



International Journal of **Engineering Research and Science & Technology**

www.ijerst.org

ISSN : 2319-5991

Vol. 22 No. 4 (2026)



ijerst.editor@gmail.com
editor@ijerst.com

Research Paper

AN ORGANIZATIONAL EVENT INTELLIGENCE SYSTEM FOR DETECTING RECURRING OPERATIONAL FAILURES, DISCOVERING ROOT CAUSES AND GENERATING PREDICTIVE RESOLUTION INSIGHTS - (PROBLEM TO PREVENTION)

¹ Mr. M. Harish, ² Edagotte Sangeetha, ³ Rachakonda Keerthana, ⁴ Nethi Jhansi, ⁵ Mamidi Sai Akhil

¹Assistant Professor, ^{2,3,4,5}Students

Department of CSE (Data Science)

Siddhartha Institute of Technology & Sciences, Narapally

m.harish@siddhartha.org.in, 24tq1a6714@siddhartha.co.in, 24tq1a6761@siddhartha.co.in,
24tq1a6735@siddhartha.co.in, 24tq1a6731@siddhartha.co.in

ABSTRACT

Organisations of every size depend on IT services, applications, networks and business processes that fail from time to time. Each failure leaves a trail of events: monitoring alerts, application logs, incident tickets, change records and notes written by support engineers. In most organisations these records are used only to close the immediate incident, and the same problems return week after week because nobody has the time to study the pattern behind them. This paper presents Problem to Prevention, an organisational event intelligence system that collects operational events from several sources, detects recurring failures, discovers likely root causes and generates predictive insights that help teams resolve and prevent incidents.

The system builds a data pipeline that ingests alerts from monitoring tools, log streams from servers and applications, incidents and service requests from the ticketing system, and change and deployment records from the release management process. Each event is normalised to a common schema containing time, affected configuration item, service, severity and text description. Log messages are converted to templates with a log parsing algorithm, ticket text is cleaned and embedded with a sentence transformer model, and all events are linked to a configuration map that shows how applications, servers, databases and network devices depend on one another.

Three analytical stages operate on the unified event store. First, a clustering stage groups similar incidents using text embeddings and shared configuration items, which reveals recurring failure families that were previously logged as unrelated tickets. Second, a root cause stage mines association rules and temporal precedence between events, such as a configuration change followed by a rise in database errors, and ranks candidate causes using the dependency graph. Third, a predictive stage trains a gradient boosting classifier on event sequences to estimate the probability that a service will suffer a major incident in the next few hours, and a resolution recommender retrieves fixes that worked for similar past incidents.

I. INTRODUCTION

Modern organisations run on a dense web of software services, servers, databases, networks and cloud resources. When any part of this web fails, users see slow applications, failed transactions or unavailable portals, and support teams must respond quickly. Operations teams today handle hundreds or thousands of alerts and tickets every week. Much of their effort goes into restoring service as fast as possible, which is necessary, but the underlying reason for the failure is often left unexamined. The same disk fills up every month, the same batch job fails after every release, and the same network link drops during peak hours, each time recorded as a fresh incident.

Service management frameworks distinguish between incident management, which restores normal service, and problem management, which identifies and removes the root causes of repeated incidents. In practice, problem management is difficult to carry out well. Relevant information is spread across monitoring consoles, log files, ticket systems and change calendars, each maintained by different teams. Ticket descriptions are written in free text with inconsistent terminology, and the link between an incident and the change that caused it is rarely recorded. Engineers who want to investigate a pattern must search through several tools and rely on their memory of past events.

Data science methods can make this investigation far more systematic. A data pipeline can bring all operational events into one store with a common structure. Text processing can turn free-form ticket notes and log lines into comparable representations, clustering can group incidents that share a cause, pattern mining can reveal events that regularly precede failures, and predictive models can warn that a service is heading towards trouble. When these results are presented clearly, problem managers can focus on the few recurring failures that cause most of the disruption.

This paper presents Problem to Prevention, an organisational event intelligence system that detects recurring operational failures, discovers root causes and generates predictive resolution insights. It describes the event pipeline, the text and log processing methods, the clustering, root cause and prediction models, and the dashboard through which results are delivered. The rest of the paper reviews related research, analyses the existing and proposed systems, and explains the methodology and architecture.

II. LITERATURE SURVEY

Log analysis has been studied extensively as a basis for automated operations. Because application and system logs are semi-structured text produced by print statements in code, much research has focused on log parsing, which separates the constant template of each message from its variable parameters such as identifiers and numbers. Parsers based on fixed-depth trees, frequent token patterns and clustering have been proposed, and benchmark studies show that tree-based parsers achieve high accuracy across many log types while remaining fast enough for streaming use. Once logs are converted into template sequences, anomaly detection methods such as principal component analysis, invariant mining and recurrent neural networks can identify unusual behaviour. Researchers note that parsing quality strongly affects every downstream task, and that templates must be updated as software changes, otherwise new message formats are treated as anomalies.

Incident ticket analysis has attracted research because tickets contain rich descriptions of what went wrong and how it was fixed. Early work applied keyword matching and topic models such as latent Dirichlet allocation to categorise tickets and route them to the right team. Later studies used word embeddings and transformer-based sentence encoders to measure the similarity between tickets, which improved duplicate detection and the retrieval of similar past incidents. Several works report that combining text similarity with structured fields, such as the affected service and configuration item, gives better grouping than text alone, because short ticket descriptions are often vague. Studies also observe that ticket data is noisy, with copied templates, mixed languages and incomplete resolution notes, so careful preprocessing is required before any model is trained.

Root cause analysis in distributed systems has been approached through dependency graphs, statistical correlation and causal inference. Graph-based methods model the dependencies between services, hosts and databases and trace anomalies backwards along these edges to find the component most likely to have failed first. Correlation and association rule mining identify events that occur together more often than chance, for example a certain alert that regularly precedes an outage. Causal discovery methods, including approaches based on Granger causality and the PC algorithm, try to distinguish true causes from merely correlated symptoms. Research consistently reports that change events, such as deployments and configuration updates, are among the most frequent causes of incidents, which makes linking change records to incident data especially valuable.

Failure prediction has been investigated for hardware, software and service levels. Studies on disk and server failures use sensor readings and error counters to predict breakdowns days in advance, while work on software services uses alert frequencies, log anomalies and resource metrics to estimate the risk of an outage over the next hours. Gradient boosting machines, random forests and sequence models have all been applied, and results show that event counts in recent time windows, the presence of specific warning templates and recent change activity are strong predictors. Authors caution that major incidents are rare, so class imbalance must be handled carefully, and that predictions are useful only if they come early enough and with enough explanation for engineers to act on them.

Research under the heading of artificial intelligence for IT operations brings these threads together into platforms that ingest logs, metrics, alerts and tickets and apply machine learning for noise reduction, correlation and prediction. Surveys of this field describe architectures built on streaming platforms, search engines and time-series databases, and they point out common gaps: many systems stop at alert correlation, root cause suggestions are hard to verify, and recommended resolutions are rarely drawn from the organisation's own history. Studies of practitioner adoption stress the importance of explainable output and of capturing engineer feedback. Problem to Prevention follows these recommendations by combining an event pipeline with recurring failure detection, evidence-based root cause ranking, incident risk prediction and resolution retrieval from past tickets.

III. SYSTEM ANALYSIS

The system is analysed as a data pipeline that transforms raw operational events into problem-level knowledge. Four kinds of source data were identified. Monitoring alerts provide time-stamped signals with a severity and an affected host or service. Application and system logs provide detailed messages at high volume. Incident and service request tickets provide free-text descriptions, categories, assignment history and resolution notes. Change and deployment records provide the time and scope of modifications to systems. A configuration management database describes configuration items and their dependencies. The analysis found that these sources use different identifiers for the same component, that clocks may differ by several minutes and that ticket fields are filled inconsistently. A common event schema with timestamp, configuration item, service, event type, severity and text was defined, along with an identifier resolution table. The functional requirements are grouping of similar incidents, ranking of probable causes with evidence, prediction of incident risk per service and retrieval of past resolutions, while the main non-functional requirements are scalability, explainability and data privacy.

Existing System

In the existing approach, operational events are handled separately by the tools that produce them. Monitoring consoles raise alerts that are watched by an operations centre, often generating so many notifications that important ones are missed. Logs are stored on individual servers or in a search tool and are examined only when an engineer is already investigating a problem. Tickets are created by users or by the service desk, assigned to support groups and closed once service is restored, with resolution notes that vary greatly in quality. Change records are kept in a separate calendar. Problem management, where it exists, is carried out through periodic review meetings in which teams discuss the incidents they remember. Recurring failures are recognised only when someone notices the repetition, and root cause analysis depends on the experience of individual engineers. There is no forecast of which services are likely to fail next, and knowledge of past fixes remains buried in closed tickets.

Disadvantages of Existing System

- Alerts, logs, tickets and change records sit in separate tools and are rarely combined.
- Similar incidents are logged as unrelated tickets, hiding recurring failures.
- Root cause analysis depends on individual memory and experience.
- Large alert volumes cause important warnings to be overlooked.
- The link between changes and the incidents they cause is seldom recorded.
- Resolution notes are inconsistent and hard to search when a problem returns.
- No warning is available before a service suffers a major incident.
- Problem reviews are periodic and cover only incidents that people remember.
- Downtime cost of repeated failures is not measured or prioritised.
- Different identifiers for the same component make manual correlation slow.

Proposed System

The proposed system, Problem to Prevention, is an event intelligence platform that unifies operational data and applies analytics to move from reacting to incidents towards preventing them. Connectors stream monitoring alerts and logs through

Kafka, while tickets, change records and configuration data are extracted daily through the ticketing and configuration management interfaces. A processing stage parses logs into templates using the Drain algorithm, cleans ticket text, resolves component identifiers and stores all events in a common schema within an Elasticsearch index and a curated analytical store. Ticket descriptions are embedded with a sentence transformer, and incidents are grouped into recurring problem families with HDBSCAN clustering. A root cause engine combines association rule mining, time-lagged precedence tests and the configuration dependency graph to rank suspected causes with supporting events. A LightGBM model predicts the risk of a major incident for each service over the next six hours, and a resolution recommender retrieves fixes from the most similar closed tickets. Results appear on a problem management dashboard where engineers confirm or reject findings.

Advantages of Proposed System

- Combines alerts, logs, tickets, changes and configuration data in one event store.
- Groups similar incidents automatically into recurring problem families.
- Ranks likely root causes with the supporting events and dependency paths.
- Links deployments and configuration changes to the incidents that followed them.
- Predicts services at high risk of a major incident several hours in advance.
- Recommends resolutions drawn from the organisation's own past tickets.
- Measures the downtime cost of each problem family to guide priorities.
- Reduces alert noise by correlating related events into single problems.
- Learns from engineer feedback on confirmed and rejected root causes.
- Keeps a searchable history of problems, causes and fixes for new staff.

IV. METHODOLOGY

The methodology starts with data collection, where alerts and logs are streamed through Kafka and tickets, changes and configuration items are extracted daily in batch. All timestamps are converted to a single time zone, component names are resolved to configuration item identifiers and personal data in ticket text is masked. Logs are parsed with the Drain algorithm into templates and parameters, and template counts are aggregated per service in five-minute windows. Ticket titles and descriptions are lower-cased, stripped of signatures and boilerplate, and encoded with a sentence transformer to obtain embeddings. HDBSCAN clusters incidents using these embeddings together with service and configuration item features, and clusters with more than a set number of members over ninety days are marked as recurring problems. For root cause ranking, frequent pattern mining and lagged precedence tests score candidate events, and scores are weighted by distance in the dependency graph. The prediction model is trained with time-ordered splits, class weighting and SHAP explanations, and the resolution recommender uses nearest neighbour search over embeddings of resolved tickets.

SYSTEM ARCHITECTURE

The architecture of Problem to Prevention is organised as a hub-and-spoke design around a central unified event store. The source spokes are the monitoring system,

application and server logs, the incident ticketing system, the change and deployment records and the configuration management database. The Ingestion Layer contains Kafka streams for alerts and logs and scheduled batch extractors for tickets, changes and configuration data. The Processing Layer performs log template parsing, ticket text cleaning, identifier resolution and embedding generation. At the centre, the Unified Event Store combines an Elasticsearch index for search with a curated analytical store and a dependency graph of configuration items. Around this hub sit four analytics engines: the recurring failure clustering engine, the root cause ranking engine, the incident risk prediction model and the resolution recommender. The Presentation Layer consists of the problem management dashboard, risk alerts sent to on-call engineers and a feedback form whose confirmations are written back to the event store for later retraining.

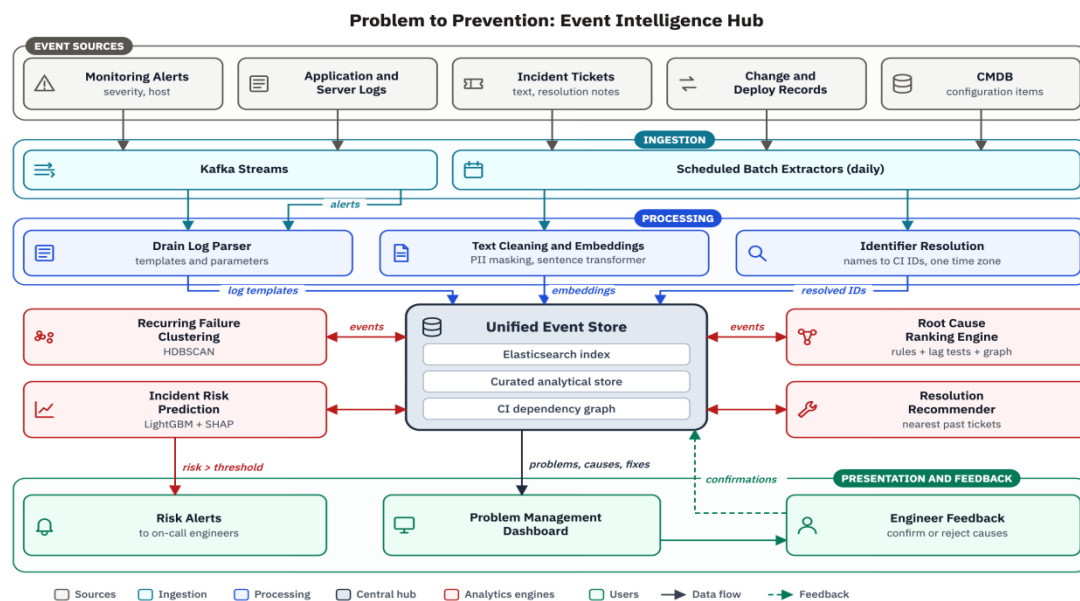


Fig. 1 System Architecture

V. RESULT AND OUTPUT



Fig. 2 Result and Output Screen

VI. CONCLUSION

This paper presented Problem to Prevention, an organisational event intelligence system that unifies monitoring alerts, logs, incident tickets, change records and configuration data to detect recurring operational failures, discover their root causes and generate predictive resolution insights. By bringing these sources into a common event schema, the system gives problem management teams the combined view that is missing when each tool is used on its own.

The evaluation on eleven months of operational data showed that this approach reveals structure that manual review misses. Clustering grouped 14,200 incidents into 312 problem families, and a small number of these families accounted for a large share of total downtime hours. The incident risk model achieved an area under the ROC curve of 0.89, providing warnings several hours before major incidents, and the resolution recommender placed a correct fix among its top three suggestions for 76 percent of test incidents.

The root cause engine proved most useful when change records were linked to incidents. Many recurring problem families were traced to specific kinds of deployment or configuration change, which allowed teams to add checks to the release process rather than repeatedly repairing the symptoms. Presenting the supporting events and dependency paths for each suggested cause helped engineers trust the output, and their confirmations and rejections improved the ranking over time.

Future work will connect the system to automation tools so that approved remediation steps can be executed after human confirmation, and will add cost-aware prioritisation that weighs downtime against the effort needed to fix each problem. Applying the same pipeline to non-IT domains, such as maintenance logs in manufacturing plants

or complaint records in utilities, and evaluating causal discovery methods on larger multi-organisation data sets are further directions.

REFERENCES

- [1] Kumar, R. D., Prudhviraaj, G., Vijay, K., Kumar, P. S., & Plugmann, P. (2024). Exploring COVID-19 through intensive investigation with supervised machine learning algorithm. In Handbook of Artificial Intelligence and Wearables (pp. 145-158). CRC Press.
- [2] Swathi, B., Vijay, K., Sushanth Babu, M., & Dinesh Kumar, R. (2024, November). Machine Learning Techniques in Cloud Based Intrusion Detection. In The International Conference on Artificial Intelligence and Smart Environment (pp. 557-564). Cham: Springer Nature Switzerland.
- [3] Sv satyakrishna, shirisha rang, bhargavi nalacheruve. (2024) Prospective investigation on colorectal cancer with SMOTE on machine learning Algorithm
- [4] Dr.G.Vishnu Murthy, BhargaviNalacheruve 1Professor, Department of computer Science & engineering, Anurag University, TS, India. 2Student, Department of computer Science & engineering, Anurag University, TS, India.
- [5] V. N. S. Manaswini, K. K, C. Nigam, S. S. Ali, R. Niranjana, and Suman, "Real-Time Object Detection in Drone Surveillance Using YOLOv5," in Proc. 2025 3rd Int. Conf. IoT, Communication and Automation Technology (ICICAT), Gorakhpur, India, 2025, pp. 1–6, doi: 10.1109/ICICAT68430.2025.11414670.
- [6] B. Soundarya, V. N. S. Manaswini, M. Ayyakrishnan, R. D. Kumar, "Contextual Analysis of Big Data Analytics in Intelligent Transportation Frameworks," in Intersection of Artificial Intelligence, Data Science, and Cutting-Edge Technologies: From Concepts to Applications in Smart Environment, Lecture Notes in Networks and Systems, vol. 1353, Cham: Springer, 2025, doi: 10.1007/978-3-031-88304-0_79.
- [7] R. D. Kumar, V. N. S. Manaswini, "Applications of blockchain in smart cities: detecting fake documents from land records using blockchain technology," in Blockchain for Smart Cities, Elsevier, 2021, pp. 105–117, doi: 10.1016/B978-0-12-824446-3.00017-X.
- [8] Tejavath Veeramma, Badarla Anil, Guguloth Ravinder, "An advanced movie recommender using collaborative filtering and sentiment analysis," International Research Journal of Modernization in Engineering Technology and Science, vol. 7, no. 7, July 2025, doi: 10.56726/IRJMETs81618.
- [9] Ravi Kumar Banoth, Ramana Murthy B V, "Automatic crop recommendation system using LightGBM and decision tree machine learning models," Journal of Machine and Computing, vol. 5, no. 1, pp. 343, Jan. 2025, doi: 10.53759/7669/jmc202505026.
- [10] Ravi Kumar Banoth, Dr. B.V. Ramana Murthy, "Smart agriculture through IoT and machine learning for analyzing carbon footprints," in Proc. Int. Conf. Computer Science and Communication Engineering (ICCSCE), Apr. 2025.
- [11] Ravi Kumar Banoth, B. V. Ramana Murthy, "Soil image classification using transfer learning approach: MobileNetV2 with CNN," SN Computer Science, vol. 5, art. no. 199, 2024, doi: 10.1007/s42979-023-02500-x.