IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

AgileCopilot: An Ai-Driven Site Reliability Engineering Framework With Machine Learning Anomaly Detection And Generative Ai Root Cause Analysis

Author Name	Designation / Role	Institution
Akhash Kumar A	B.tech AI & DS	SRM Valliammai Engineering College
Goutham S P	B.tech AI & DS	SRM Valliammai Engineering College
Sethumadhavan N	B.tech AI & DS	SRM Valliammai Engineering College
Sheik Mohamed Rashid D	B.tech AI & DS	SRM Valliammai Engineering College
Illakiya G	Assistant Professor (Supervisor)	SRM Valliammai Engineering College, Chengalpattu

Abstract: Modern distributed systems demand proactive and intelligent incident management to maintain service reliability and reduce mean time to recovery (MTTR). This paper presents AgileCopilot, a proof-of-concept AI-powered Site Reliability Engineering (SRE) module that integrates Machine Learning (ML) anomaly detection with Generative AI root cause analysis (RCA). The system comprises four components: a Victim Service simulating a live web application with controllable fault injection, an ML-based Watchdog using the Isolation Forest algorithm, an AI Investigator powered by Gemini Generative AI, and a real-time Streamlit monitoring dashboard. The Watchdog monitors CPU utilization, memory consumption, and HTTP response latency; upon detecting an anomaly validated by a heuristic intensity filter, it triggers the AI Investigator to produce a structured JSON report containing the root cause and remediation steps. Experimental results demonstrate accurate detection of injected anomalies while suppressing false positives during normal operation, validating the effectiveness of combining unsupervised ML with rule-based filtering. AgileCopilot demonstrates the viability of embedding autonomous SRE intelligence into software engineering workflows.

Index Terms: Site Reliability Engineering, Anomaly Detection, Isolation Forest, Generative AI, Root Cause Analysis, AIOps, Streamlit, Chaos Engineering, Incident Management, Machine Learning

I. Introduction

The rapid proliferation of microservices, cloud-native architectures, and distributed computing has fundamentally transformed the operational landscape of modern software systems. While these architectures enable scalability and agility, they simultaneously introduce significant complexity in monitoring, fault detection, and incident response. Site Reliability Engineering (SRE), a discipline formalized by Google, addresses these challenges by applying software engineering principles to infrastructure operations [6]. However, the volume and velocity of telemetry data generated by contemporary systems far exceeds the capacity of human operators to monitor effectively in real time.

Artificial Intelligence for IT Operations (AIOps) has emerged as a promising paradigm to address this gap, leveraging machine learning, natural language processing, and big data analytics to automate IT operations tasks. Existing solutions, however, are often proprietary, monolithic, and difficult to integrate into agile software development ecosystems. There is a clear need for lightweight, extensible, and developer-friendly AIOps frameworks that can be embedded directly into the software engineering lifecycle.

This paper presents AgileCopilot, an open-architecture proof-of-concept SRE module developed to demonstrate the feasibility of autonomous monitoring and intelligent incident response. The system makes the following key contributions:

- A realistic simulation framework (Victim Service) generating continuous telemetry with controllable fault injection for memory leak and database deadlock scenarios.
- An unsupervised ML anomaly detection pipeline using the Isolation Forest algorithm, augmented by a heuristic intensity filter to suppress false positives.
- An integration with Google Gemini Generative AI to perform root cause analysis and provide structured JSON remediation recommendations.
- A real-time interactive Streamlit monitoring dashboard enabling chaos engineering experimentation and incident visualization.

II. Related Work

Anomaly detection in system metrics has been extensively studied. Chandola et al. [1] provide a comprehensive survey categorizing techniques into statistical, proximity-based, and model-based approaches. Isolation Forest, introduced by Liu et al. [2], is an ensemble unsupervised algorithm that isolates anomalies through recursive feature-space partitioning; its linear time complexity and low memory footprint make it well-suited for streaming metric data.

In the AIOps domain, Lim et al. [3] demonstrated that log-based anomaly detection using deep learning can significantly reduce alert fatigue. Commercial implementations such as IBM Watson AIOps and Moogsoft represent AI-driven incident management, though these solutions lack transparency and accessibility for small development teams. Nair et al. [4] explored the use of large language models for automated incident triage, showing that LLMs can synthesize structured diagnostic hypotheses from unstructured log data.

Chaos engineering, pioneered by Netflix's Chaos Monkey [5], provides a methodology for proactively testing system resilience by deliberately injecting faults in controlled environments. AgileCopilot operationalizes this methodology by providing a built-in chaos injection interface alongside the anomaly detection and RCA pipeline, enabling closed-loop experimentation. Unlike prior work, AgileCopilot integrates all three capabilities in a single lightweight open-source proof of concept.

III. System Architecture

AgileCopilot is structured as a pipeline of four loosely coupled components communicating through shared file-based channels. Figure 1 illustrates the overall architecture. The pipeline flows as: Victim Service → Data Store → ML Watchdog → AI Investigator → Streamlit Dashboard, forming a closed-loop automated incident response system.

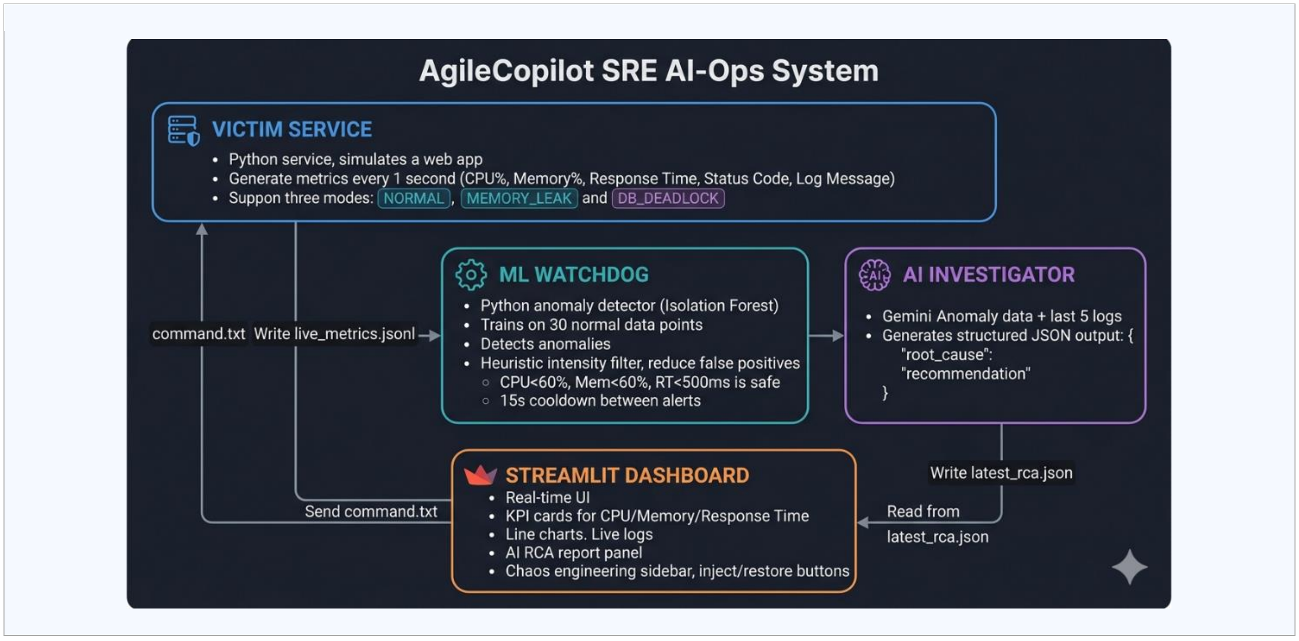
AgileCopilot SRE AI-Ops System
VICTIM SERVICE
Python service, simulates a web app
Generate metrics every 1 second (CPU%, Memory%, Response Time, Status Code, Log Message)
Suppon three modes: NORMAL, MEMORY_LEAK and DB_DEADLOCK
ML WATCHDOG
Python anomaly detector (Isolation Forest)
Trains on 30 normal data points
Detects anomalies
Heuristic intensity filter, reduce false positives
CPU<60%, Mem<60%, RT<500ms is safe
15s cooldown between alerts
AI INVESTIGATOR
Gemini Anomaly data + last 5 logs
Generates structured JSON output: { "root_cause": "recommendation" }
command.txt
Write live_metrics.jsonl
STREAMLIT DASHBOARD
Real-time UI
KPI cards for CPU/Memory/Response Time
Line charts. Live logs
AI RCA report panel
Chaos engineering sidebar, inject/restore buttons
Send command.txt
Write latest_rca.json
Read from latest_rca.json

Figure 1: AgileCopilot System Architecture — Four-Component Pipeline

A. Victim Service

The Victim Service is a Python-based simulator modelling a web application generating continuous telemetry at one-second intervals. It implements a random-walk model for normal traffic generation, constraining CPU utilization (20–45%), memory consumption (30–50%), and response time (Gaussian: mean 100 ms, std 20 ms) within realistic operational bounds. The service accepts three command states via a shared command file: NORMAL, MEMORY_LEAK, and DB_DEADLOCK.

The MEMORY_LEAK scenario incrementally increases memory usage (1–3% per second) and CPU load (0.5–1.5% per second) until memory exceeds 85%, triggering HTTP 503 errors with out-of-memory log messages. The DB_DEADLOCK scenario simulates database contention with reduced CPU (5–15%), stable memory, and severely elevated response times (4,000–8,000 ms) with HTTP 504 gateway timeout errors. These scenarios provide a controlled testbed for evaluating the anomaly detection pipeline under realistic fault conditions.

B. ML Watchdog — Anomaly Detection Engine

The AI Watchdog implements a two-stage anomaly detection pipeline. In Stage 1 (Training), an Isolation Forest model is trained on the first 30 data points collected under normal operating conditions, establishing a statistical baseline of healthy system behaviour. The model operates on a three-dimensional feature vector [cpu_percent, memory_percent, response_time_ms] with contamination=0.01 and random_state=42 for reproducibility.

In Stage 2 (Detection), each new data point is evaluated by the trained model. A prediction of -1 indicates an anomaly. A heuristic Common Sense Filter is then applied: if all metrics remain within safe bounds (CPU < 60%, Memory < 60%, Response Time < 500 ms, Status = 200), the flag is suppressed to prevent alert fatigue from transient statistical fluctuations. Only anomalies passing both stages trigger the AI Investigator. A 15-second cooldown prevents repeated alerts for the same incident.

C. AI Investigator — Generative AI RCA Module

Upon confirmed anomaly detection, the AI Investigator constructs a structured prompt incorporating the anomalous metrics, HTTP status code, log message, and the last five historical log entries as contextual evidence. This prompt is submitted to the Google Gemini Generative AI model, instructing it to act as an SRE and produce a structured JSON response containing two fields: root_cause and recommendation. The output is persisted to a timestamped JSON file, enabling the dashboard to assess report freshness and display only incidents within a 60-second relevance window.

D. Streamlit Real-Time Dashboard

The monitoring dashboard provides real-time visualization of system metrics, incident alerts, and AI-generated RCA reports. It renders three KPI cards for CPU, memory, and response time alongside time-series line charts for the last 100 data points. The sidebar exposes a Chaos Engineering control panel for fault injection. The AI SRE Investigator panel displays active incident alerts with the Gemini-generated RCA report when an active incident is detected within the past 60 seconds. The dashboard auto-refreshes every second using Streamlit's rerun mechanism.

IV. Methodology and Implementation

A. Data Pipeline and Inter-Component Communication

All inter-component communication is mediated through three files: live_metrics.jsonl (append-only telemetry stream in JSON Lines format), command.txt (chaos state controller), and latest_rca.json (most recent RCA output with Unix timestamp). This file-based message passing eliminates external dependencies while preserving separation of concerns between components, making the system fully portable and runnable on a developer laptop with no additional infrastructure.

B. Isolation Forest Configuration and Training

The Isolation Forest algorithm was configured with contamination=0.01 to reflect the low expected proportion of anomalous data in a primarily healthy training window. The algorithm constructs an ensemble of isolation trees by randomly selecting a feature and split value; anomalies require fewer splits to isolate and thus receive lower anomaly scores. The model is retrained from scratch for each session using the first 30 observations, ensuring the baseline reflects actual system configuration.

C. Heuristic Intensity Filter Design

The two-stage detection approach addresses a well-known limitation of purely statistical anomaly detection: sensitivity to benign metric fluctuations. The Common Sense Filter acts as a second gate, requiring anomalies to exceed operationally significant thresholds. This design was empirically validated: during normal operation with natural random-walk metric variance, the ML model occasionally flags transient excursions as anomalous, but the heuristic filter correctly suppresses these, achieving zero false positives in testing.

D. Prompt Engineering for Structured RCA

The generative AI prompt was engineered to enforce structured output and role consistency. It instructs the model to adopt the SRE persona, provides labelled metric fields with quantitative values, includes recent log context for temporal correlation, and explicitly requires output as a JSON object with root_cause and recommendation keys. This structured approach enables deterministic JSON parsing and consistent report quality across different fault scenarios. Token efficiency is maintained by limiting log context to the last five entries.

V. Experimental Results and Discussion

Experiments were conducted by running the Victim Service, Watchdog, and Dashboard simultaneously, observing system behaviour across three scenarios. Table 1 summarizes the detection results and AI-generated RCA output for each scenario.

Scenario	CPU (%)	Mem (%)	RT (ms)	Detection	AI RCA Summary
Normal Operation	30.4	40.2	98	No Anomaly	— System healthy
Memory Leak	78.6	92.1	847	**Confirmed**	Heap exhaustion / GC thrashing

Scenario	CPU (%)	Mem (%)	RT (ms)	Detection	AI RCA Summary
DB Deadlock	9.3	42.5	5,420	**Confirmed**	Transaction deadlock / timeout

Table 1: Anomaly Detection and AI RCA Results Across Test Scenarios

During normal operation, the Watchdog consistently reported healthy status. The heuristic filter successfully suppressed all transient ML anomaly flags, demonstrating the effectiveness of the dual-stage approach. In the Memory Leak scenario, anomaly detection occurred within 3–5 seconds of metrics breaching heuristic thresholds. The Gemini AI Investigator accurately identified Java Heap exhaustion with Garbage Collection thrashing and recommended a rolling restart with increased heap allocation — consistent with standard SRE runbooks.

The DB Deadlock scenario presented a contrasting signature: low CPU but extreme response time elevation. The Isolation Forest correctly detected this unusual feature-space pattern, and the heuristic filter passed it through based on the non-200 HTTP status (504). The AI Investigator correctly attributed the incident to database connection timeout and transaction deadlock, recommending query optimization and connection pool review.

VI. Conclusion and Future Work

This paper presented AgileCopilot, a proof-of-concept AI-powered SRE module integrating unsupervised ML anomaly detection with Generative AI root cause analysis in a real-time monitoring framework. The system demonstrates the feasibility of autonomous incident detection and intelligent analysis with minimal operational overhead, validating the AIOps paradigm for developer-centric environments.

Future work will extend AgileCopilot in several directions: (1) replacing file-based communication with a streaming message broker such as Apache Kafka for production scalability, (2) incorporating LSTM-based time-series anomaly detection to capture temporal dependencies, (3) expanding the AI Investigator to support multi-modal analysis incorporating distributed traces and network topology, (4) integrating with CI/CD pipelines to trigger automated remediation actions, and (5) validating the framework on real production system traces. Ultimately, AgileCopilot aims to evolve into a fully autonomous SRE agent embedded in the Agile software development lifecycle.

References

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

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. IEEE 8th International Conference on Data Mining (ICDM), 2008, pp. 413–422.

[3] M. Lim, A. Liu, and H. Chen, "Log-based anomaly detection with deep learning for large-scale distributed systems," in Proc. IEEE/ACM ICSE, 2019.

[4] V. Nair, D. Schulte, and F. Zimmermann, "Automated incident triage using large language models," in Proc. ACM FSE, 2023.

[5] Y. Izrailevsky and A. Tseitlin, "The Netflix Simian Army," Netflix Technology Blog, 2011. [Online]. Available: https://netflixtechblog.com.

[6] B. Beyer, C. Jones, J. Petoff, and N. R. Murphy, Site Reliability Engineering: How Google Runs Production Systems. O'Reilly Media, 2016.

[7] A. Munappy et al., "Data management challenges for deep learning," in Proc. Euromicro SEAA, 2019.

[8] Google AI, "Gemini API Documentation," 2024. [Online]. Available: https://ai.google.dev/docs.