AIOps for Reliability: Evaluating Large Language Models for Automated Root Cause Analysis in Chaos Engineering

Tomasz Szandała[1,2][0000−0003−4525−0444]

[1] Wroclaw University of Science and Technology, Wroclaw, Poland
[2] The University of Applied Sciences and Arts of Southern Switzerland (SUPSI), Lugano, Switzerland
tomasz.szandala@pwr.edu.pl

Abstract. As modern IT infrastructures grow in complexity, root cause analysis (RCA) is becoming increasingly crucial for Site Reliability Engineering (SRE). Traditional RCA relies heavily on human expertise, making incident resolution time-consuming and error-prone. With the rise of AIOps (Artificial Intelligence for IT Operations), Large Language Models (LLMs) have emerged as potential tools for automating incident detection and diagnosis.

This study evaluates the capability of GPT-4o, Gemini-1.5, and Mistral-small in diagnosing system failures purely from observability metrics within a chaos engineering framework. We simulate eight real-world failure scenarios in a controlled e-commerce environment and assess LLMs' performance in zero-shot and few-shot settings compared with Site Reliability Engineers. While LLMs can identify common failure patterns, their accuracy is highly dependent on prompt engineering. In zero-shot settings, models achieve moderate accuracy (44–58%), often misattributing harmless load spikes as security threats. However, few-shot prompting improves performance (60–74% accuracy), suggesting that LLMs require structured guidance for reliable RCA.

Despite their potential, LLMs are not yet ready to replace human SREs, who achieved over 80% accuracy due to hallucinations, misclassification biases, and lack of explainability. The findings highlight that LLMs can be co-pilots in incident response, but human oversight remains essential. GitHub with code and dataset: https://github.com/szandala/llms-chaos-engineering

Keywords: AIOps · DevOps · Large Language Models · Chaos Engineering · Root Cause Analysis

1 Introduction

Site Reliability Engineers (SREs) are high-demand infrastructure design and management specialists who ensure the continuous operation of global service providers like Microsoft (Azure), Google (GCP), Amazon (AWS), and their customer institutions, critical to modern society [15, 17]. A key responsibility is

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

responding to alerts to minimise the blast radius of potential incidents and maintaining service availability and reliability. Over 40% SREs report overwhelm and weariness from the amount of on-call duties they provide, and over half of them highlight it leads to the degradation of their work quality [6]. On average, addressing a single incident, including root cause analysis (RCA), remediation, and follow-up activities like writing postmortems and fixing bugs, takes approximately 6 hours [9]. RCA is a critical process in system reliability engineering [8], often requiring expert knowledge to identify the failure's underlying reasons [20].

Artificial intelligence (AI) is emerging in SRE in the form of AIOps (Artificial Intelligence for IT Operations) [14, 25], with Large Language Models (LLMs) demonstrating significant capabilities in tasks such as coding [4, 31], static analysis [24], and IT infrastructure design [21].

These advancements raise an essential question: *Can an LLM effectively receive an alert and diagnose the root cause of an incident within a complex system*?

This paper explores whether an LLM can autonomously process observability data from chaos experiments to determine the root cause of incidents. Chaos engineering provides a structured approach to resilience testing by deliberately injecting failures into a system to observe its behaviour under adverse conditions [19, 22, 7]. Specifically, we evaluate its effectiveness in diagnosing faults within a controlled e-commerce environment consisting of three hosts running a Django-based application, a Redis caching layer, and a PostgreSQL database with a single replica.

The key research questions addressed in this study are:

RQ1 : Can an LLM correctly identify an incident (e.g., Redis failure, database slowdown) based on metrics without additional fine-tuning?

RQ2 : How does its performance vary in zero-shot vs. few-shot settings? Does providing an example in the prompt significantly improve the results?

RQ3 : What are the limitations and practical barriers to deploying LLMs for automated incident analysis?

We utilised GPT-4o (referred to as GPT), Gemini 1.5 Flash (referred to as Gemini), and Mistral Small (referred to as Mistral) for the research as they are among the most popular and widely used models available in free chat interfaces [13, 27]. These models have been extensively tested, ensuring their reliability for various tasks. Additionally, they offer a good balance of speed and accessibility, making them practical for large-scale evaluations. Their availability as free, reasonably fast options allows for broader experimentation without significant computational constraints, making them ideal candidates for the study.

This paper's main contribution empirically demonstrates that Large Language Models can accurately identify infrastructure incidents from raw metrics without fine-tuning. It highlights the critical role of prompt engineering and human oversight in ensuring reliable automated analysis. It contributes to the broader discourse on LLM applicability in Site Reliability Engineering to make complex infrastructure systems more tractable and diagnosable.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

2 State of the Art

2.1 LLMs in Coding

LLMs have shown remarkable capabilities in automatic code generation. OpenAI's Codex demonstrated that an LLM trained on billions of lines of code could produce functioning code from natural language prompts [11, 28]. Since then, numerous code-focused LLMs have emerged, e.g. Meta's Incoder, Amazon's CodeWhisperer, BigCode's SantaCoder/StarCoder, Salesforce's CodeGen, to translate user intent into code [32]. Song et al. [32] have proven that GPT-4 can correctly solve around 88% of problems in the HumanEval coding benchmark [11], far surpassing older models, like GPT-3 73% and open-source StarCoder 34% on the same tasks.

Many failures originate from the model missing parts of the specification in natural language, like skipping necessary conditions or overwhelming the model with too extensive information. Early results indicate that with the proper prompts, LLMs can identify and correct some mistakes, although reliable human-like debugging remains an open challenge [36].

2.2 LLMs for DevOps

Infrastructure as Code (IaC) has become a cornerstone of modern DevOps, allowing cloud infrastructure to be defined and managed through code (e.g. Terraform scripts, Ansible playbooks) [18].

This has prompted interest in using LLMs to generate, evaluate and manage infrastructure code automatically [34, 26, 23] LLMs can reliably evaluate DevOps solutions for small-scale problems [35], comparing their assessments to those of human experts in IT. The study finds that LLMs produce evaluations consistent with human reviewers and thus can serve as an effective tool for assessing infrastructure designs.

Srinivasan et al. [34] conducted a survey and experiments on LLM-generated Terraform configurations. In their tests, OpenAI's GPT-3.5 model was prompted to generate configurations for 49 cloud resources. The results showed moderate success: with a single attempt, GPT-3.5 produced exact-match correct configs about 59% of the time. However, the 40% misaccuracy means many configurations were wrong. The most prominent issue are still hallucinations – the models invented resource types and attributes that do not exist or misnamed them [16, 29]. Unfortunatelly, humans are still humans and express tendency, often referred to as *automation bias*, that lead operators to accept AI-generated outputs without sufficient scrutiny, posing significant risks in safety-critical infrastructure scenarios [10].

2.3 LLMs in Evaluation and Incident Analysis

SRE and DevOps teams deal with massive streams of logs and alerts, often 24/7 [33]; parsing these to identify anomalies or the root cause of an incident is

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

time-consuming and error-prone. LLMs, with their strength in language understanding, are being explored as tools to automate the analysis of such unstructured operational data [2].

Several recent studies focus on using LLMs to parse event logs and detect issues. For instance, Almodovar et al. [5] successfully fine-tuned a language model based on BERT for log anomaly detection (LogFiT). A study by Akhtar et al. [2] reports that with fine-tuning, LLM-based log analysis systems often reach F1-scores in the 0.9–1.0 range on benchmark tasks, a significant improvement over earlier statistical or ML approaches that were usually below 0.9.

This indicates that when fine-tuned, LLMs can reliably distinguish normal vs. abnormal events and classify the type of incident occurring. Moreover, according to Shen et al. [30], fine-tuning can sometimes be replaced with few-shot prompting. They observed an off-the-shelf model with a few examples in prompt performed comparably with a model that was fine-tuned on historical data.

Ahmed et al. [1] presented one of the first large-scale studies on using LLMs for cloud incident RCA. They evaluated GPT-3 family models on 40,000+ real incidents from Microsoft's production systems, testing whether the models could identify the likely root cause and suggest mitigation steps. The models were tried in zero-shot mode, as well as fine-tuned on a subset of incidents. The findings were encouraging because the LLMs often generated reasonable causes and helpful remediation advice.

By analysing trends in logs and metrics, an LLM might forecast incidents before they happen [37]. For example, HuntGPT [3] integrates an anomaly detection system with an explainable AI module and an LLM, aiming to detect and explain incidents in one loop.

2.4 Identified Research Gaps

While research on LLMs in software engineering has progressed swiftly, notable gaps and open challenges remain. One prominent gap is in root cause analysis for infrastructure incidents: despite initial studies, this topic is still in its infancy. We now have a handful of papers showing it is possible to get decent answers from LLMs about why an incident occurred, but they often require fine-tuning for specific problems (**RQ1**).

Another gap lies in generalisation and context awareness. Current LLM solutions for incident analysis are specific to the environment they were designed for. The goal would be to devise a fast, inexpensive method to improve the model's analysis, preferably using prompt engineering (**RQ2**).

Finally, a critical gap lies in trust, safety, and ethics of applying LLMs in infrastructure contexts. Many authors note the risk of hallucinations and errors leading to misguided actions. Therefore it is vital to evaluate how far from the real problem the model drifts (**RQ3**).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

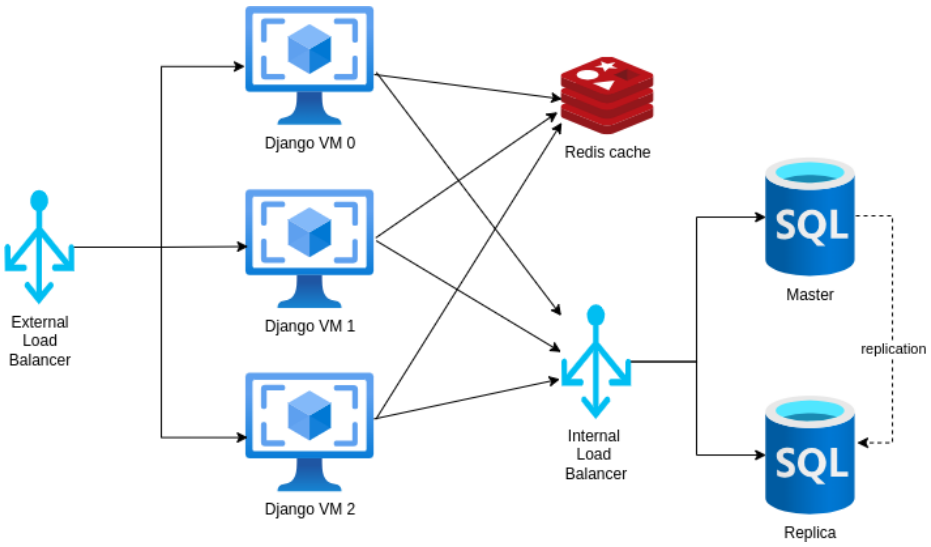

Fig. 1. Diagram of testing environment infrastructure.

3 Methods

3.1 Infrastructure Setup

To evaluate the feasibility of an automated root cause analysis using LLMs, a small e-commerce application was launched on virtual machines (VMs) (Figure 1). The main application tier consists of three VMs (each with 2 vCPUs and 4 GB RAM), fronted by a load balancer. These instances run a Django application that handles both the web-based front-end and the internal API.

A PostgreSQL database was installed on two VMs (each 2 vCPUs, 4 GB RAM), storing approximately 200 GB of synthetic data. Synchronous replication was enabled, ensuring that one VM operated as the primary database server while the second acted as a live replica. In the event of a failure on the primary node, traffic would automatically be re-routed to the replica, enabling a fault-tolerant environment representative of production-level deployments.

A Redis server was provisioned on a single VM (2 vCPUs, 2 GB RAM) in master mode. Redis was utilised for caching frequently accessed data (e.g., product listings and user session tokens). An upper memory limit has been set and enforced with a *volatile-lru* eviction policy, removing the least recently used keys with a set expiration.

Although more powerful machines could have been employed, smaller VM configurations facilitated more frequent occurrences of specific stress conditions (e.g., server overload), thus yielding richer data on performance degradation.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

3.2 Monitoring and Alerting

All VMs—application, database, and cache—were instrumented to collect a standard set of performance and reliability metrics (summarised in Table 1). These metrics included CPU/memory utilisation of all components, error rates, cache-specific metrics and database latency metrics. The metrics were sampled at 30-second intervals. Whenever any metric exceeded its defined critical threshold, an alert aggredated the previous five minutes of metric data and forwarded it for further investigation.

Table 1. Metrics and their threshold used in the experiment

Metrics	Alert Threshold	Short Description
djangoCpu(0,1,2)	>80%	CPU usage for each of the Django instances
djangoMem(0,1,2)	>80%	Memory usage for each of the Django instances
http4xxRate	>1%	Ratio of HTTP 4xx errors to total requests
http5xxRate	>0.5%	Ratio of HTTP 5xx errors to total requests
p95LatencyMs	>300 ms	95th percentile of request latency
p99LatencyMs	>500 ms	99th percentile of request latency
redisCpu	>80%	CPU usage on the Redis server
redisMem	>80%	Memory usage on the Redis server
redisHitRate	<75%	Ratio of Redis queries served from cache
redisEvictedKeys	>50	Redis keys evicted due to memory constraints
dbCpuMaster	>80%	CPU usage on the primary PostgreSQL server
dbMemMaster	>80%	Memory usage on the primary PostgreSQL server
dbCpuReplica	>80%	CPU usage on the replica PostgreSQL server
dbMemReplica	>80%	Memory usage on the replica PostgreSQL server
dbReplicationLag	>150 ms	Delay in synchronising from master to replica

3.3 Chaos Engineering Scenarios

To gather data for RCA, a series of eight fault scenarios was devised inspired by chaos engineering principles:

1. **Sudden Failure of a Single Django Instance:** One VM in the application tier is abruptly terminated, simulating an unplanned outage. The load balancer must redistribute requests among the remaining two instances.
2. **Cache Outage:** Redis goes offline or becomes unresponsive, preventing session and hot-data caching. The system experiences increased latency as queries must be served directly from the database.
3. **Primary Database Failure:** The primary PostgreSQL node fails, triggering a failover to the synchronous replica. This scenario tests the system's ability to handle rapid role promotion and re-routing of database connections.
4. **High-Volume Data Upload:** One Django instance pushes a large data update (e.g. product catalogue), potentially saturating both database I/O and application CPU usage.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

5. **Network Throttling:** Artificially imposed network delay or packet loss affects traffic among the VMs, replicating real-world connectivity issues.
6. **Bugged Deployment Causing Memory Leak:** A new release of the Django application contains a severe memory leak. Over time, memory utilisation sharply increases, ultimately impacting response times and possibly causing the affected VM to crash.
7. **Bugged Deployment Causing URLs Misconfiguration:** An incorrect routing configuration directs requests to non-existent endpoints, raising errors of the 4xx class (e.g., 404) and reducing successful requests significantly.
8. **A Surge of Traffic:** A spike of simulated user traffic increases the load on the entire stack. This is not strictly an *incident* but tests how well the infrastructure scales under peak demand without failing.

Rather than focusing solely on assessing availability or reliability, the scenarios were deliberately chosen to gather diverse metric patterns for subsequent analysis by an LLM-based workflow. The study was limited to eight single-fault scenarios. They cover the most frequent failures seen in production - compute, cache, database, network, and two typical bad-deploy patterns - yet remain mutually distinguishable in their metric footprints. This set gives both humans and LLMs a realistic but non-overlapping taxonomy to reason about.

Overall, this methodology provided a realistic, small-scale e-commerce environment and a controlled injection of diverse fault scenarios, generating a robust data set for investigating how automated, metric-based analyses can pinpoint the underlying causes of operational incidents.

4 Results

A total of 50 alerts were generated during the experiments, each triggered by one of eight chaos engineering actions. For every alert, a single snapshot containing the previous five minutes of metric data was presented to participants, either human SRE Team or large language models, to identify the root cause in one sentence.

4.1 Zero-Shot Performance

In the zero-shot setup, humans and LLMs were merely informed of the overall infrastructure.

The prompt was: *Given a 5-minute snapshot of metrics for our small e-commerce application—which includes 3 VMs running a Django app, a Redis cache on a separate VM (non-critical), and a PostgreSQL database with a master and a replica for fallback, please diagnose in one sentence the root cause of the alert triggered by metric* ***A*** *(threshold set to* ***'a'****). Note that incidents are synthetic infrastructure or application problems introduced by Chaos Engineering.*

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

Table 2. Zero-shot predictions for root cases for SRE Team

		Predicted								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**other**
Actual	**1**	**6**	-	-	-	-	-	-	-	-
	2	-	**4**	2	-	-	-	-	-	-
	3	-	2	**4**	-	-	-	-	-	-
	4	-	1	-	**1**	3	-	-	1	-
	5	-	-	-	-	**4**	-	-	2	2
	6	-	-	-	-	1	**5**	-	-	-
	7	-	-	-	-	-	-	**5**	-	1
	8	-	-	-	-	4	-	-	**2**	-

Table 3. Few-shot predictions for root cases for SRE Team

		Predicted							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual	**1**	**6**	-	-	-	-	-	-	-
	2	-	**4**	2	-	-	-	-	-
	3	-	1	**5**	-	-	-	-	-
	4	-	1	-	**3**	1	-	-	1
	5	-	-	-	-	**7**	-	-	1
	6	-	-	-	-	-	**6**	-	-
	7	-	-	-	-	-	-	**6**	-
	8	-	-	-	-	2	-	-	**4**

Table 4. Zero-shot predictions for root cases for GPT-4o

		Predicted								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**other**
Actual	**1**	**5**	-	-	-	1	-	-	-	-
	2	-	**3**	2	-	1	-	-	-	-
	3	-	3	**3**	-	-	-	-	-	-
	4	-	1	-	**1**	3	-	-	-	1
	5	-	2	-	-	**5**	-	-	-	1
	6	2	-	-	-	-	**4**	-	-	-
	7	-	-	-	-	1	-	**5**	-	-
	8	-	1	-	-	2	1	-	**-**	2

Table 5. Few-shot predictions for root cases for GPT-4

		Predicted							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual	**1**	**6**	-	-	-	-	-	-	-
	2	-	**3**	2	-	1	-	-	-
	3	-	3	**3**	-	-	-	-	-
	4	-	1	-	**2**	2	-	-	1
	5	-	-	-	-	**4**	-	-	4
	6	-	-	-	-	-	**6**	-	-
	7	-	-	-	-	1	-	**5**	-
	8	-	-	-	-	2	-	-	**4**

Humans achieved the result of 62% accuracy (Table 2). SREs were able to derive patterns indicative of specific faults under conditions the data was provided in linear chart form. Noteworthy, only humans correctly suspected a non-harmful-incident event - an increased legitimate traffic, simulating promotion, rather than categorising it as a security-related *Distributed Denial of Service* (DDoS) or network-based failure.

LLMs achieved significantly lower results. GPT-4 achieved 0.52, Gemini 0.58, and Mistral 0.44 (Tables 4,6,8). The root causes suggested by these models were classified post hoc into one of the eight known scenarios or "other" if it did not fit pre-defined ones. While GPT-4 consistently avoided security explanations (e.g., DDoS attacks) due to explicit mention in the prompt, Gemini and Mistral occasionally hypothesised DDoS incidents, highlighting the models' tendency to overattribute problems to network-security threats.

4.2 Few-Shot Performance

In the few-shot phase, SREs and LLMs participants received the same promptm but extended with a closed list of the eight known chaos engineering scenarios, mitigating free-form speculation:

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

Table 6. Zero-shot predictions for root cases for Gemini

		Predicted								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**other**
Actual	**1**	5	-	-	-	1	-	-	-	-
	2	-	3	2	-	1	-	-	-	-
	3	-	2	4	-	-	-	-	-	-
	4	-	1	-	2	2	-	-	1	-
	5	-	1	1	-	4	-	-	1	1
	6	-	-	-	-	-	6	-	-	-
	7	-	-	-	-	1	-	5	-	-
	8	-	1	-	-	2	-	-	-	3

Table 7. Few-shot predictions for root cases for Gemini

		Predicted							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual	**1**	6	-	-	-	-	-	-	-
	2	-	4	1	-	1	-	-	-
	3	-	2	4	-	-	-	-	-
	4	-	1	-	2	2	-	-	1
	5	-	1	-	-	6	-	-	1
	6	-	-	-	-	-	6	-	-
	7	-	-	-	-	1	-	5	-
	8	-	-	-	-	2	-	-	4

Table 8. Zero-shot predictions for root cases for Mistral

		Predicted								
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**other**
Actual	**1**	**3**	1	-	-	2	-	-	-	-
	2	-	**3**	2	-	1	-	-	-	-
	3	-	2	**3**	-	-	-	-	-	1
	4	-	1	-	**1**	3	-	-	1	-
	5	-	2	1	-	**4**	-	-	1	-
	6	-	-	-	-	1	**5**	-	-	-
	7	-	-	-	-	1	-	**3**	-	2
	8	-	1	-	-	5	-	-	**-**	-

Table 9. Few-shot predictions for root cases for Mistral

		Predicted							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Actual	**1**	**5**	-	-	-	1	-	-	-
	2	-	**3**	2	-	1	-	-	-
	3	1	3	**2**	-	-	-	-	-
	4	-	1	-	**2**	2	-	-	1
	5	1	1	-	-	**4**	-	-	2
	6	1	-	-	-	-	**5**	-	-
	7	-	-	-	-	1	-	**5**	-
	8	-	-	-	-	2	-	-	**4**

(...previous prompt...) Knowing these are possible classes of root causes:
#1. Simulation of a worker node failure: Sudden shutdown of a VM running Django.
#2. Redis (Master) failure: Unexpected shutdown of the node hosting the Redis instance.
#3. Database failure (PostgreSQL): Shutdown of the master database server.
#4. Mass data update, e.g., products also performed by the one Django application.
#5. Injection of network slowdown or packet loss (network chaos using tc (traffic control)).
#6. Deploying a new version of the application with memory leak.
#7. Deploying a new version of the application with a bug causing redirection to an incorrect URL.
#8. Traffic surge (promotion) on the portal (not an incident, but degrades performance).
Please assign a single RCA label to each incident.

This adjustment improved overall accuracy, allowing SREs to correctly identify 82% root causes (Table 3).

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

Humans exhibited more nuanced reasoning, especially in differentiating network degradation from increased load, by observing whether database replication lag had risen significantly. They correctly concluded that replication lag would remain stable even under higher user traffic but spike during genuine network slowdowns.

LLMs achieved higher results yet still underperformed humans. GPT-4 attained 0.66, Gemini 0.74, and Mistral 0.60 (Tables 5,7,9). All models performed better than in the zero-shot setting, presumably due to explicit constraints that guided them away from irrelevant categories like security breaches or introducing the concept of standard traffic spikes. Nonetheless, they maintained a relatively high tendency to blame general network issues compared to humans, underscoring the challenge of purely metric-based differentiation in machine-generated analyses.

Overall, while human SREs outperformed the tested LLMs in identifying correct root causes, LLMs still proven to be valuable analysts. Despite few-shot prompting improved model accuracy, discrepancies remained, notably in distinguishing harmless load increases from real system impairments.

5 Discussion

The overall results indicate that LLMs can successfully detect and attribute many common failures purely through the lens of time-series data. On the other hand, the degree of accuracy was heavily contingent on factors such as prompt design and the availability of scenario clues.

Prompt engineering emerged as the critical element for LLMs' performance. In the zero-shot context—where the model was tasked to diagnose a problem without any additional guidance—the models (GPT, Gemini, and Mistral) produced substantially lower accuracy than scenarios in which they received more verbose prompts.

This pattern underscores the variability and fragility of LLM-based incident diagnostics. A more verbose prompt substantially improved the performance of all three models, effectively guiding their decision boundaries closer to reality. On the other hand, if the prompt was too detailed, the model was more prone to limit reasoning and forcibly avoided out-of-the-box ideas that may have been valuable additions.

A noteworthy observation was that LLMs tended to default to negative or failure-centric diagnoses, even in scenarios that represented harmless increases in traffic. As a result, innocent events, such as legitimate promotional spikes, were branded as attacks or network faults. This indicates a bias induced by the prompt—when asked to identify an "incident", the model assumes there must be something "broken". For real-world deployments, it is crucial to remember that not all metric anomalies correspond to genuine faults. Ensuring the prompt does not inadvertently narrow the model's perspective could reduce false positives.

Interestingly, Gemini consistently outperformed other models. One hypothesis is that Google may be internally fine-tuning Gemini with infrastructure-

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

related data, especially given its public positioning as an assistant for Google Cloud Platform configuration [12]. If a model's pre-training and fine-tuning data are skewed towards cloud-based incidents and provisioning, it logically follows that it would exhibit superior performance in diagnosing these scenarios.

Beyond sheer accuracy, there are ethical considerations when deploying LLMs in mission-critical domains, such as incident response. Hallucinations and misdiagnoses can mislead engineers, causing them to waste valuable time. Misclassification of harmless events could also create unnecessary panic, triggering unneeded mitigations or escalations. The "automation bias", where humans overrely on AI outputs, is already a known pitfall in safety-critical contexts. Hence, while LLM-driven triaging might offer valuable speed and scalability, human oversight will likely remain mandatory until we achieve a more robust and explainable solution.

Additionally, the deployment of LLMs in operational environments can raise questions about data privacy, as logs and metrics may contain sensitive details about system internals or customer transactions. Ensuring compliance with data protection regulations or corporate security policies is paramount. Even if the LLM is hosted entirely on-premises, the risk of inadvertent data leaks or unauthorised insights must be evaluated. Infrastructure-focused domain fine-tuning, while beneficial for performance, could also expose an organisation to model drift if the LLM's knowledge is not continuously updated, or if it learns outdated or insecure best practices.

5.1 Future Directions

Based on the findings, several future works can be suggested:

- The design of prompts that dynamically adjust the hypothesis space to include both positive and negative causes of metric changes (e.g., legitimate spikes vs. malicious surges) may reduce mislabelling errors.
- Combining real incident data with synthetic scenarios, like chaos engineering, could help expand the model's understanding of edge cases.
- Even if the model performs well in tests, it may still pose a risk for production applications. It is vital to ensure the explainability of each suggestion/decision provided by the model.
- Possible countermeasures for hallucination may be by *Chain of Thought*, where LLM is asked to explain step-by-step its conclusions. This approach will be studied in future work.

In conclusion, the study highlights the potential of using LLMs for root cause analysis in infrastructure incidents, albeit with caution. Prompt engineering emerges as a crucial prerequisite for safe and effective adoption. In their current form, LLMs function best as co-pilots—collaborating with, rather than supplanting, skilled SREs. Any practical implementation should blend these AI-driven insights with domain expertise and critical discernment, which only human operators can provide.

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

6 Conclusions

This study demonstrates that Large Language Models can, to some degree, detect infrastructure faults purely from system metrics without requiring prior fine-tuning (RQ1). However, their effectiveness strongly depends on prompt engineering, with zero-shot modes exhibiting frequent misdiagnoses and more detailed few-shot prompts leading to markedly higher accuracy (RQ2). Despite these encouraging results, the limitations of current LLM-based incident analysis highlight the need for more robust, explainable AI mechanisms; until then, human Site Reliability Engineers remain essential for interpreting system state, validating diagnoses, and overseeing remediation (RQ3).

In practical terms, these findings reveal that while LLMs can serve as powerful co-pilots in root cause analysis, they are not yet ready to assume the full responsibility for production environments. The risks of hallucinations, misdirected suspicion (e.g., attributing routine traffic surges to malicious activity), lack of broader context and transparent justifications all highlight the importance of maintaining human oversight.

Equally, the substantial improvement observed under few-shot prompts underscores the role of domain-specific guidance in boosting the accuracy of LLMs. As the field evolves, subsequent work should integrate adaptive prompt structures and develop LLM explainability features so operational teams can more confidently rely on AI-assisted diagnostics without sacrificing safety or clarity.

References

1. Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., Rajmohan, S.: Recommending root-cause and mitigation steps for cloud incidents using large language models. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). pp. 1737–1749. IEEE (2023)
2. Akhtar, S., Khan, S., Parkinson, S.: Llm-based event log analysis techniques: A survey. arXiv preprint arXiv:2502.00677 (2025)
3. Ali, T., Kostakos, P.: Huntgpt: Integrating machine learning-based anomaly detection and explainable ai with large language models (llms). arXiv preprint arXiv:2309.16021 (2023)
4. Almanasra, S., Suwais, K.: Analysis of chatgpt-generated codes across multiple programming languages. IEEE Access (2025)
5. Almodovar, C., Sabrina, F., Karimi, S., Azad, S.: Logfit: Log anomaly detection using fine-tuned language models. IEEE Transactions on Network and Service Management **21**(2), 1715–1723 (2024)
6. Andersen, K., Vasiliou, L.: The sre report 2025 (2025), https://www.catchpoint.com/, independent research report on Site Reliability Engineering (SRE)
7. Basiri, A., Behnam, N., De Rooij, R., Hochstein, L., Kosewski, L., Reynolds, J., Rosenthal, C.: Chaos engineering. IEEE Software **33**(3), 35–41 (2016)
8. Beyer, B., Murphy, N.R., Rensin, D.K., Kawahara, K., Thorne, S.: The site reliability workbook: practical ways to implement SRE. " O'Reilly Media, Inc." (2018)
9. Breneman, J.E., Sahay, C., Lewis, E.E.: Introduction to reliability engineering. John Wiley & Sons (2022)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

10. Carnat, I.: Human, all too human: accounting for automation bias in generative large language models. International Data Privacy Law p. ipae018 (2024)
11. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.D.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
12. Cloud, G.: Gemini ai (2025), https://cloud.google.com/products/gemini?hl=en, accessed: 2025-02-28
13. De Nadai, C.: The inherent predisposition of popular llm services: Analysis of classification bias in gpt-4o mini, mistral nemo and gemini 1.5 flash (2024)
14. Diaz-De-Arcaya, J., Torre-Bastida, A.I., Zárate, G., Miñón, R., Almeida, A.: A joint study of the challenges, opportunities, and roadmap of mlops and aiops: A systematic survey. ACM Computing Surveys **56**(4), 1–30 (2023)
15. Dubovoi, O.: It job market in 2025: Trends, roles, and opportunities. DEV Community (January 2025), https://dev.to/empiree/it-job-market-in-2025-trends-roles-and-opportunities-bf, accessed: 2025-02-28
16. Fan, A., Gokkaya, B., Harman, M., Lyubarskiy, M., Sengupta, S., Yoo, S., Zhang, J.M.: Large language models for software engineering: Survey and open problems. In: 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE). pp. 31–53. IEEE (2023)
17. Half, R.: Data reveals which technology roles are in highest demand (February 2025), https://www.roberthalf.com/us/en/insights/research/data-reveals-which-technology-roles-are-in-highest-demand, accessed: 2025-02-28
18. Jayaram, V., Sankiti, S.R., Sughaturu Krishnappa, M., Veerapaneni, P.K., Carimireddy, P.K.: Accelerated cloud infrastructure development using terraform. V. Jayaram, SR Sankiti, MS Krishnappa, PK Veerapaneni, and PK Carimireddy," Accelerated Cloud Infrastructure Development Using Terraform," International Journal of Emerging Technologies and Innovative Research **11**(9), f382–f387 (2024)
19. Jernberg, H., Runeson, P., Engström, E.: Getting started with chaos engineering-design of an implementation framework in practice. In: Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). pp. 1–10 (2020)
20. Kahles, J., Törrönen, J., Huuhtanen, T., Jung, A.: Automating root cause analysis via machine learning in agile software testing environments. In: 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). pp. 379–390. IEEE (2019)
21. Khojah, R., Mohamad, M., Leitner, P., de Oliveira Neto, F.G.: Beyond code generation: An observational study of chatgpt usage in software engineering practice. Proceedings of the ACM on Software Engineering **1**(FSE), 1819–1840 (2024)
22. Kikuta, D., Ikeuchi, H., Tajiri, K., Nakano, Y.: Chaoseater: Fully automating chaos engineering with large language models. arXiv preprint arXiv:2501.11107 (2025)
23. Kon, P.T.J., Liu, J., Qiu, Y., Fan, W., He, T., Lin, L., Zhang, H., Park, O., Elengikal, G., Kang, Y., et al.: Iac-eval: A code generation benchmark for cloud infrastructure-as-code programs. Advances in Neural Information Processing Systems **37**, 134488–134506 (2025)
24. Kwon, S., Lee, S., Kim, T., Ryu, D., Baik, J.: Exploring the feasibility of chatgpt for improving the quality of ansible scripts in edge-cloud infrastructures through code recommendation. In: International Conference on Web Engineering. pp. 75–83. Springer (2023)
25. Notaro, P., Cardoso, J., Gerndt, M.: A survey of aiops methods for failure management. ACM Transactions on Intelligent Systems and Technology (TIST) **12**(6), 1–45 (2021)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25

26. Pujar, S., Buratti, L., Guo, X., Dupuis, N., Lewis, B., Suneja, S., Sood, A., Nalawade, G., Jones, M., Morari, A., et al.: Automated code generation for information technology tasks in yaml through large language models. In: 2023 60th ACM/IEEE Design Automation Conference (DAC). pp. 1–4. IEEE (2023)
27. Rangapur, A., Rangapur, A.: The battle of llms: A comparative study in conversational qa tasks. arXiv preprint arXiv:2405.18344 (2024)
28. Rasheed, Z., Waseem, M., Kemell, K.K., Ahmad, A., Sami, M.A., Rasku, J., Systä, K., Abrahamsson, P.: Large language models for code generation: The practitioners perspective. arXiv preprint arXiv:2501.16998 (2025)
29. Reddy, G.P., Kumar, Y.P., Prakash, K.P.: Hallucinations in large language models (llms). In: 2024 IEEE Open Conference of Electrical, Electronic and Information Sciences (eStream). pp. 1–6. IEEE (2024)
30. Shen, Z., Liu, Z., Qin, J., Savvides, M., Cheng, K.T.: Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 9594–9602 (2021)
31. Siddiq, M.L., Roney, L., Zhang, J., Santos, J.C.D.S.: Quality assessment of chatgpt generated code and their use by developers. In: Proceedings of the 21st International Conference on Mining Software Repositories. pp. 152–156 (2024)
32. Song, D., Zhou, Z., Wang, Z., Huang, Y., Chen, S., Kou, B., Ma, L., Zhang, T.: An empirical study of code generation errors made by large language models. In: 7th Annual Symposium on Machine Programming (2023)
33. Spadaccini, A., Guliani, K.: Being an on-call engineer. Operating Systems and Sysadmin p. 43
34. Srivatsa, K.G., Mukhopadhyay, S., Katrapati, G., Shrivastava, M.: A survey of using large language models for generating infrastructure as code. arXiv preprint arXiv:2404.00227 (2024)
35. Szandała, T.: Chatgpt vs human expertise in the context of it recruitment. Expert Systems with Applications **264**, 125868 (2025)
36. VM, K., Warrier, H., Gupta, Y., et al.: Fine tuning llm for enterprise: Practical guidelines and recommendations. arXiv preprint arXiv:2404.10779 (2024)
37. Wang, S.K., Ma, S.P., Chao, C.H., Lai, G.H.: Low-code chatops for microservices systems using service composition. In: 2023 IEEE International Conference on e-Business Engineering (ICEBE). pp. 55–62. IEEE (2023)

ICCS Camera Ready Version 2025
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-97564-6_25