International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): 3071-0154
Volume-1 Issue-2 || Apr-Jun 2025 || PP. 46-50

AI-Enhanced Incident Management in SRE Operations

DOI: https://doi.org/10.63345/ijarcse.v1.i2.106

Justin Rajakumar Maria Thason

BCBSA - Ajace, Inc

Enterprise AI, Agentic AI & Professional Research and Contributions

www.ijarcse.org || **Vol. 1 No. 2 (2025): April Issue**

Date of Submission: 07-03-2025	**Date of Acceptance:** 25-03-2025	**Date of Publication:** 06-04-2025

Abstract : SRE teams face alert storms, extended triage and investigations that make the outage resolution period longer. In this research paper, the authors introduce an AI-powered incident management pipeline, which (a) correlates alerts to incidents, (b) classifies incidents into severities and teams responsible for their investigation and resolution, (c) searches through similar incidents from the past and runbooks, and uses a language model to suggest possible causes and remedies, and (d) writes a postmortem document, using read-only guardrails and human approval for each action. Using a six-month replay of a platform of approximately 140 microservices, the authors found that the pipeline has reduced 47,968 raw alerts down to 9,757 incident candidates, or by 80%. The classification accuracy was 83.0% for 1,200 labeled incidents, compared with 74.0% for a fine-tuned BERT classifier ($p < 0.001$). In 71.5% of 400 incidents, the correct root cause was among the top three recommendations. In a controlled pilot with 300 assisted incidents and 300 baseline incidents, the mean time to resolve incidents was reduced from 78.4 to 52.2 minutes ($p < 0.001$), and the mean time to acknowledge from 9.8 to 4.1 minutes.

Keywords: Site reliability engineering; incident management; alert correlation; root cause analysis; large language models; MTTR; postmortem

Introduction

Operations are an integral part of the software process, which includes service level objectives, error budgets, and incident management (Beyer et al., 2016). However, in real life, SRE operations are associated with a flood of correlated alerts from different monitors, unclear ownership of incidents, and investigation through dashboards, logs, and historic tickets to find a solution that usually becomes evident during previous incidents.

The three major delays in the incident resolution process are (i) identification of the actual problem among many alerts, (ii) diagnosing the root cause of the issue, and (iii) deciding on how to address the problem in the most secure way. All these phases can be improved through pattern recognition over historic data. Large language models allow not only parsing the free-text incident notes, runbooks, and logs, but also composing explanations that could be easily checked by the engineers.

In this study, the authors evaluate the efficiency of such AI-assisted incident management in practice and describe ways to apply it safely. The authors provide the design of the pipeline, which incorporates guardrails, the evaluation of each step against the baselines, and a controlled pilot comparing assisted and unassisted incidents handling.

@2025. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): 3071-0154
Volume-1 Issue-2 || Apr-Jun 2025 || PP. 1-5

Literature Review

SRE literature highlights the importance of monitoring, incident response, and blameless postmortems (Beyer et al., 2016), and empirical research proves the link between operational capabilities and software development (Forsgren et al., 2018; Kim et al., 2016). Incident triage is the common problem that is discussed in the literature.

Specifically, Chen et al. (2019) conducted the investigation of incident triage in online service systems and found that incidents are frequently transferred between different teams until they are assigned to the team who is responsible for their resolution.

Automatic analysis of operational data had a long tradition. Chandola et al. (2009) conducted the survey of anomaly detection methods, while He et al. (2021) performed a review of log analysis for reliability engineering, including parsing, anomaly detection, and failure diagnosis. The DeepLog method utilized recurrent neural networks to detect anomalies in log sequences (Du et al., 2017), while Zhang et al. (2019) used the approach that allowed reducing the instabilities in the log data. However, these methods just find the symptoms of the issue but do not explain them using the company-specific terms.

Recently, language models have been used for incident management. Specifically, Ahmed et al. (2023) studied the language models in the generation of the explanations of the root cause and suggestions of the remediation of the cloud production incidents. They found that fine-tuned and retrieval-based models provided better results than prompt-based approaches. The use of BERT (Devlin et al., 2019) helped to improve the text classification. Retrieval augmentation allowed grounding generation in the documents that were not memorized by the model (Lewis et al., 2020).

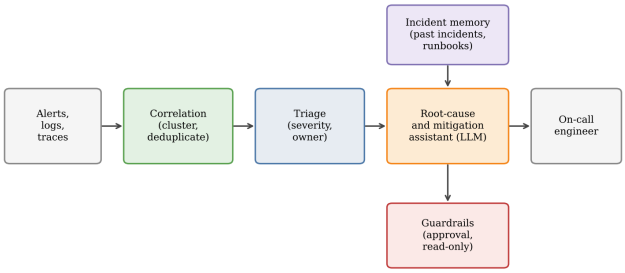
The joint effect of the correlation, triage, and recommendation on response times in a controlled comparison and safety requirements for the employment of the recommendations generated by the model in production are less discussed in the literature. The current study aims to fill the gap.

Research Methodology

Pipeline

Figure 1 shows the pipeline. Alerts, logs, and traces are ingested and correlated into incident candidates using temporal proximity, service dependency graph distances, and textual similarity of the alerts' annotations. A triage model classifies incident's severity and designates an owning team. For each incident, the system finds the most similar incidents from the past and the corresponding sections of the runbooks using the incident memory that is based on tickets and postmortems, and suggests a list of the probable causes along with the evidences and the remediation of the problem. The system does not make any changes in the production and does not use the data from it due to guardrails.

Figure 1. Pipeline for AI-assisted incident management with correlation, triage, retrieval-grounded recommendation, and guardrails.



Data and evaluation

The data consisted of 18 months of incidents with resolved root causes. The classification accuracy was evaluated on 1,200 held-out incidents, and the root-cause recommendations on 400 incidents with causes validated in postmortems using time-based splits to prevent the leakage of the data from the incident memory. The alert correlation experiment used the six-month alert stream. Pilot study design and measures

The authors conducted the comparison between 300 assisted incidents and 300 consecutive prior incidents without assistance, stratified by severity (40, 110, and 150 incidents of severities 1, 2, and 3, respectively).

Measures and analysis

For evaluation of the correlation efficiency, the authors used the reduction in the number of alerts for investigation and cluster purity. The classification accuracy was estimated using accuracy, and two-proportion z-tests and chi-square test were used for

@2025. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

International Journal of Advanced Research in Computer Science and Engineering (IJARCSE)
ISSN (Online): 3071-0154
Volume-1 Issue-2 || Apr-Jun 2025 || PP. 1-5

comparison of models. The recommendation quality was estimated as top-k accuracy to the true cause. Efficiency of incident handling was evaluated using mean time to acknowledge (MTTA) and mean time to resolve (MTTR); Welch's t-test was used for the comparison of the measures and effect size. The time spent on postmortem drafting was estimated for 24 reviews per arm.

Results

Alert correlation and triage

Using the six-month stream of alerts, correlation decreased the number of alerts down to 9,757 incident candidates from 47,968 raw alerts (the 80% reduction). The weekly volumes are shown in Figure 2. The manual examination of 200 clusters revealed that 94.1% of them consisted of alerts generated by a single underlying problem.

Figure 2. Raw alert volume and incident candidates after correlation per week.

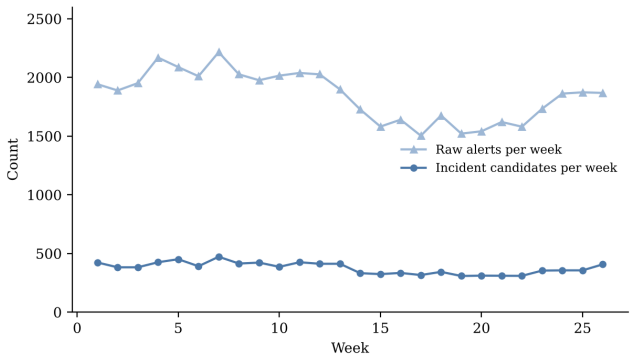


Table 1. Classification accuracy on 1,200 held-out incidents

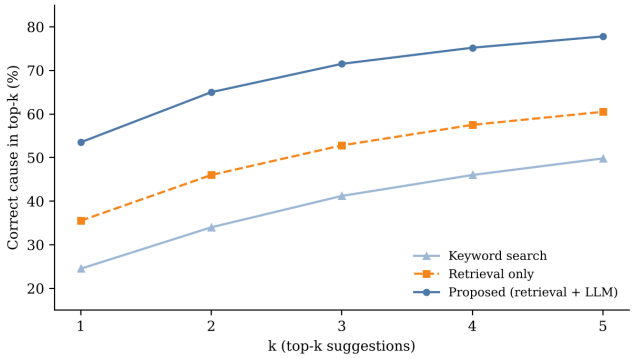
Model	Correct (n)	Accuracy (%)	p vs proposed
Rule-based	684	57.0	<0.001
TF-IDF + logistic	792	66.0	<0.001
Fine-tuned BERT	888	74.0	<0.001
Proposed	996	83.0	—

There was a difference in classification accuracy among models ($\chi^2(3) = 211$, $p < 0.001$; Table 1). The proposed model with the integration of text, service, and historical features with a language model achieved the highest accuracy of 83.0%, which is 9.0 percentage points higher than for the fine-tuned BERT model ($p < 0.001$) and 26.0 points higher than for the rule-based triage.

Root-cause recommendation

The correct root cause was included in the top three recommendations for 71.5% of incidents, compared with 52.8% for retrieval and 41.2% for keyword search across the tickets (Figure 3). Top-1 accuracy was 53.5%, compared with 35.5% and 24.5% for the respective baselines. The increase in performance with the growth of k was diminishing when $k > 3$, which supports the presentation of the list of three recommendations to the engineer.

Figure 3. Top-k accuracy of the root-cause suggestions for three approaches across 400 incidents.



Pilot: response times

Table 2. Mean time to resolve by severity (minutes, mean ± SD)

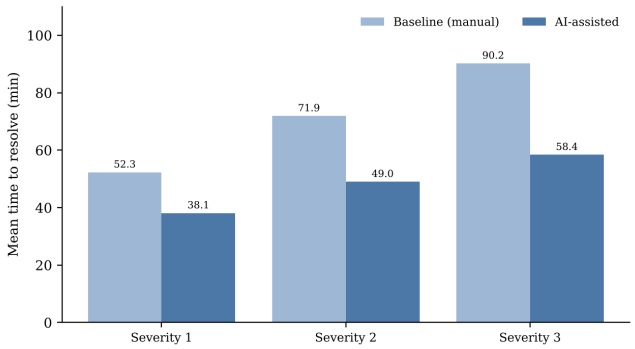
Severity	n per arm	Baseline	AI-assisted	Reduction (%)	p
Severity 1	40	52.3 ± 18.5	38.1 ± 14.2	27	<0.001
Severity 2	110	71.9 ± 26.4	49.0 ± 20.1	32	<0.001
Severity 3	150	90.2 ± 33.8	58.4 ± 25.3	35	<0.001
All	300	78.4 ± 32.3	52.2 ± 23.3	33	<0.001

Handling of incidents with assistance took 52.2 ± 23.3 minutes instead of 78.4 ± 32.3 minutes (Welch $t = 11.4$, $p < 0.001$, $d = 0.93$; Table 2, Figure 4), which made the resolution period 33% shorter. The reduction became statistically significant for all severities and more

@2025. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

prominent for severity 3 incidents (35% instead of 27%) because the first minutes of resolution are more dependent on the human coordination. The mean time to acknowledge was decreased from 9.8 ± 5.4 to 4.1 ± 2.3 minutes ($p < 0.001$).

Figure 4. Mean time to resolve by severity for baseline and AI-assisted handling.



Drafting a post-incident review took 61 ± 22 minutes with assistance versus 142 ± 38 minutes manually ($p < 0.001$). Engineers accepted the suggested remediation without any changes in 61% of cases, changed it in 27%. In 3 out of 300 assisted incidents, the engineers found the suggestion incorrect and rejected it before taking any actions; there were no cases when suggestions independently led to the change in the production system.

Discussion

Each stage of the pipeline has showed improvement in a way consistent with the intrinsic character of the problem. The decrease in duplicated effort can be explained by the reduction in correlation since service dependency and temporal relations account for a considerable part of alert co-occurrence. Gains in classification relative to BERT demonstrate that the structured operational features provide additional information beyond text (Devlin et al., 2019; Chen et al., 2019). Better performance of the recommendation over the retrieval indicates the advantage of a model capable of synthesis of evidence from several incidents; this approach aligns with Ahmed et al. (2023) and the principle of retrieval augmentation proposed by Lewis et al. (2020).

The proportionally larger gains for lower-severity incidents indicate the fields where automation can be used to advantage: routine incidents look like those in the past and benefit from memory, whereas high-severity issues include novel combination and need human coordination. This finding corresponds to the Site Reliability Engineering (SRE) approach, according to which automation should reduce toil and let engineers focus on judgment (Beyer et al., 2016). It also allows suggesting the rational deployment sequence: start with lower-severity services. Safety and speed should be taken into account equally. The recommendations were incorrect in some cases; therefore, guardrails – read-only access, approval gates, and logging – are necessary parts of the design instead of optional features. Postmortem drafting is a low-risk activity that, additionally, maintains an up-to-date incident memory and helps in keeping the culture of the blameless reviews.

Limitations

For the pilot study, the authors used consecutive-period comparisons, which introduces potential confounds because of the fluctuations in system load, staffing, or release cadence, although the stratification by severity and a constant on-call roster mitigate the risk. The data were obtained from a single platform and organization, and the quality of previous tickets' documentation affected the incident memory. The root cause accuracy was estimated against the conclusions in postmortems, which might be incomplete. The cost of inference for the model and impact of AI-assisted management on engineer skill development over time were not quantified.

Conclusion

The AI-assisted incident management pipeline with strict guardrails, consisting of alert correlation, severity classification, retrieval of analogous incidents, and drafting of recommendations has reduced alert noise by 80%, found the correct root cause in 71.5% of cases among the top-three recommendations, and made the mean time to resolve 33% shorter in the controlled pilot. The most prominent gains were achieved for routine incidents, accompanied by a small but real risk of incorrect recommendations that was kept under control through human approval. Future research will cover the randomized deployment, automatic remediation of the well-understood faults, and the impact of AI-assisted management on on-call workload.

OPEN ACCESS
@2025. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org

References

1. Ahmed, T., Ghosh, S., Bansal, C., Zimmermann, T., Zhang, X., & Rajmohan, S. (2023). Recommending root-cause and mitigation steps for cloud incidents using large language models. Proceedings of the 45th International Conference on Software Engineering (ICSE), 1737–1749.
2. Beyer, B., Jones, C., Petoff, J., & Murphy, N. R. (Eds.). (2016). Site reliability engineering: How Google runs production systems. O'Reilly Media.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), Article 15.
4. Chen, J., He, X., Lin, Q., et al. (2019). An empirical investigation of incident triage for online service systems. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 111–120.
5. Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT, 4171–4186.
6. Du, M., Li, F., Zheng, G., & Srikumar, V. (2017). DeepLog: Anomaly detection and diagnosis from system logs through deep learning. Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 1285–1298.
7. Accelerate: The science of lean software and DevOps. IT Revolution Press.
8. He, S., Zhu, J., He, P., & Lyu, M. R. (2021). A survey on automated log analysis for reliability engineering. ACM Computing Surveys, 54(6), Article 130.
9. Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps handbook. IT Revolution Press.
10. Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.
11. Zhang, X., Xu, Y., Lin, Q., et al. (2019). Robust log-based anomaly detection on unstable log data. Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 807–817.

@2025. This is an open access article distributed under the terms of the Creative Commons License [CC BY NC 4.0] and is available on www.ijarcse.org