feedback), Port (antenna readiness), and Scheduling (resource allocation).

Within each family, only dynamic parameters are subject to anomaly detection. Key indicators (e.g. `feedbackState`, `noOfLayers`) trigger the causal backtrace when anomalous; other parameters provide supporting context.

Families are connected by directed upstream relationships encoding causal precedence. For example, degraded SRS SINR may cause the scheduler to select a lower MIMO rank, which in turn reduces throughput and raises BLER. When a key indicator anomaly is detected, the system traces these links in reverse to identify the originating family. This structure serves two purposes: (1) it drives the automated backtrace in temporal anomaly detection, and (2) it provides causal context to the LLM-based diagnosis agent.

3.4 Statistical Detection Methods

3.4.1 Point Anomaly Detection

Point anomaly detection operates on individual log entries in isolation, without requiring temporal context from surrounding events. Each log line is evaluated independently, enabling real-time, per-event alerting with minimal computational overhead. This stage comprises two categories of checks.

Part A: Event-triggered Checks Certain event types (e.g. `BFCNRMDBF.658`, CSI discarded) are anomaly indicators by presence alone. Similarly, domain-knowledge thresholds flag values that should not occur under normal operation, such as SRS scheduling failures (`noOfFailedUeSrMrs` $\neq 0$) or scheduling entity limits (`reachMaxSe` $= 1$).

Part B: Data-Driven Checks For each numerical field, the MAD is computed over the entire sequence to distinguish fixed parameters from variable ones. A MAD $= 0$ indicates the parameter rarely changes and is likely a configuration or fixed setting; any value change is flagged as a configuration anomaly. A MAD > 0 identifies continuously varying parameters that require closer inspection; these are scanned using the modified Z-score (Equation 2.3), and points with $|M_i| > 10$ are flagged as spikes. This conservative threshold suppresses false positives in the initial broad scan, while the temporal analysis in Section 3.4.2 applies a lower threshold for root-cause identification. No predefined parameter list is required; anomalous fields are discovered automatically.

3.4.2 Temporal Anomaly detection

As noted in Section 2.1.3, root causes and symptoms often appear in different trace points separated by multiple TTIs. To capture these cascading failures, this section combines two analyses: a sliding-window frequency detector for temporal bursts, and a causal backtrace mechanism that traces symptoms upstream to their root cause.

Part A: Sliding-Window Frequency Analysis

Individual occurrences of certain discrete parameter values (e.g., a single NACK or a single QPSK modulation event) are expected under normal operation. However, a high density of such events within a short window indicates a transient anomaly. We detect these bursts as follows:

1. For each monitored discrete parameter, a domain-knowledge predicate defines which values constitute anomaly indicators (e.g., `feedbackState = 0` for NACK, `txCount` ≥ 4 for excessive retransmissions, `modulation = 1` for QPSK fallback).
2. A sliding window of width $W = 200\,\text{ms}$ moves through the event sequence in chronological order. For each window position, the local anomaly rate r_{local} is the number of events that satisfy the anomaly predicate divided by the total number of events in that window
3. The global baseline rate r_{global} is computed over the entire log (excluding a 3-second warmup period). A Z-score quantifies the deviation:

$$z = \frac{r_{\text{local}} - r_{\text{global}}}{\sqrt{r_{\text{global}}(1 - r_{\text{global}}) \,/\, n_{\text{window}}}} \tag{3.1}$$

   where n_{window} is the number of events in the window.
4. Windows with $z \geq 3.0$ are flagged. To avoid redundant alerts, only the highest-scoring window per 200 ms slot is retained.

The output of sliding-window frequency analysisis a set of temporally localised anomaly bursts, each annotated with the affected parameter, timestamp, local rate, and Z-score.

Part B: Causal Backtrace Analysis

This stage takes the anomaly bursts identified by Part A and maps each burst to its corresponding parameter family (Section 3.3). It follows a result-first strategy: starting from outcome-family anomalies, the system traces upstream through the causal graph to check whether upstream parameters are also anomalous in the same time region.

1. **Upstream Backtrace.** It follows a result-first strategy: starting from outcome-family anomalies, the system traces upstream through the causal graph (e.g. BLER ← MIMO ← Port ← SRS SINR). Within each flagged time window, the upstream parameters are divided into three categories

based on their statistical characteristics, and each category is paired with a detection method suited to its nature:

- Continuous parameters are checked with MAD(Section 3.4.1), using a lower threshold ($|M_i| > 3.5$) for root-cause sensitivity;
- Discrete event parameters are checked via the sliding-window frequency analysis (Section 3.4.2);
- Near-constant parameters flag any value outside the 1st–99th percentile range.

2. **Causal Chain Construction.** Anomalous upstream parameters are collected and sorted by timestamp to reconstruct the temporal propagation path:

   Root cause (family, parameter, deviation, time) → *intermediate effects* → *observed outcome*

   If no upstream anomaly is found, the system reports that the root cause lies outside the monitored parameter space (e.g., external interference or UE-side behaviour). The resulting causal chain is persisted to the database, where it serves as input evidence for the LLM-based diagnosis agent (Section 3.6).

3.5 Machine Learning Detection Methods

In a real channel environment, individual parameters may each fall within their normal range, yet their combination can reveal a mismatch between the scheduler's decision and the actual channel state. For example, moderate SINR paired with high rank and aggressive MCS is too optimistic for the actual channel conditions, leading to decoding failures and increased NACK rate.

To capture such multi-parameter anomalies, we analyse entire scheduling cycles rather than individual parameters. Isolation Forest identifies which cycles are overall outliers, and a Bayesian Network then diagnoses which parameter relationships caused the anomaly.

3.5.1 Isolation Forest Anomaly Detection

Feature Engineering

We build one feature vector per scheduling cycle by joining records from four tables. The base table is DRADLMDBFNR.73 (one row per TTI decision), which provides scheduling and resource allocation parameters. We enrich it with channel measurements, CSI feedback, and HARQ outcomes from three additional tables. Table 3.1 lists all features and their sources(see Section 2.1.3 for parameter definitions). `ack_rate` is derived from the HARQ feedback: for each initial transmission, it indicates whether the first attempt was successfully decoded.

Since retransmission-related anomalies (e.g. excessive NACKs) are already addressed by the upstream backtrace (Section 3.4.2), this stage focuses exclusively

Table 3.1: Feature sources and join strategy.

Source Table	Join Method	Features
DRADLMDBFNR.73	Base table	`mcs`,`noOfLayers`, `estimatedSinr`, `olAdj`, `allocatedDataInBytes`, `txCount`*
BFCNRMDBF.355	Nearest timestamp	`srs_sinr`
BFCNRMDBF.364	Nearest timestamp	`wbCqi`, `ri`
UPCUEDLNR.172	Key match (`harqKey`)	`ack_rate`

* Filtered out; see text.

on the scheduler's initial allocation decisions. Retransmissions (`txCount` > 1) use MCS and redundancy versions chosen by the HARQ logic rather than the scheduler, so they are filtered out. After filtering, `txCount` is uniformly 1 and carries no information, so it is dropped from the feature set.

Model Configuration and Output

The model trains unsupervised on the joint log record illustrated in Section 3.5.1 and computes the anomaly score (Equation 2.4) for every record. The contamination parameter is set to 5%, meaning the top 5% of records ranked by anomaly score are flagged as anomalous, roughly matching the anomaly rate observed in practice.

The model writes two tables to the database. The first, `if_anomalies`, stores one row per flagged record with its timestamp, all feature values from Section 3.5.1, and the anomaly score. The second, `if_findings`, compares each feature's median between the normal and anomalous subsets and records the percentage deviation (Table 3.2).

Table 3.2: `If_findings` table schema.

Field	**Description**
`feature`	Parameter name
`normal_median`	Median value in the normal subset
`anomaly_median`	Median value in the anomalous subset
`pct_diff`	Percentage deviation: $100 \times$ (anomaly median $-$ normal median)/|normal median|

These results show what is different but not why; both tables are passed to the rule engine and LLM agent for causal interpretation.

3.5.2 Bayesian Network Conditional Analysis

The Bayesian Network identifies which specific parameter relationships are violated in the flagged records.

Network Structure

To model the expected relationships between scheduling parameters, we construct a Bayesian Network whose edges mirror the scheduling closed loop described in Section 2.1.3 (Table 3.3), defined from domain knowledge rather than learned from data.

Table 3.3: Bayesian Network edges.

Parent	Child	Rationale
srs_sinr	wbCqi	UL measurement influences DL CQI
srs_sinr	estimatedSinr	SRS informs scheduler SINR
wbCqi	mcs	CQI guides MCS selection
wbCqi	ri	Channel quality affects rank
ri	noOfLayers	RI constrains layer count
estimatedSinr	mcs	SINR estimate drives MCS
mcs	ack_rate	MCS affects decoding success
noOfLayers	ack_rate	More layers, higher BLER risk
olAdj	mcs	OLLA offset corrects MCS

The real system has a feedback loop: ack_rate $\rightarrow$ olAdj $\rightarrow$ mcs $\rightarrow$ ack_rate. We cut the ack_rate $\rightarrow$ olAdj edge to keep the graph acyclic, since we model a single scheduling instant rather than the closed-loop behaviour over time.

Per Equation (2.5), each node's probability depends only on its parents. For `mcs`, which has three parents, the model learns:

$$P(\text{mcs} \mid \text{estimatedSinr},\ \text{wbCqi},\ \text{olAdj}) \tag{3.2}$$

A record whose joint probability is low under this model contains parameter combinations that seldom appear in normal operation.

Discretisation

Continuous features are binned at boundaries corresponding to physical transitions (Table 3.4). MCS bin edges align with modulation-order switching points defined in the 3GPP specification; SINR is partitioned into 5 dB intervals covering the typical operating regions observed in the dataset.

Model Fitting

Conditional probability tables are estimated exclusively from normal samples (i.e. records not flagged by Isolation Forest), so that the Bayesian Network captures the

Table 3.4: Discretisation bins for Bayesian Network nodes.

Parameter	Bins	Unit
`srs_sinr`, `estimatedSinr`	< 0 | 0–5 | 5–10 | 10–15 | > 15	dB
`mcs`	0–4 | 5–10 | 11–19 | 20–27	index
`wbCqi`	< 5 | 5–8 | 8–11 | > 11	index
`noOfLayers`, `ri`	1L | 2L | 3L | 4L	layers
`olAdj`	< -2 | -2–0 | 0–2 | 2–4 | > 4	dB
`ack_rate`	NACK | ACK	—

distribution of normal scheduling behaviour. To mitigate overfitting on sparse parent/child combinations, parameters are estimated with a BDeu (Bayesian Dirichlet equivalent uniform) prior, which distributes a virtual sample count uniformly across all parent/child state combinations. The equivalent sample size is set to 50, providing moderate smoothing without dominating the observed data in well-populated cells.

Conditional Violation Detection

For each parent/child pair in the DAG, we cross-tabulate $P(\text{child} \mid \text{parent})$ separately over the normal and anomalous subsets. If the probability in a cell differs by more than 15 percentage points between the two subsets, we flag it as a conditional violation. We keep the top 20 violations ranked by absolute difference.

For example, in normal records $P(\text{mcs} = \text{high} \mid \text{estimatedSinr} = \text{very\_high}) = 0.85$, but in anomalous records it drops to 0.30. In other words, 70% of anomalous scheduling cycles pick a conservative MCS even though the channel is good—the expected relationship is broken. The BN tells us *what* is wrong (MCS too low given the SINR), but not *why*. The Isolation Forest fills this gap by ranking feature contributions: if `olAdj` shows an abnormally negative value as the dominant anomalous feature, the combined conclusion is that OLLA over-corrected, dragging down the effective SINR estimate and causing the scheduler to under-select MCS.

Output

The model writes one table, `bn_conditionals`, to the database (Table 3.5). Each row represents one detected conditional violation.

This table is consumed by the rule engine and LLM agent alongside the Isolation Forest results.

3.5.3 Correlation-Based Differential Diagnosis

The Isolation Forest and Bayesian Network identify that an anomaly exists and which parameter relationships are violated, but they cannot distinguish between root causes that produce similar statistical signatures. For example, both a real channel degradation and an SRS measurement error cause MCS to drop while SRS

Table 3.5: `Bn_conditionals` table schema.

Field	Description
`child`	Child node name (e.g. `mcs`)
`parent`	Parent node name (e.g. `srs_sinr`)
`parent_value`	Discretised parent state (e.g. `very_high`)
`child_value`	Discretised child state (e.g. `high`)
`prob_normal`	P(child_value | parent_value) in normal subset
`prob_anomaly`	P(child_value | parent_value) in anomalous subset
`diff`	Probability difference (anomaly − normal)
`n_samples`	Number of anomalous records in this parent state

SINR changes. The difference lies in whether SRS and PUSCH move in the same or opposite directions. This module resolves such ambiguities using correlation analysis.

Correlation Metrics

The module computes five correlation coefficients from the parsed log tables:

SRS/PUSCH correlation: whether the channel estimate (SRS SINR) and actual reception (PUSCH post-EQ SINR) move together.

- Positive (>0.3): confirms a real channel change.
- Negative (<-0.2): indicates measurement mismatch.
- A separate computation on NACK-only slots verifies the mismatch persists during decoding failures.

Normalised port correlation: spatial consistency across antenna ports.

- Below 0.8: suggests degraded spatial resolution.

IpN port correlation: whether interference rises simultaneously on all ports.

- Above 0.7: points to a single external coherent source.

SINR/IpN correlation: whether SINR drops coincide with interference rises.

- Below -0.7: confirms interference as the cause.

SRS and PUSCH measurements are aligned in time using nearest-timestamp matching within a 100 ms tolerance window. The Pearson correlation coefficient is computed over the aligned series.

Output

The correlation metrics and pattern assignment are stored alongside the Isolation Forest and Bayesian Network results described earlier, completing the evidence set available to downstream components.

3.6 LLM-Based Diagnosis

The statistical and machine learning methods described above produce structured numerical evidence (feature deviations, conditional violations, causal chains) but do not directly yield human-interpretable explanations. A large language model serves as the final reasoning layer that consumes this evidence and synthesises a root-cause diagnosis.

3.6.1 Motivation

Traditional rule-based expert systems can map known patterns to conclusions, but they fail silently on novel or compound anomalies not covered by existing rules. Conversely, the statistical and ML outputs provide rich quantitative signals but require domain expertise to interpret. The LLM bridges this gap: it combines the structured evidence with injected domain knowledge to produce causal explanations, even for previously unseen anomaly combinations.

3.6.2 Input Evidence Structure

The LLM receives a fixed-format evidence package assembled from the database, containing:

1. **Isolation Forest findings**: per-feature median comparison between anomalous and normal subsets, with percentage deviations.
2. **Bayesian Network violations**: parent/child conditional probability differences exceeding the 15 percentage-point threshold.
3. **Statistical correlation metrics**: SRS/PUSCH correlation, normalised port correlation, IpN port correlation, and SINR/IpN correlation with threshold-based interpretations (Section 3.5.3).
4. **Causal chain**: the temporal propagation path constructed by the family-aware backtrace (Section 3.4.2).
5. **Rule engine conclusions**: any fired rules with their severity and pre-written explanations.
6. **Baseline status**: KPI health check results indicating which metrics are outside normal bounds.

3.6.3 Reasoning Strategy

The diagnosis follows a structured reasoning pattern:

$$\text{Root cause} \rightarrow \text{Causal chain} \rightarrow \text{Observable symptoms} \rightarrow \text{Recommended action}$$

The model is instructed to reason backwards from observed symptoms (e.g. elevated NACK rate) through intermediate effects (e.g. aggressive MCS selection)

to the originating root cause (e.g. SRS measurement mismatch). This mirrors the diagnostic reasoning process of experienced radio network engineers.

When multiple anomaly patterns co-exist in the same log, the model is required to distinguish independent root causes from shared ones, and to identify temporal ordering between them.

3.6.4 Scope and Limitations

The LLM is constrained to reason only over the provided evidence; it does not have direct access to raw log data. This design choice ensures that diagnoses are traceable to specific quantitative findings. However, it also means that anomalies not captured by the upstream detection methods will not appear in the diagnosis.

The detailed agent architecture, prompt design, knowledge injection strategy, and model selection are presented in Chapter 4.

Chapter 4

System Implementation

This chapter describes the engineering realisation of the diagnostic system. The statistical and machine learning methods presented in Chapter 3 form the analytical core; the components below orchestrate their execution, inject domain knowledge, and present results to the engineer.

4.1 System Overview

The system is implemented in Python and deployed as a Streamlit web application, which consists of five components:

1. A log parser that ingests decoded base-station event logs and stores structured records in an SQLite database.
2. An automated analysis pipeline that executes anomaly detection upon log loading.
3. A rule engine that matches detection outputs against expert-defined diagnostic patterns.
4. A ReAct-based LLM agent that provides an interactive chat interface for follow-up investigation.
5. A feedback store that persists engineer-confirmed diagnoses for future retrieval.

Upon loading a log file, the system executes the following stages without user intervention:

1. MAD-based statistical anomaly detection across all parameters.
2. KPI health check (e.g. BLER $< 10\%$, mean MCS > 10).
3. Isolation Forest and Bayesian Network analysis on per-scheduling records.
4. Correlation-based differential diagnosis (e.g. SRS/PUSCH direction).
5. Rule engine evaluation against all detection outputs.
6. LLM-based root-cause synthesis from the accumulated evidence.

All intermediate results are persisted to the database, enabling both the rule engine and the LLM agent to access them in subsequent steps.

4.2 Rule Engine

A deterministic rule engine complements the data-driven methods by encoding verified diagnostic patterns. Rules are defined in YAML and organised by root-cause category (e.g. MCS overestimation, OLLA instability, channel degradation). Each rule specifies:

- Conditions on Isolation Forest feature deviations (e.g. `MCS pct_diff > 10%`; see Table 3.2).
- Conditions on Bayesian Network conditional violations (e.g. $P(\text{MCS} = \text{high} \mid \text{estimatedSinr} = \text{low})$ differs by more than 10 percentage points; see Table 3.5).
- Conditions on correlation diagnosis patterns (e.g. SRS/PUSCH divergence confirmed).
- A severity level, causal conclusion, and recommended action.
- Supersession relations: when a more specific rule fires, it suppresses less specific ones sharing the same root cause.

The current deployment contains 27 rules (14 scheduling, 13 radio quality) covering the most common anomaly patterns identified during development. Engineers can add new rules at runtime via natural language through the `add_rule` tool. To illustrate the rule structure, three representative examples are given below.

Rule 1: MCS over-aggressive (High severity). Fires when the BN detects $P(\text{mcs} = \text{high} \mid \text{estimatedSinr} = \text{low})$ differs by more than 10 percentage points and the IF shows MCS deviation exceeding 10%. Conclusion: the scheduler selects high MCS despite low SINR, overestimating channel quality.

Rule 2: MU-MIMO pairing instability (High severity). Fires when the MU unpaired ratio exceeds 10% and the IF shows srs_sinr +50%, estimatedSinr −15%, mcs −20%. Conclusion: SRS SINR recovers when unpaired but drops when paired; the scheduler oscillates between MU/SU modes.

Rule 3: Layer/Rank mismatch (Critical severity). Fires when the BN detects noOfLayers|ri difference exceeding 15 percentage points and the IF shows noOfLayers > 20% with ack_rate < 0.7. Conclusion: scheduled layers deviate from UE-reported RI, causing decode failures.

4.3 LLM Agent

This section describes the LLM-based reasoning component that synthesises detection results into root-cause diagnoses for engineer review.

4.3.1 Model Selection

Three candidate models were evaluated:

1. **Mistral-Nemo-CPI-3GPP**: a Mistral Nemo variant additionally trained on Ericsson CPI documentation and 3GPP specifications.

2. **Qwen2.5-32B-Instruct**: a general-purpose instruction-tuned model.

3. **DeepSeek-V3**: a general-purpose model with strong reasoning capability.

The domain-specific model hallucinated frequently: it cited non-existent parameters and fabricated causal relationships not present in the evidence. It also anchored on specification text rather than reasoning over the statistical evidence provided. General-purpose models were more reliable with explicit anomaly evidence as input, producing fewer hallucinations and more accurate diagnoses. The final configuration uses Qwen2.5-32B-Instruct for agent tool routing and DeepSeek-V3 for expert reasoning and report generation. Both models are served locally via an internal inference platform.

4.3.2 Knowledge and Evidence Input

The LLM does not rely solely on its pre-trained knowledge. The system prompt is augmented at runtime with domain knowledge documents covering the scheduling signal chain, parameter families, verified diagnosis patterns, detection rules, and agent skills (Table 4.1). These are maintained as Markdown and YAML files, allowing engineers to update them without retraining the model.

Table 4.1: Knowledge documents injected into the LLM prompt.

Category	Content
Signal chain & parameters	5G scheduling closed-loop explanation, parameter definitions and units
Parameter families	Family grouping, causal graph structure, upstream/downstream relations
Diagnosis patterns	Known anomaly patterns with causal mechanisms and examples
Detection rules	YAML rule definitions (conditions, conclusions, actions)
Agent skills	Tool inventory, intent-to-tool mapping, usage examples

When the `diagnosis` tool is invoked, the system additionally assembles all detection results from the database (Isolation Forest findings, Bayesian Network violations, correlation metrics, rule engine conclusions, and spike anomalies) into a plain-text evidence block. This block is appended to the prompt and sent to DeepSeek-V3 in one API call. The model reads all evidence at once and generates the diagnosis in a single completion.

4.3.3 ReAct Architecture

The ReAct [24] agent bridges the engineer and the detection pipeline through iterative reasoning. It operates in a Thought → Action → Observation loop: reason about the next step, invoke a tool, observe the output, repeat until a conclusion is reached.

Table 4.2: ReAct agent tool inventory.

Category	Tool	Description
Detection	scan	Knowledge and data driven detection
	scheduling_analysis	Isolation Forest + Bayesian Network
	baseline	KPI health check
Diagnosis	deep_scan	Causal backtrace with family grouping
	diagnosis	Root-cause synthesis from all evidence
	explain	Context-aware single-anomaly explanation
	case_diagnose	Few-shot diagnosis from historical cases
Data access	query	Natural-language-to-SQL
	lookup_record	Keyword search on raw log entries
	kpi	BLER and throughput computation
Output	report	Summary report generation
	save_case	Save diagnosis for future retrieval
Rule mgmt	add_rule / delete_rule	Natural-language rule CRUD

The agent allows up to 15 iterations per session with a 120-second timeout. Each step invokes exactly one tool. Table 4.2 lists the twelve available tools.

Tool Routing

Tool selection is driven by the system prompt, which contains:

1. A skill description and input specification for each tool.
2. Intent-to-tool mapping rules (e.g. "root cause / why" → `diagnosis`; "scheduling analysis" → `scheduling_analysis`).
3. Disambiguation rules for overlapping intents (e.g. simple record lookups bypass the LLM via `lookup_record`; aggregation queries go through `query`).

Execution Example

A typical session proceeds as follows:

```
Question:    why is MCS dropping?
Thought:     Possibly a scheduling anomaly. Run analysis.
Action:      scheduling_analysis
Observation: IF: mcs deviation -35%; BN: P(MCS=high|SINR=high)
```

```
             violated ...
Thought:     Anomaly confirmed. Synthesise diagnosis.
Action:      diagnosis
Observation: Root cause: OLLA over-compensation ...
Final Answer: MCS drop caused by OLLA over-compensation.
```

The agent chains tools as needed and stops as soon as the accumulated evidence answers the question.

4.4 Human Feedback and Case Retrieval

After the automated pipeline produces a diagnosis, the engineer validates it through a structured feedback form.

4.4.1 Feedback Collection

The engineer provides three inputs:

1. **Severity**: real issue, minor, needs investigation, or false positive.
2. **Agreement**: agree, partial edit, or full rewrite. For the latter two, the LLM diagnosis is pre-filled in an editable text area.
3. **Comment**: optional free text (e.g. known solution or additional context).

Upon submission, the system assembles a document from the feedback and the current analysis state: log identifier, rule engine conclusions, LLM diagnosis (original or edited), structured evidence (IF deviations, BN violations, OLLA status, baseline failures), and the engineer's annotations.

4.4.2 Vector Storage and Retrieval

The document is embedded and stored in a ChromaDB persistent vector database. When a new diagnosis is generated in a subsequent session, the system queries the store using the first 500 characters of the diagnosis as input and retrieves the 3 most similar confirmed cases. Each result is displayed with the source log name, date, engineer severity/agreement indicators, comment, and a truncated evidence excerpt.

Engineers use the retrieved cases to compare against the current diagnosis, spot recurring patterns, and check consistency with past judgements.

4.4.3 Incremental Knowledge Accumulation

Each confirmed or corrected case enriches the vector store. Over time the retrieved references become more relevant, reducing the time engineers spend validating diagnoses. The stored cases also provide a foundation for future extensions such as few-shot prompting or automated confidence scoring.

4.5 Implementation Details

subsectionTechnology Stack The full technology stack used in the implementation is summarised in Table 4.3.

Table 4.3: Technology stack.

Component	Technology
Web interface	Streamlit
Database	SQLite (log storage), ChromaDB (vector store)
ML framework	scikit-learn (Isolation Forest), pgmpy (Bayesian Network)
LLM orchestration	LangChain (ReAct agent, prompt management)
LLM inference	Local deployment via internal serving platform
Visualisation	Plotly

4.5.1 User Interface

The web interface is divided into two panels (Figure 4.1):

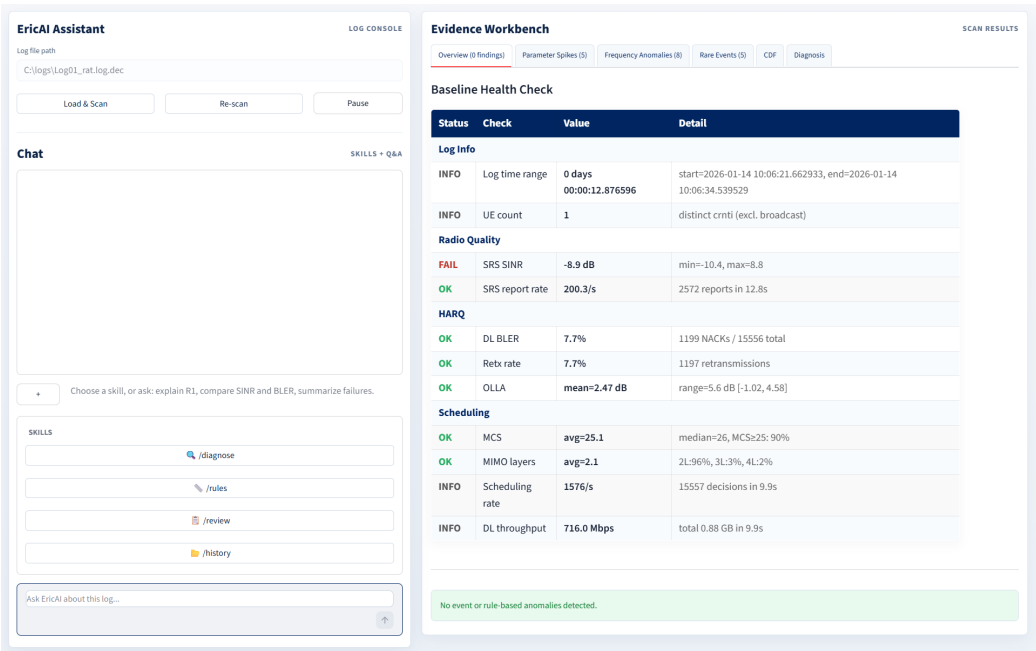

Figure 4.1: System dashboard with chat interface and six analysis tabs.

- **Left panel**: chat interface and log upload. Engineers type natural-language queries; the agent responds with analysis results. A feedback form below allows confirming or correcting diagnoses. As shown in Figure 4.2.
- **Right panel**: six tabs displaying pipeline results: Overview (baseline health, rule findings), Parameter Spikes, Frequency Anomalies, Rare Events, CDF comparisons, and Diagnosis (rule engine output). The Parameter Spike part is shown in Figure 4.3.

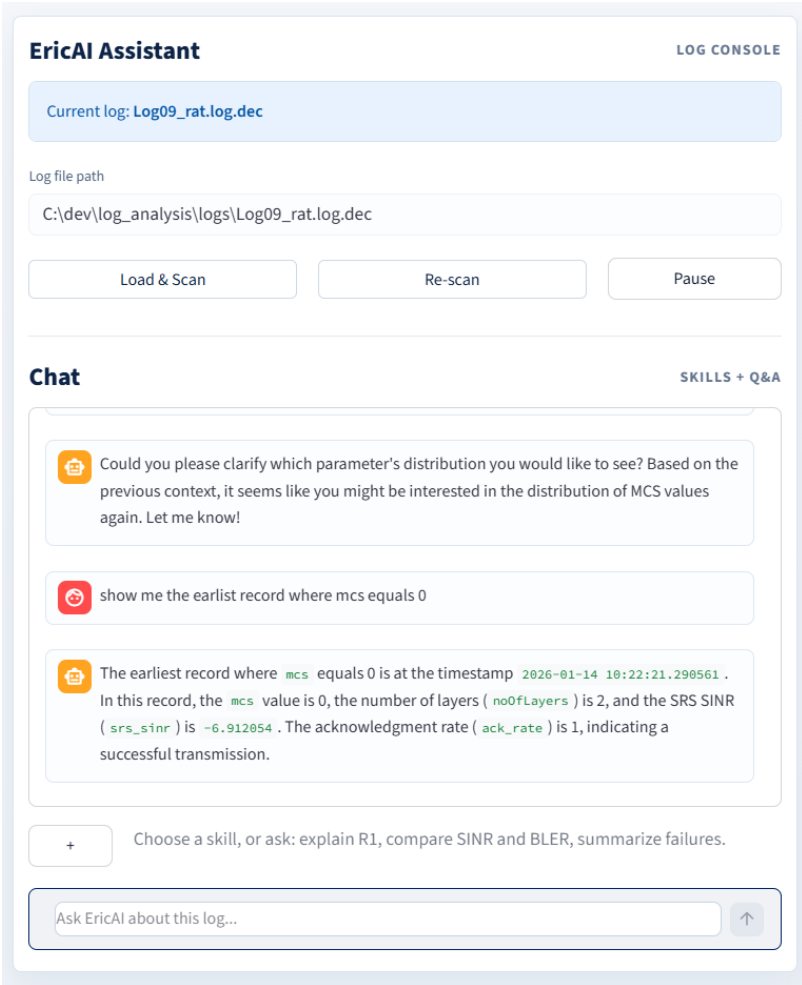

Figure 4.2: Left panel: log upload and interactive chat.

In the chat interface, engineers can use the Diagnose tab to generate an LLM-based RCA. The system produces a structured diagnostic report, including the identified root cause, causal chain, supporting evidence, and recommended actions, as shown in Figure 4.4. Engineers can review and refine the generated diagnosis

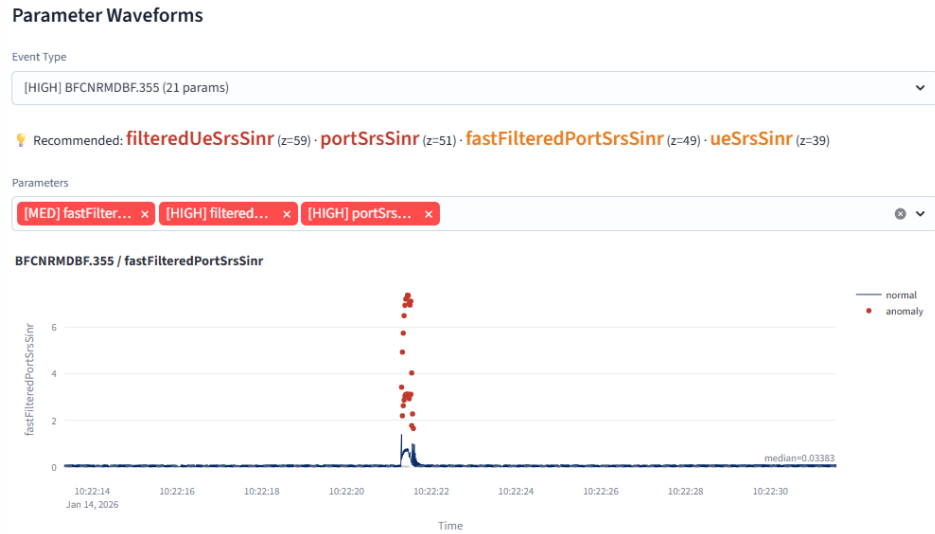

Figure 4.3: Right panel: parameter spikes with detected anomalies.

directly within the chat interface. Afterward, the Review tab can be used to generate a review draft. Once the draft has been validated and approved by the engineer, it is stored in the knowledge base for future reference and retrieval, as shown in Figure 4.5.

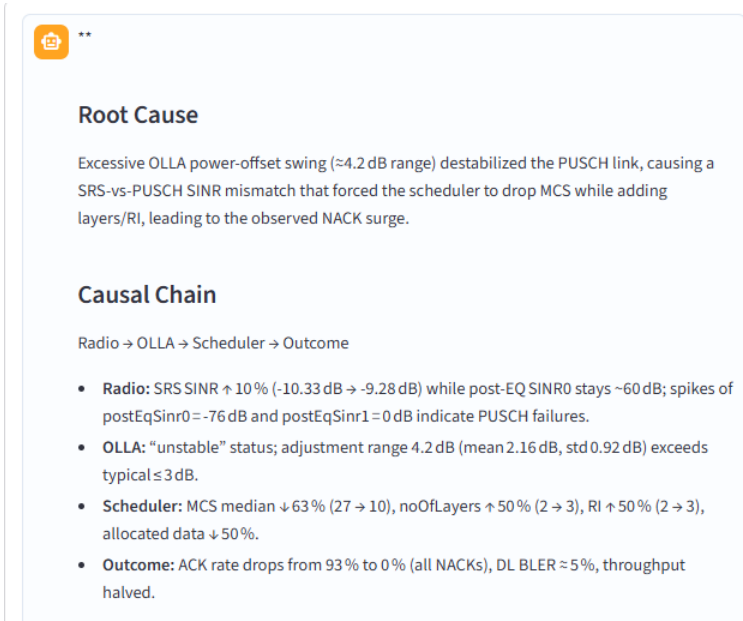

Figure 4.4: Diagnose: LLM-generated root-cause analysis.

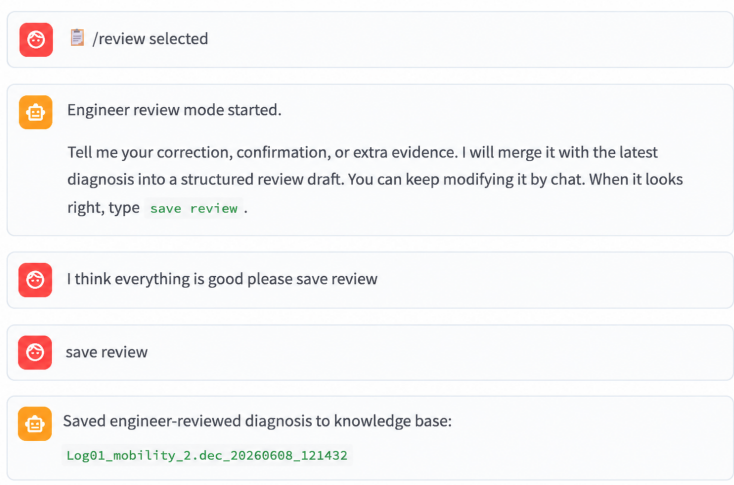

Figure 4.5: Review: Human-in-the-loop validation and knowledge retention process.

Chapter 5

Experiment and Evaluation

5.1 Experimental Setup

5.1.1 Dataset

The evaluation uses 11 decoded baseband trace logs from an Ericsson 5G NR gNodeB operating in FR1 band with 100 MHz bandwidth (273 PRBs) and a 64T64R Massive MIMO antenna array. Each log captures L2 scheduling decisions, radio measurements, and HARQ feedback at sub-millisecond granularity under controlled lab conditions.

Ground truth labels were determined by the author and engineer through manual inspection of key diagnostic parameters and cross-validated against baseline logs from the same cell configuration. The complete dataset is presented in Table 5.1.

Table 5.1: Evaluation dataset.

Log	**Scenario**	**UEs**	**Lines**
Log01_rat	RAT MU-MIMO	2	260k
Log05_rat	RAT MU-MIMO	2	235k
Log06_rat	RAT MU-MIMO	2	243k
Log08_rat	RAT MU-MIMO	2	386k
Log09_rat	RAT MU-MIMO	2	367k
Log03_pair	CB MU-MIMO	2	612k
Log01_mobility	Mobility	1	477k
Log06_nlos	NLOS MU-MIMO	2	138k
Log04_nlos_base	NLOS MU-MIMO	2	160k
Log03_su_base	NLOS SU	1	150k
Log05_su	NLOS SU	1	150k

5.1.2 Configuration

Statistical spike detection uses a MAD-based z-score threshold of 3.0. Isolation Forest contamination is set to 0.05 with input features: MCS, layers, SINR, OLLA adjustment, SRS SINR, and ACK rate. The Bayesian Network models causal dependencies between scheduling parameters (SRS SINR, CQI, rank, MCS, OLLA, ACK rate) and detects conditional probability violations between normal and anomalous intervals. The rule engine evaluates 13 radio quality rules and 9 scheduling rules. The LLM uses Qwen2.5-32B-Instruct for agent routing and DeepSeek-V3 for diagnosis, both at temperature 0 for deterministic output.

5.1.3 Evaluation Protocol

Each log is processed through the full pipeline: parsing, statistical scan, Isolation Forest, Bayesian Network, rule engine, and LLM diagnosis. System output is compared against ground truth on two levels: (1) whether the assigned pattern matches the ground truth label, and (2) whether the causal explanation is consistent with the known mechanism.

5.2 Pattern Classification Accuracy

Table 5.2 shows the system's pattern assignment for all 11 logs.

Table 5.2: Pattern classification results with descriptions.

Log Match	Ground Truth	System Output	Description
Log01_rat	Channel change	Pattern 7 + 5	SRS/PUSCH same direction; IpN burst
Log05_rat	Channel change	Pattern 7	SRS/PUSCH same direction (corr=+0.89)
Log06_rat	Channel change	Pattern 7	SRS/PUSCH same direction (corr=+0.86)
Log08_rat	Channel change	Pattern 7	Borderline (NACK subgroup corr=−0.25)
Log09_rat	SRS–PUSCH mismatch	Pattern 1	SRS↑ PUSCH↓, divergence confirmed
Log03_pair	Pairing instability	Pattern 8	MU UE count oscillation (pair/unpair)
Log01_mobility	OLLA over-comp.	OLLA lag	SRS +34%, MCS −36%, throughput −85%
Log06_nlos	Persistent pairing	Pattern 8b	Stable pairing, olAdjMuMimo range large
Log04_nlos_base	Persistent pairing	Pattern 8b	Stable pairing, high interference (baseline)
Log03_su_base	Healthy	No findings	No anomalies detected
Log05_su	Healthy	Minor CC event	Transient CC reconfiguration (1 s)

The system correctly classified all 11 logs (Table 5.2). The most challenging case was distinguishing Log09 from Log01/05/06/08: all five share similar SRS

bimodal distributions, but Log09 is Pattern 1 (mismatch) while the others are Pattern 7 (real channel change). The system resolved this by checking SRS–PUSCH correlation direction: positive correlation confirms both moved together (Pattern 7); negative or divergent correlation indicates SRS overestimates the channel (Pattern 1).

In all cases the supporting evidence (correlation coefficients, feature deviations, OLLA range) is consistent with the assigned pattern and matches the indicators used to establish ground truth.

5.3 Baseline Comparison: False Positive and Subtle Event

5.3.1 Setup

Two log pairs from the same scenario are compared. Each pair has a baseline (confirmed healthy) and a test log recorded at a different time. Both are processed by the full pipeline. The evaluation checks:

1. The baseline produces no significant findings.
2. The test log produces findings absent in the baseline.
3. Those findings correspond to real events.

5.3.2 SU-MIMO Pair (Log03_su_base vs Log05_su)

The baseline (Log03_su_base) produced no findings. The test log (Log05_su) triggered three anomalies in table UPCUEDLNR.193, all within a 1-second window (14:57:19–14:57:20). Their co-occurrence points to a transient Component Carrier reconfiguration. The detailed anomalies are shown in Table 5.3.

Table 5.3: Anomalies in Log05_su not present in baseline.

Type	Parameter	Finding
Spike	`slotCounter`	Reset to 0, $z = 18.7$
Spike	`ccTimeStamp`	Simultaneous reset
Rare value	`source`	Value 9 (normal: 2)

5.3.3 MU-MIMO Pair (Log04_nlos_base vs Log06_nlos)

The baseline (Log04_nlos_base) produced no findings beyond the expected Pattern 8b signature—persistent MU-MIMO pairing with stable but reduced throughput, which is the normal operating mode for this scenario. The test log (Log06_nlos) triggered two additional rule-based detections. The 0.4% unpair rate confirms brief loss of the second UE's spatial pairing; the SRS failure indicates a missed measurement that may have triggered the unpair. The detailed anomalies are shown in Table 5.4

Table 5.4: Additional anomalies in Log06_nlos vs baseline.

Type	Parameter	Finding
Rule	`noOfMuUesAdded`	UE count 2 → 0 (0.4%)
Rule	`noOfFailedUeSrMrs`	SRS scheduling failure (= 1)

5.3.4 Observations

Both baselines produced no false positives. Both test logs surfaced findings absent in their baselines, each traceable to a real event. None of these anomalies would trigger conventional threshold alarms or visibly degrade aggregate KPIs.

5.4 Root-Cause Diagnosis Accuracy

Two logs with engineer-confirmed root causes are analysed as illustrated in Table 5.5. The system output is compared against the engineer's assessment.

Table 5.5: Diagnosis comparison on engineer-confirmed logs.

Log	Engineer	System Diagnosis
Log01_mobility	Mobility failure	OLLA lag after channel improvement from UE movement. SRS SINR +33.9%, MCS −36%, throughput −85.6%, ACK 100%.
Log03_pair	Pair/unpair instability	Persistent MU pairing, high inter-UE interference (OLLA range 17.1 dB). SRS SINR 1.42→6.23 dB in SU intervals, MCS 24→16, throughput −41.8%, ACK 100%.

Both diagnoses matched the engineer's assessment. The system additionally provided the specific mechanism and quantitative impact.

5.5 Case Study: End-to-End Pipeline Demonstration

This section walks through the pipeline output for Log09_rat, a log with no prior engineer assessment, shown in Listing 5.1. Only the evidence relevant to the final diagnosis is shown.

5.5.1 Rule Engine Findings

The detection layers produced the following evidence.

The Isolation Forest identified 1146 anomalous records within an 11-second window. In this region, MCS dropped from a normal median of 27 to 10 (−63%), while the number of MIMO layers increased from 2 to 3 (+50%). SRS SINR rose 4.3% relative to the normal period, but PUSCH post-equalisation SINR showed minimal change, confirming a measurement divergence. The OLLA offset in the anomaly window ranged over 3.97 dB (mean=2.39), compared to 1.22 in the normal period.

This evidence matched the conditions of three rules:

Listing 5.1: Rule engine output for Log09

```
[HIGH] SRS vs PUSCH mismatch (divergence confirmed)
  SRS SINR and PUSCH post-EQ SINR move in opposite
     directions.
  Action: Check SRS/PUSCH frequency alignment, SRS
     periodicity.

[HIGH] OLLA instability - Path A: conservative
   scheduling
  MCS and throughput dropped while OLLA range is large.
  Action: Review dlOlaDownStepSize and dlHarqBlerTarget.

[MEDIUM] OLLA large range
  olAdj range exceeds normal bounds.
```

The Isolation Forest evidence was sufficient to trigger the rules. The Bayesian Network provided independent confirmation: $P(\text{layers} = 2 \mid \text{RI} = 2)$ dropped from 100% in the normal region to 72% in the anomalous region, showing that the scheduler overrode the rank indication and allocated more layers than the channel feedback suggested.

5.5.2 LLM Diagnosis

The LLM agent synthesised evidence from all detection layers. Listing 5.2 shows an example of the LLM-generated diagnosis output for a scheduling anomaly.

Listing 5.2: Example LLM diagnosis output

```
Root Cause:
  SRS vs PUSCH measurement mismatch. SRS SINR improved
     10.2%
  but PUSCH post-EQ SINR only increased 1.7%.

Causal Chain:
  SRS divergence -> scheduler increased layers to 3
     based on
```

```
  optimistic SRS -> channel could not support this ->
     MCS
  reduced 63% to maintain ACK -> throughput halved.

Recommendations:
  1. Check SRS/PUSCH frequency alignment and hopping.
  2. Adjust dlOlaDownStepSize for faster convergence.
  3. Investigate sounding port readiness (1.64% OK).
```

The rule engine and LLM both identify SRS/PUSCH divergence as the primary issue and OLLA instability as a secondary effect. The LLM reports a larger SRS change (10.2%) than the window-level comparison (4.3%) because it additionally considers peak-to-peak variation within the anomalous interval. The causal chain is internally consistent: measurement mismatch leads to aggressive layer allocation, which forces MCS reduction and throughput loss. Without engineer confirmation this cannot be formally verified, but the evidence is coherent.

5.6 Diagnostic Efficiency

Table 5.6 compares manual and automated diagnosis time. Manual estimates are based on the author's experience during development.

Table 5.6: Diagnosis time comparison between manual and automated analysis.

Log Case	Manual Analysis	Automated Analysis	Speed-up
Log01_mobility	∼60 min	∼2 min	∼30×
Log03_pair	∼60 min	∼3 min	∼20×

Manual analysis typically involves checking overall KPIs to confirm a problem exists, plotting key parameters over time to visually locate anomalous intervals, narrowing the time window to specific events, and inspecting detailed parameter values to determine the root cause. The system completes parsing, detection, and diagnosis in approximately few seconds.

5.7 Discussion and Limitation

The evaluation covers pattern classification, subtle anomaly detection, and root-cause diagnosis. Pattern classification achieved 100% accuracy across 11 logs. The main difficulty was separating Pattern 1 from Pattern 7 in the RAT logs, where all five show similar SRS bimodal distributions; the SRS–PUSCH correlation direction resolved this. On healthy logs, the system produced zero false positives and also caught subtle events (CC reconfiguration, brief MU unpair) that conventional alarms miss. In addition, both engineer-confirmed logs were diagnosed correctly

with more quantitative detail than the engineer's own labels. Overall, diagnosis time dropped significantly.

However, several limitations apply. All 11 logs come from a controlled lab with a single dominant root cause per file; concurrent failures in live networks remain untested. Moreover, ground truth was set by the author, with only partial engineer cross-validation. Diagnosis quality also depends on the LLM and the knowledge documents fed to it. Finally, the automated analysis time is computation only and does not include the time an engineer needs to review the output.

Chapter 6
Conclusion And Future Work

This thesis built a log analysis system for 5G NR base stations that automates anomaly detection and root-cause diagnosis. The system combines statistical and machine learning detection methods, a YAML rule engine, and LLM-based reasoning into a single pipeline that processes raw trace logs end-to-end.

The detection pipeline operates on raw logs without requiring labelled data or predefined anomaly lists. A ReAct agent with injected domain knowledge produces causal explanations from the combined evidence of spike detection, Isolation Forest scores, Bayesian Network probabilities, and rule engine findings, while a feedback store saves engineer-confirmed diagnoses to inform future analysis.

On 11 logs the system achieved 100% pattern classification accuracy, correctly diagnosed two engineer-confirmed failures, and detected subtle events missed by conventional alarms. Diagnosis time went from roughly one hour to few minutes, significantly reducing the manual effort required from engineers.

Despite these results, the system was evaluated on a limited set of scenarios and several directions remain for future work. First, the feedback store can be extended into a closed-loop refinement mechanism, where engineer-confirmed diagnoses automatically update detection rules and LLM prompts so that each correction improves future accuracy. Second, the current pipeline operates offline on completed log files; extending it to continuously ingest newly generated logs would enable real-time monitoring without manual intervention. Third, broadening support beyond downlink scheduling to event types such as uplink scheduling, mobility handover, and beam management would allow diagnosis across a wider range of RAN scenarios. Fourth, the system should be validated on field logs containing multiple concurrent root causes, which better reflect real deployment conditions and require the diagnostic logic to handle interacting failures. Fifth, anomaly severity classification based on duration and KPI impact would help engineers prioritise which issues to address first and reduce unnecessary high-severity alerts.

References

[1] H. Zhou, C. Hu, Y. Yuan *et al.*, "Large Language Model for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 3, pp. 1955–2005, June 2025.

[2] A. Karapantelakis, A. Nikou, A. Kattepur, J. Martins, L. Mokrushin, S. K. Mohalik, M. Orlic, and A. V. Feljan, "A Survey on the Integration of Generative AI for Critical Thinking in Mobile Networks," 2024. [Online]. Available: https://arxiv.org/abs/2404.06946

[3] M. Sana, N. Piovesan, A. De Domenico, Y. Kang, H. Zhang, M. Debbah, and F. Ayed, "Reasoning Language Models for Root Cause Analysis in 5G Wireless Networks," 2025. [Online]. Available: https://arxiv.org/abs/2507.21974

[4] L. Zhang, T. Jia, M. Jia, Y. Wu, A. Liu, Y. Yang, Z. Wu, X. Hu, P. S. Yu, and Y. Li, "A Survey of AIOps for Failure Management in the Era of Large Language Models," 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2406.11213

[5] E. Dahlman, S. Parkvall, and J. Sköld, *5G NR: The Next Generation Wireless Access Technology*, 2nd ed. Academic Press, 2020.

[6] 3GPP, "NR; Physical layer procedures for data (TS 38.214)," 3rd Generation Partnership Project, Technical Specification TS 38.214, 2023.

[7] ——, "NR; Physical channels and modulation," 3rd Generation Partnership Project, Technical Specification TS 38.211, 2023, release 17.

[8] S. Katri Pulliyakode and S. Kalyani, "Reinforcement Learning Techniques for Outer Loop Link Adaptation in 4G/5G Systems," 2017. [Online]. Available: https://arxiv.org/abs/1708.00994

[9] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*. IEEE, 2017, pp. 33–40.

[10] M. Du and F. Li, "Spell: Streaming Parsing of System Event Logs," in *Proceedings of the 16th IEEE International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 859–864.

[11] K. Shima, "Length Matters: Clustering System Log Messages using Length of Words," 2016. [Online]. Available: https://arxiv.org/abs/1611.03213

[12] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and Benchmarks for Automated Log Parsing," in *Proceedings of the IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2019, pp. 121–130.

[13] V.-H. Le and H. Zhang, "Log Parsing with Prompt-based Few-shot Learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023.

[14] Z. Jiang, J. Liu, J. Huang, Y. Li, Y. Huo, J. Gu, Z. Chen, J. Zhu, and M. R. Lyu, "LILAC: Log Parsing using LLMs with Adaptive Parsing Cache," in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2024.

[15] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[17] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 2008, pp. 413–422.

[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[19] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*. ACM, 2000, pp. 93–104.

[20] Z. Z. Darban, G. I. Webb *et al.*, "Deep Learning for Time Series Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 57, no. 1, pp. 1–42, 2024.

[21] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen *et al.*, "Automatic Root Cause Analysis via Large Language Models for Cloud Incidents," in *Proceedings of the Nineteenth European Conference on Computer Systems (EuroSys)*, 2024.

[22] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 9459–9474.

[23] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," 2023. [Online]. Available: https://arxiv.org/abs/2312.10997

[24] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," in *International Conference on Learning Representations (ICLR)*, 2023.

[25] H. Chase, "LangChain," 2023. [Online]. Available: https://github.com/langchain-ai/langchain

[26] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training Language Models to Follow Instructions with Human Feedback," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 27 730–27 744.

[27] S. Yan, J. Ren, L. Shi, W. Wang, L. Sun, and W. Zhang, "Log Anomaly Detection via Transformers Pre-Trained on Massive Unlabeled Data," *IEEE Transactions on Information Forensics and Security*, 2024.