

An Explainable Causal Temporal Framework for Failure Prediction in Event-Driven Distributed Systems Using Logs, Metrics, Traces, and Event Streams

Durga Narayana Varma Addepalli
Independent Researcher, USA

Abstract

Distributed systems based on events are becoming more problematic in terms of their reliability as a result of the growing complexity of interactions between microservices, message queues, databases, and infrastructure elements. This study presents an Explainable Causal Temporal Failure Prediction Framework, a combination of logs, metrics, traces, and event streams to predict incidents proactively. The frameworks represent temporal dependencies and propagation of causal failures with the help of LSTM, Transformer, and Causal Graph Neural Networks (CGNN). Failure prediction, early warning capability and evaluation of root cause analysis are tested in synthetic cloud-native environments. This new method enhances operational intelligence through the detection of failure signs, prediction descriptions, and prediction reliability management of a distributed system.

Keywords: *Causal Failure Prediction, Event-Driven Systems, Distributed Systems, Logs, Metrics, Traces, Event Streams, Explainable AI, LSTM, Transformer, Causal Graph Neural Network, Predictive Reliability Management*

I. INTRODUCTION

Background

The rapid expansion of cloud-native apps has already revolutionized the software delivery using microservices, event-based architecture, message queues, distributed databases and serverless ensemble [1]. However, the growing number of interdependencies between services complicate the diagnosis of failures as well. As an illustration, a slow consumer may add to the queue depth, inducing and propagating retries and resource consumption, which may propagate failures to other parts that would depend on it, such as databases, producers and downstream services. Such interactions suggest that it might be inadequate to monitor individual infrastructure indicators of how the state of operations becomes cascading failures. Thus, the determinants of predicting failures must be both the time variations of observability signals and the connections among the distributed elements.

Research Problem

Traditional monitoring in the distributed system is usually done based on the alert of threshold, anomaly detection and management of incidents that happens.

While these techniques prove useful in determining the abnormal operational conditions, they help in limited degree in identifying whether an ever evolving condition is inclined towards the formation of a system failure or how a failure can expand over the dependent components. To illustrate this, high CPU utilisation can be used to mean that there is pressure on resources but then there is no single point in isolation that can explain why the condition will continue to trigger an increase in the queue, retrying, or overload the database or cause downstream service failure [2]. The gap in the research is thus that very little is done to combine the aspects of predicting the temporal failure, the causal dependency analysis, heterogeneous observability data and explainable solely in one platform proactive failure detection. Current surveillance methods can study the individual metrics, logs or traces, whereas the correlation between the changing operational signatures and dependency between the components needs additional research. This gap is filled in this study by integrating multimodal observability data, with temporal prediction and causal dependency modelling, with the aim of extending the scope of understanding on whether failure conditions could be predicted prior to an observed service failure.

Research Aim and Objectives

Research Aim

The study seeks to implement a causal temporal model that can be explained and predict failures in event-led distributed frameworks through combination of metrics, logs, trace, and also event streams to prevent incidents proactively.

Research Objectives

- *To create a causal model of expressing temporality of relationships between components in event-led distributed frameworks.*
- *To combine logs, metrics and traces with event streams to analyze failures holistically.*
- *To discover early signs of retry storms, resource saturation and cascading failures.*
- *To develop a predictive framework that can be explained to be proactive incident detection and prevention.*

Novel Contribution

10.48047/jocaaa.2024.33.04.57

The study provides an explainable causal time model that combines diverse observability information, involves logs and metrics, traces, and event streams, to anticipate failures causing service failure [3]. Post hoc, contrasting with the classical methods of anomaly-fueled, the principal frameworks score causal relationships between distributed constituents, detects patterns of failure propagation, and gives feedback-describable forecasts to proactively control reliability through anticipating events in event-driven frameworks.

Current methods used to predict failures in distributed systems predominantly concentrate on investigations of anomalies, separate telemetry providers or on post-failure diagnostics. However they usually do not have the capability to compute dynamic causal relationships between distributed elements and at the same time include heterogeneous observability data. Hence, this study fulfills the gap by introducing an explainable causal-temporal model that merges logs and metrics, traces, and event streams with the causal-temporal learning models (LSTM and Transformer) and causal dependency modelling models (CGNN). Predicting failures is only one part of the key contribution of this work which also gives explanations of the propagation paths of failure that can be read out.

II. RELATED WORK

Causal Temporal Modeling of Component Relationships in Event-Driven Distributed Systems Reliability Prediction

ICAD Framework Overview

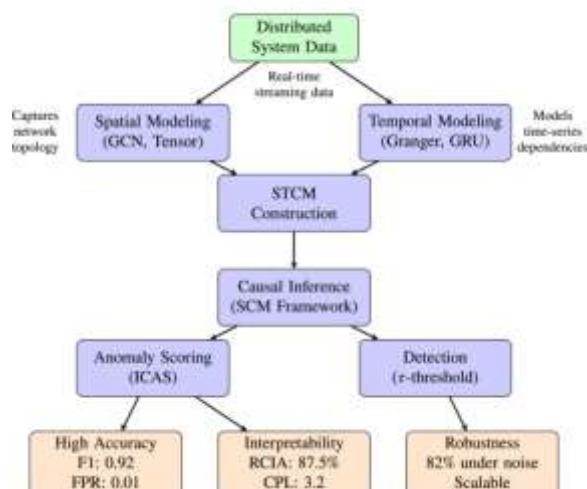


Fig 1. Existing Approaches and Research Gap in Causal Temporal Modeling of Distributed System Dependencies

Previous work on distributed system reliability has delved into dependency modelling and temporal analysis through graphs and probabilistic models and machine learning methods [4]. Although approaches commit service interactions, most of them are

concerned with the correlation and not causation [5]. Existing causal models are not adaptable and dynamic with events that happen in real-time [6]. Thus, there is a gap in the current literature in terms of developing scale causal time models that can model a changing relationship between distributed elements to predict failures proactively.

Multimodal Integration of Logs, Metrics, Traces, and Event Streams for Failure Analysis

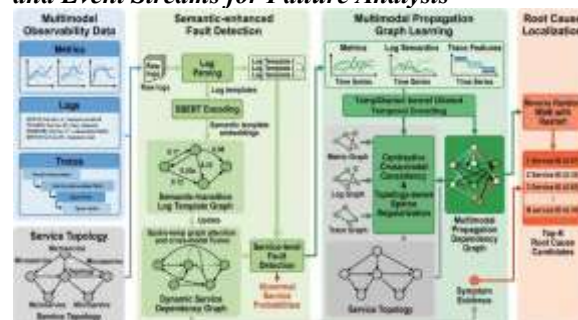


Fig 2. Multimodal Observability Data Integration Framework for Distributed System Failure Analysis

Recent works highlight observability-based methods through analysis of logs, metrics, traces and event streams to enhance the monitoring of distributed systems [7]. The majority of the current methods analyze these data sources independently and cannot adequately grasp cross-layer failure patterns [8]. Despite multimodal approaches enhancing accuracy of detection, there are difficulties in synchronizing data, correlating features, and within real time [9]. More studies are needed to assess cohesive models of heterogeneous operational signals to enable a holistic study of failures.

Early Detection of Retry Storms, Resource Saturation, and Cascading Failures through Indicators

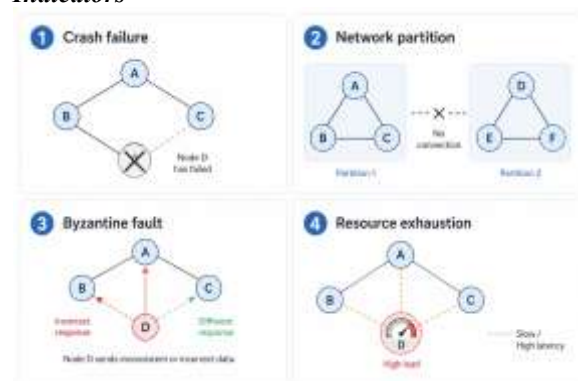


Fig 3. Failure Propagation Patterns and Early Warning Indicators in Event-Driven Distributed Systems

Failure prediction studies have assessed failure detection, resource tracking and workload forecasting to detect risks in operation [10]. Nevertheless, current

solutions often recognize symptoms while degradation has taken place instead of anticipating ways of non-propagation of failure [11]. Composite temporal dependencies are found within retry storms, queue congestion, as well as resource saturation, and are poorly represented by traditional approaches [12]. Finally, existing literature presents the necessity of more sophisticated predictive models that are able to detect an early warning sign and avoid the cascading failure [13].

Explainable Failure Prediction Framework for Proactive Incident Detection and Prevention in Systems

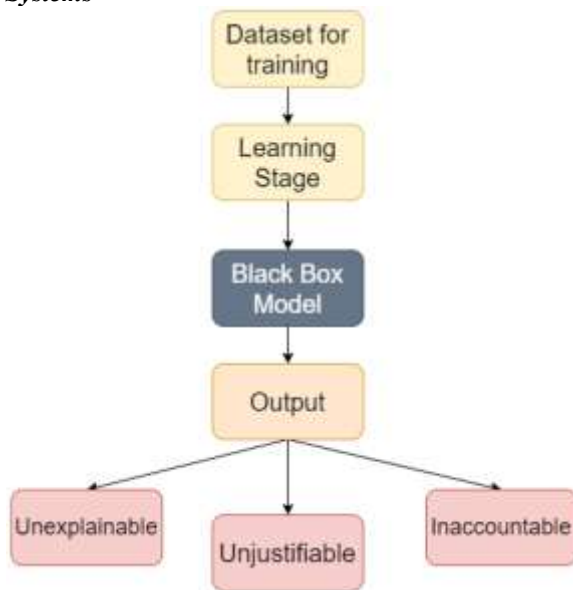


Fig 4. Explainable Causal Failure Prediction

Framework for Proactive Incident Management

Frameworks for predicting failures utilizing machine learning have been demonstrated to be adequate at predicting abnormal system behavior, while many of them are black-box models that are not as interpretable [14]. Current studies have the prediction accuracy while lack adequate interpretation of the reasons of failure as well as the spread patterns [15]. This constraint lessens the trust and decision-making powers of operation [16]. The growth of future research needs to have exploratory predictive frameworks which incorporate causal rationality with predictive intellect in order to assist in proactive incident discovery and automated reliable administration of distributed frameworks [17].

Research Gap

Current predication failure strategies in the distributed system aim at anomaly detection and individual data examination but do not have causal insights into intricate interactions of services [18]. Existing solutions have difficulty in uniting logs, metrics, traces, and event streams along with offering explainable predictions [19]. Hence, a major research

gap is the lack of widespread studies on creating a scaffold causal temporal model of proactive failure prediction in event-led distributed frameworks.

III. PROPOSED CAUSAL FAILURE PREDICTION FRAMEWORK

System Architecture

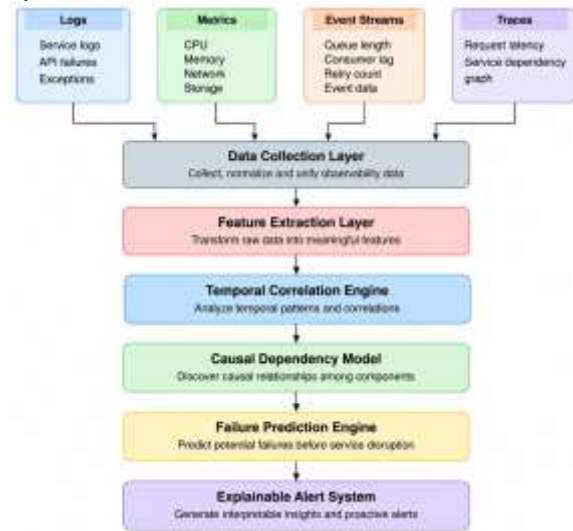


Fig 5. End-to-End Architecture of the Explainable Causal Failure Prediction Framework

The suggested structure brings various sources of observability through the utilization of a layered architecture [20]. Event streams, logs, metrics and traces are gathered and processed to assess meaningful features. Patterns of behavior are recognized by a temporal correlation engine and relationships of propagation of failures are recognized by a causal dependency model [21]. The prediction engine failure tool produces early warnings, which is aided by an explainable alert system to assist an operational decision-making framework.

Data Collection Layer

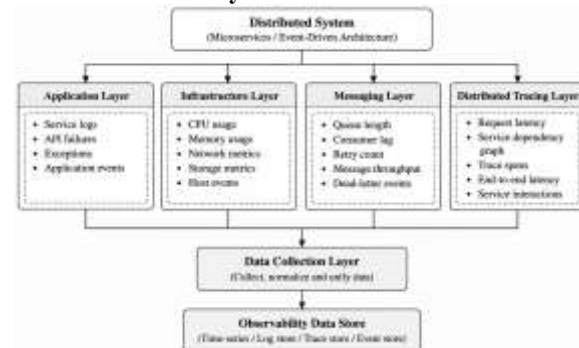


Fig 6. Multisource Data Collection Architecture for Distributed System Monitoring

The mentioned data collection layer collects diverse operational data of applications, infrastructure, messaging along with tracing of environment [22]. Application logs contain service logs or api failures

and exception logs, whereas infrastructure logs contain CPU, memory, network as well as storage behavior [23]. Messaging information tracks the length of queues, consumer lag and retries. The distributed traces can give measurements of latency as well as service dependency relations.

Feature Engineering

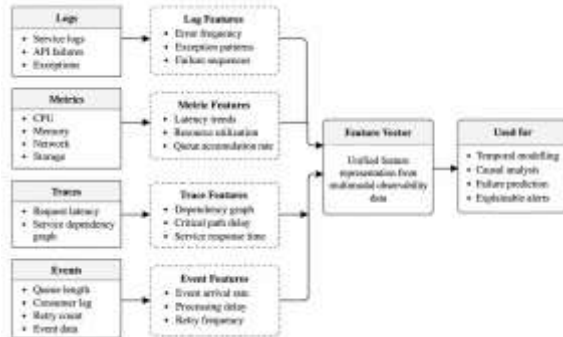


Fig 7. Feature Extraction Process from Multimodal Observability Data

Predictive indicators are features derived through feature engineering to analyze failure. Log features assist to record frequency of errors, pattern of exceptions and order of failure [24]. Metric features depict trends within latency, changes in resource utilization as well as queue growth [25]. Trace features define dependency relationships as well as critical delays, whereas event features determine the arrival rates, processing delays along with retry rates to predict time.

Temporal and Causal Modeling

LSTM

LSTM is designed to capture the temporal dynamics of observations in systems since the symptoms of failure tend to manifest over time by alterations in latency, retries, and resource usage. LSTM is able to remember significant historical data and ignore short-term fluctuations thanks to the memory mechanism.

Transformer

The use of transformer models is made possible as attention mechanisms could recognize significant relationships between events at varying time intervals. Transformers, in contrast to recurrent models, can run long sequences faster, and model more intricate temporal dependence between distributed signals in a system.

CGNN

CGNN builds upon the learning of GNNs and adds causal relationships between parts of the system to it. Reading between the lines, the nodes are interpreted as distributed components, such as services, queues and databases, whereas the edges are interpreted as potential propagation of failures paths. CGNN gives approximations on the ability of failures to propagate using the dependencies between components.

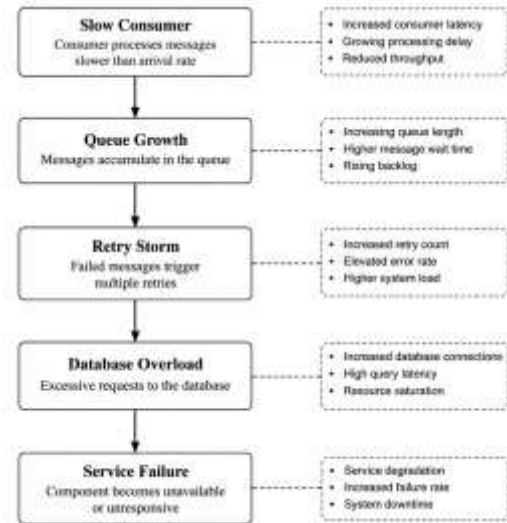


Fig 8. Causal Dependency Graph for Failure Propagation in Event-Driven Systems

Model employs both the temporal along with causal modelling to learn evolution of failures of distributed frameworks [26]. Temporal predictors Temporal models involving Temporal Convolution Networks, LSTM and also Transformers are models that describe sequential behavior and make predictions [27]. Bayesian Networks, Structural Causal Models, as well as Causal Graph Neural Networks are causal models, which determine dependency relationship and paths of errors spread to service failure by slow consumers.

Explainability Layer

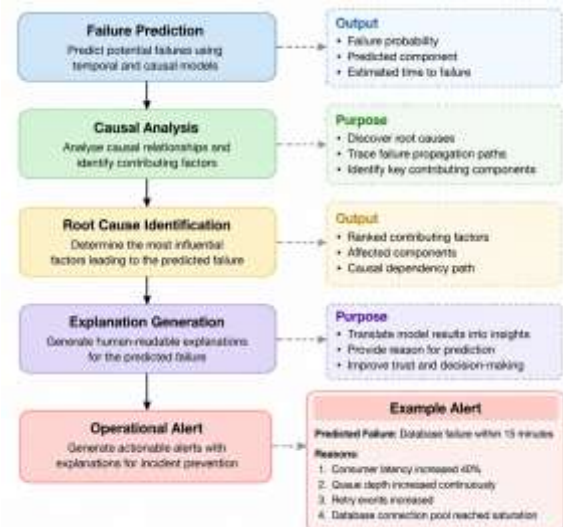


Fig 9. Explainable Prediction and Alert Generation Mechanism

The Explainability layer allows provision of reasons in a form that is explainable, enhances operational trust and decision-making [28]. Framework determines causal factors in addition to creating failure probabilities, involving consumer latency increased,

10.48047/jocaaa.2024.33.04.57

queue continually growing, more and more retry events, and saturation of the database [29]. This allows engineers to track the cause of failure, to prioritize corrective measures and to proactively prevent the occurrence of a series of incidences.

IV. RESEARCH METHODOLOGY

Experimental Environment

The recommended framework needs to be tested on a simulated cloud-native distributed environment according to microservices, Kafka/RabbitMQ message brokers and Kubernetes orchestration, distributed databases, and monitoring features. Various services with various workload conditions can offer operation signals to the system. “*Temporal Convolutional Network (TCN)*”, “*Long Short-Term Memory (LSTM)*”, “*Transformer*”, and also “*Causal Graph Neural Network (CGNN)*” machine learning models can be utilized on temporal prediction along with causal dependency modelling. The state of the system is:

$$St = \{Lt, Mt, Tt, Et\}$$

Where logs are denoted by Lt , metrics are denoted by Mt , traces are denoted by Tt , and events are denoted by Et and t shows observation time for the respective measures.

Failure Injection

Directed failure cases can be incorporated into a synthetic distributed environment to test the prediction capacity in a practical operation. Failures can be application level as well as messaging level (slowdown and queue overflow of consumers) failures [30]. CPU saturation, memory pressure can be aspects of infrastructure failures.

Failure states are represented as causal transitions within the system components based on a dependency graph,

$$G = (V, E)$$

Where V denotes distributed elements, E denotes causal relationships affecting failure propagation.

Dataset Generation

Synthetic data can be created through running on experimental environment data on multimodal observability, involving logs, metrics, distributed traces, as well as event streams. Every observation window is represented as failure or normal state using binary classification:

$$y_i \in \{0,1\}$$

Where 0 represents normal operation, 1 represents a failure condition. Extracted features is developing input vectors,

$$X_t = [x_1, x_2, \dots, x_n]$$

Which can be evaluated by Transformer, LSTM, TCN, and CGNN models to learn temporal trends along with causal failure relationships.

Evaluation Metrics

Classification as well as predictive reliability measures can be utilized to measure suggested frameworks. The precision, accuracy, recall, as well as F1-score can be utilized to measure performance of prediction. Accuracy is investigated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Where precision is represented as:

$$Precision = \frac{TP}{TP + FP}$$

Recall as:

$$Recall = \frac{TP}{TP + FN}$$

F1-score is measured as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Early warning capability is evaluated using prediction lead time

$$Tl = Tf - Tp$$

Where Tf denotes failure occurrence time, Tp denotes prediction time. System reliability improvements can be evaluated by minimizing incidents along with minimization of recovery time.

Pseudocode

```

Algorithm 1
Explainable Causal Failure
Prediction Framework

Initialize

    Observability Data

        Logs

        Metrics

        Traces

        Event Streams

    Models

        LSTM

        Transformer

        Causal Graph

Neural Network

    Failure Threshold = 0.5
  
```

```

Start loop

    Collect system
    observations

    IF Log error rate
    increases THEN
        update failure
        features

    IF Queue depth increases
    AND Consumer latency increases
    THEN
        identify possible
        retry storm

    IF Resource utilization
    exceeds threshold THEN
        update failure
        probability

    Extract temporal features

    Construct causal
    dependency graph

    Apply Temporal Prediction
    Model

    Apply Causal Graph Model

    Calculate Failure
    Probability

    IF Failure Probability >
    Threshold THEN

        Generate Failure
        Alert

        Identify Root
        Cause

        Generate
        Explanation

```

```

ELSE
    Continue

Monitoring

    Update system state

End loop

```

Fig 10. Algorithm for Explainable Causal Failure Prediction Framework

The proposed explainable causal failure prediction procedure is described in Algorithm 1 and unites the concept of multimodal observability data, time learning, causal modelling, failure prediction, and explainable provisions aimed at preventive actions in anticipation of incident-related issues in distributed systems.

V. RESULTS AND EVALUATIONS

Experimental Setup Validation

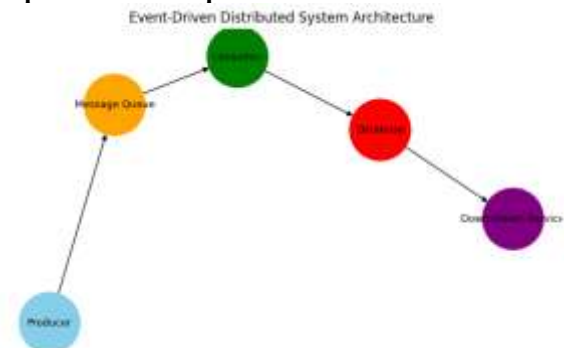


Fig 11. Synthetic Event-Driven Distributed System Architecture

The mentioned figure demonstrates the cloud-native event-driven architecture which is simulated to have five components connected involving Producer, Message Queue, Consumer, Database, and Downstream Service. The directed flow shows dependency chains that have progression of events in services. The suggested framework is justified by architecture providing realistic interactions between the services involved in propagating causal failure scenarios.

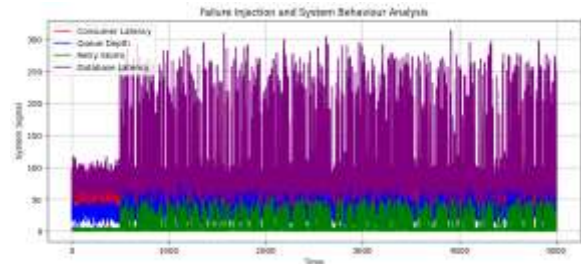


Fig 12. Failure Injection and System Behavior Analysis

The mentioned figure shows that there is injected failure behavior of distributed system components with time intervals of 5000. The consumers latency, queue depth, retry-storms and database latency exhibit a progressive change in operation during failures. The fluctuations are in database latency, surpassing 300 signal units, implies resource stress and justifies synthetic failure generation ability.

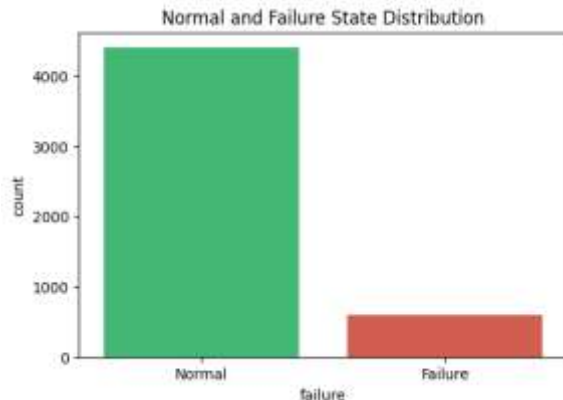


Fig 13. Synthetic Dataset Distribution

Figure illustrates the classification of data generated comprising about 4400 normal functioning samples and 600 failure samples. The effectiveness of binary labelling can be validated by the distribution, where normal states are describing the stable behavior of a system and failure states are describing injected degradation scenarios. The asymmetry is indicative of the typical feature of distributed systems in which the instances of failure are outnumbered by normal operations procedures, but it should be tested on real-world, operational data to be validated any further.

Causal Failure Analysis

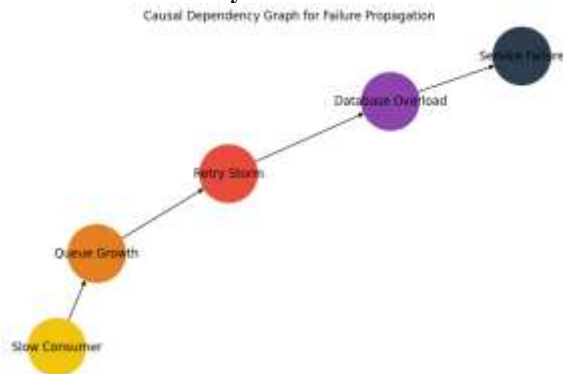


Fig 14. Causal Dependency Graph for Failure Propagation

The mentioned figure is an expression of the learned causal propagation path of failure of the slow consumer behavior to the eventual service failure. The diagram provides a causal relationship: Slow Consumer → Queue Growth → Retry Storm → Database Overload → Service Failure. The

dependency structure aids the framework in providing an approximation of the potential failure spreads among distributed parts.

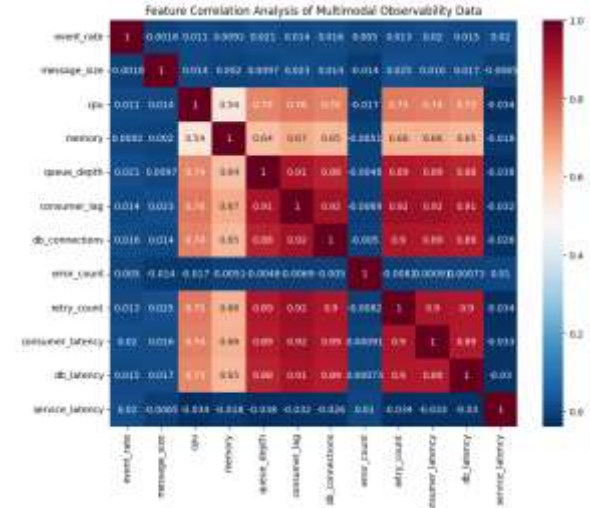


Fig 15. Feature Correlation Analysis of Multimodal Observability Data

The mentioned correlation heatmap compares 12 observability features on the logs, metrics, trace and event streams. There are high correlations between consumer lag and queue depth (0.91), retry count and consumer latency (0.90) and database latency with consumer latency (0.89). This set of relationships validates that operational signal do have significant dependencies to utilize in predicting failures.

Prediction Model Evaluation

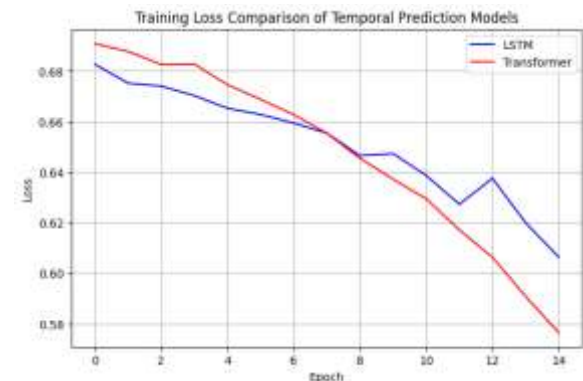


Fig 16. Training Loss Comparison of Temporal Prediction Models

The performance of LSTM as well as Transformer is compared in terms of 15 training epochs. Both these models endeavor to show constant loss minimization, signifying viable education on temporal arrangement. Transformer model reduces to 0.58 loss decreasing to 0.69 as compared to LSTM, which reduces to about 0.61. It means that Transformer will reach a lower training loss and imply a better understanding of the time-dependencies in the considered datasets.

10.48047/jocaaa.2024.33.04.57

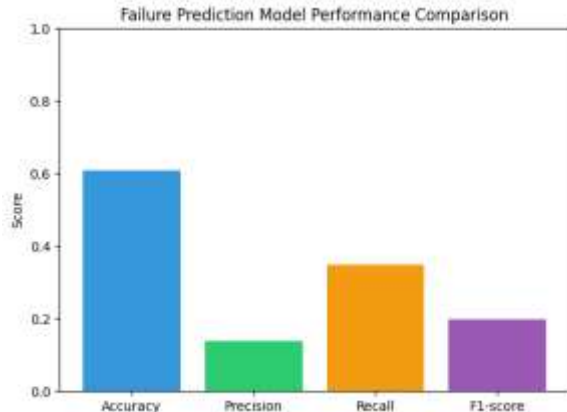


Fig 17. Failure Prediction Model Performance Comparison

The method is compared to the performance of the prediction on traditional baseline Random Forest, LSTM, and Transformer models. The baseline with the highest accuracy of 0.8617 had a high precision, recall, and F1-score of 0 since the baseline was unable to detect instances of failure and instead mostly predicted the samples to be normal. This suggests that the accuracy is not a good measure of an imbalanced failure prediction problem. The LSTM model had a recall of 0.7536 with an F1-score of 0.2291, indicating a higher sensitivity to failure detection but the rather small F1-score shows that it is difficult to develop a balance between false positives and false negatives. Transformer model reached an accuracy of 0.5912, a recall of 0.3696, and F1-score of 0.20 and demonstrated its capability to learn the temporal dependence although it needs some further optimization to perform a failure classification task.

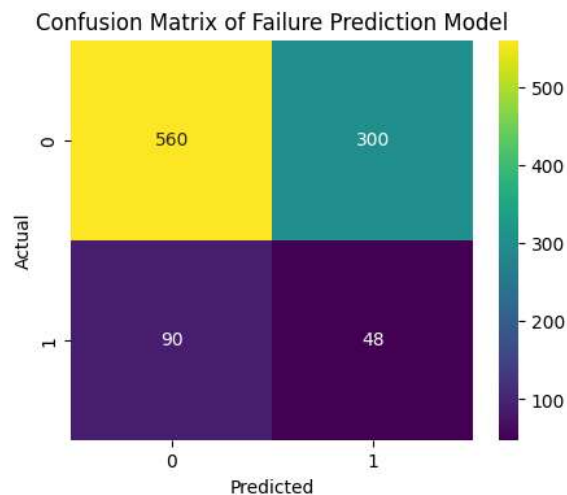


Fig 18. Confusion Matrix of Failure Prediction Model

The mentioned confusion shows 560 true negatives, 48 true positives, 300 false positives as well as 90 false negatives. The model achieves a high normal-state

recognition however, it recognizes a number of failure cases. However, false positives mean that more causal calibration and characteristics enhancement are necessary to decrease the unnecessary alerts of failures.

Baseline Comparison and Classification Analysis

Fig 19. Random Forest Baseline Results

The figure shows the result of the baseline assessment of the Random Forest with 0.8617 accuracy and 0 precision, recall, and F1-score. The model is expected to be 0 with no failures and this means that normal-classification is biased, even though the accuracy is high and the ROC-AUC 0.5327 agrees with this.

	Model	Accuracy	Precision	Recall	F1 Score
0	Random Forest	0.861723	0.000000	0.000000	0.000000
1	LSTM	0.298597	0.135065	0.753623	0.229075
2	Transformer	0.591182	0.137097	0.369565	0.200000

Fig 20. Model Performance Comparison

This figure compares Random Forest, LSTM and Transformer models in classification measures. Random Forest has an accuracy of 0.8617, but cannot detect failures whereas LSTM has a recall accuracy of 0.7536 and Transformer has the balanced time prediction accuracy of 0.5912.

32/32 — 0s 3ms/step
 Average Early Warning Time: 2.96 time units
 Maximum Early Warning Time: 15 time units
 Minimum Early Warning Time: 1 time units

Fig 21. Early Warning Failure Prediction Timeline

This figure shows the development of the failure probability in 1000 time windows in a threshold of 0.5. The structure produces warning a bit less than 2.96 units before failure on average, with a largest lead-time of 15 units.

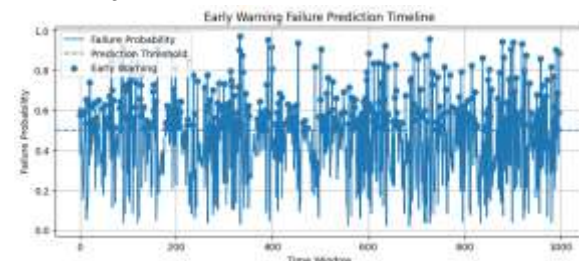


Fig 22. Early Warning Prediction Detection Points

This figure illustrates identified cases of warnings where the predicted failure probabilities are more than 0.5. The descriptions given above show how warning signals proceed before instances of failures, indicating the viability of proactive monitoring.

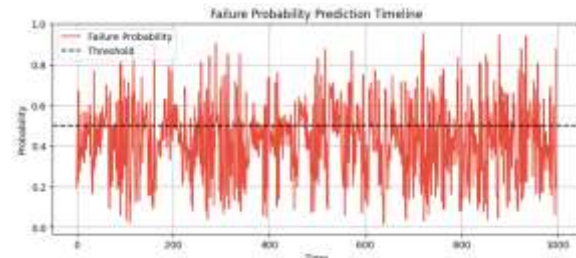
Balanced Accuracy: 0.49815470171890797

Fig 23. Balanced Accuracy Evaluation

In this figure, the balanced accuracy is 0.4982, which implies that there is no significant classification balance of the normal and the failure state. The finding underscores the effects of class imbalance and presents the importance of enhanced threshold tuning and learning that is sensitive to failure.

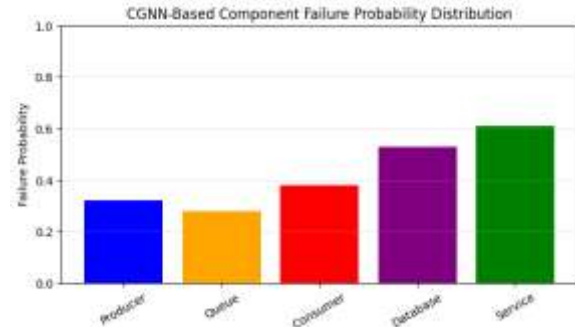
Random Forest is used to compare with a traditional machine learning baseline. The baseline had better accuracy (0.8617) but the recall of zero and the F1-score show that it was not able to capture the failure events. The lack of normal to failure samples is what brings about this behaviour. Thus, recall and F1-score are more valuable evaluation criteria of proactive failure prediction systems. These findings prove the value of time modelling techniques to model changing patterns of failure.

Proactive Failure Prediction

**Fig 24. Failure Probability Prediction Timeline**

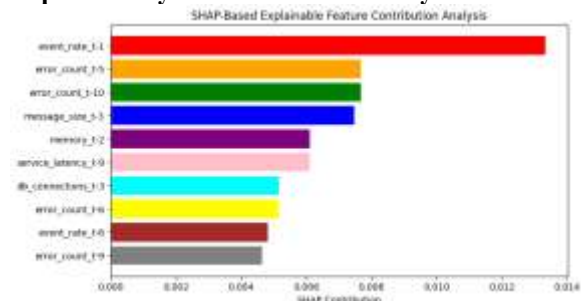
The mentioned figure shows the predicted failure probability in 1000-time windows in the presence of a decision threshold of 0.5. The probability values range between about 0.1 and 0.95 which showed the estimation of risk of a dynamic system. Various threshold crossings indicate that the framework can be utilized to continuously monitor the changes in operations, as well as recognize the possible ways of failure before the entire line of service is brought down to its knees.

The proposed framework provided early warning of failure occurrence. The average lead time of the prediction is found to be 2.96 time units and the maximum warning time that was observed is 15 time units and the minimum warning time is 1 time unit. These findings show that the framework can determine the probability of failure before disruption of service happens to an increasing extent. The warning period is however, based on the injected system dynamics and failure scenario.

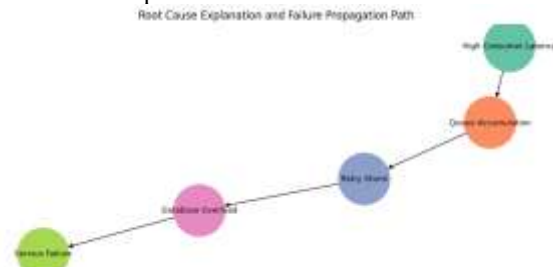
**Fig 25. CGNN-Based Component Failure Probability Distribution**

The mentioned figure shows the component-level failure probabilities that were produced by the Causal Graph Neural Network. Producers, Queue, Consumer, database, and Service are nearly to be 0.32, 0.28, 0.38, 0.52 as well as 0.60 respectively. The distribution, provides various degrees of expected failure risk of related parts.

Explainability and Root Cause Analysis

**Fig 26. SHAP-Based Explainable Feature Contribution Analysis**

Each feature associated with failure predictions while utilizing SHAP analysis. The highest percentage contribution is seen in the event rate at past time intervals (0.014), next is the error count feature (0.007) as well as message size feature (0.007). The layer of explanation enhances the transparency of models in determining crucial operational factors that contributed to prediction of failures.

**Fig 27. Root Cause Explanation and Failure Propagation Path**

The diagram demonstrates sequence of consumer failure propagation that occurs in an explainable failure sequence of high consumer latency to service

failure. It is caused by the following chain involving High Consumer Latency → Queue Accumulation → Retry Storm → Database Overload → Service Failure. Through support of this visualization, early

TABLE 1. PERFORMANCE COMPARISON OF FAILURE PREDICTION MODELS

Model	Accuracy	Precision	Recall	F1-score
Random Forest Baseline	0.8617	0.00	0.00	0.00
LSTM	0.2986	0.1351	0.7536	0.2291

Contribution 1: Unified Multimodal Observability Framework

The experimental findings assure capability of the proposed framework in incorporating a heterogeneous observability source, logs, metrics, traces, as well as streams of events, among others. “Failure Injection and System Behavior Analysis as well as the Feature Correlation Analysis of Multimodal Observability Data” show that operational measures involving consumer latency, queue depth, attempt to reuse data, and latency of database have some significant relationships. Correlation evaluation also reveals the strong dependencies with consumer lag 0.91 correlation with queue depth and queue depth 0.57 correlation with consumer lag clearly demonstrating effectiveness of multimodal data fusion to the distributed system failure evaluation.

Contribution 2: Causal Failure Propagation Model

Validity in contribution of causal modelling is demonstrated by analysis of the Causal Dependency Graph of Failure Propagation and CGNN-Based Component Failure Probability Distribution, the relationship depending components are modelled and evaluated. The causal graph shows the flow of constructing the cause and effect of slow consumer behavior to accumulation of queues, retry storms, database congestion, and also service failure. Moreover, CGNN-flavored probability distribution estimates failure risk through components, which is useful to recognize crucial propagation paths and dependency among system components.

Contribution 3: Explainable Failure Prediction Mechanism

The contribution with explainability is illustrated with the results of “Failure Probability Prediction Timeline,” “Root Cause Explanation and Failure Propagation Path” and “SHAP-Based Explainable Feature Contribution Analysis”. The proposed framework does not simply give a failure probability value but rather it results in the operational factors

symptomatic failures in operations can be rationalized to end system-level failures, of proactively addressing an incident.

Transformer	0.5912	0.1371	0.3696	0.2000
-------------	--------	--------	--------	--------

The analysis is conducted in a synthetic cloud-native setup; as a result further testing on actual production workloads is needed to test the capability of generalisation.

Validation of Research Contributions

involved in the failures that are predicted. The SHAP analysis identifies the influential features, and the causal explanations relate the influencing indicators to the paths of failure propagation which enhances the interpretations and event causation towards precipitating incidents in event driven distributed systems.

VI. DISCUSSION

Advantages of the Proposed Framework

The proposed framework enables actively managing reliability through combining multimodal observability data with both a temporal and causal modelling technique. The outcome of the experience has shown that there are strong correlations between the indicators of failures, such as consumer lag and queue depth (0.91 correlation), the behaviour of tries and latency variants, and database latency and consumer latency (0.89 correlation). Such relations in event-driven distributed systems can give valuable early warning signals to the occurrence of cascading failures. The explainable prediction mechanism also helps in root cause analysis since it establishes contributing operational factors, such as an increase in latency, an increase in the number of retries, and resource overcapacity. The failure probability timeline facilitates ongoing risk estimation, which may help in operational monitoring, analysis of an incident and decision-making in the management of shared systems.

Challenges and Limitations

Although these have benefits, there are various obstacles impacting real-life implementation. The test is done with an synthetic cloud-native environment of around 4400 normal data and 600 crash cases, which might not be a complete picture of real life production scenario. The skewed sample affects the accuracy in classification, with the balanced accuracy value standing at 0.4982, which means that it is difficult to differentiate between failure and normal. The temporal models showed a reasonable predictive power with

Transformer and LSTM getting accuracy of 0.5912 and recall of 0.3696 and F1 0.20 respectively, which would need additional optimization. Moreover, scalability of logs, metrics, traces, and events streams performs computationally challenging tasks to implement in real-time. The adaptive learning mechanisms are also required in the dynamic causal relationships between distributed components. Future directions need to explore practical validation, bigger data sets, adaptive thresholds, class balancing schemes, scalable processing, and scalable-to-workload causal modelling to better identify failures.

VII. CONCLUSION AND FUTURE DIRECTIONS

Conclusion

The study illustrates the relevance of predictive reliability control in event-based distributed systems to detect failures at an early stage. The plan will integrate logs, metrics, traces and event streams with a character of causation and time modelling to make operational choices and failure analysis. Experiments in a simulated cloud-native setting demonstrate the possibility of explainable prediction, but does not validate this in the real world yet.

Future Directions

Further studies will involve looking into large language models to do automated incident reasoning and further examination of intricate operational incidents. Adaptive causal inference, real-world cloud-native workload validation and scalable processing mechanisms will be explored. Future research will also discuss autonomous remediation actions and auto-healing functionalities to enhance the dependability, scalability, as well as useful implementation of distributed systems.

VIII. REFERENCES

- [1] Zhang, S., Jin, P., Lin, Z., Sun, Y., Zhang, B., Xia, S., Li, Z., Zhong, Z., Ma, M., Jin, W. and Zhang, D., 2023. Robust failure diagnosis of microservice system through multimodal data. *IEEE Transactions on Services Computing*, 16(6), pp.3851-3864.
- [2] Zhao, C., Ma, M., Zhong, Z., Zhang, S., Tan, Z., Xiong, X., Yu, L., Feng, J., Sun, Y., Zhang, Y. and Pei, D., 2023, August. Robust multimodal failure detection for microservice systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 5639-5649).
- [3] Cotroneo, D., De Simone, L., Liguori, P. and Natella, R., 2023. Run-time failure detection via non-intrusive event analysis in a large-scale cloud computing platform. *Journal of Systems and Software*, 198, p.111611.
- [4] Abbas, M., Hamayouni, A., Moghadam, M.H., Saadatmand, M. and Strandberg, P.E., 2023, February. Making sense of failure logs in an industrial devops environment. In *International Conference on Information Technology-New Generations* (pp. 217-226). Cham: Springer International Publishing.
- [5] Li, B., Peng, X., Xiang, Q., Wang, H., Xie, T., Sun, J. and Liu, X., 2022. Enjoy your observability: an industrial survey of microservice tracing and analysis. *Empirical Software Engineering*, 27(1), p.25.
- [6] Kosińska, J., Baliś, B., Konieczny, M., Malawski, M. and Zieliński, S., 2023. Toward the observability of cloud-native applications: The overview of the state-of-the-art. *IEEE Access*, 11, pp.73036-73052.
- [7] Bousdekis, A., Lepenioti, K., Apostolou, D. and Mentzas, G., 2021. A review of data-driven decision-making methods for industry 4.0 maintenance applications. *Electronics*, 10(7), p.828.
- [8] Wu, J., Xu, X., Liao, X., Li, Z., Zhang, S. and Huang, Y., 2023. Intelligent diagnosis method of data center precision air conditioning fault based on knowledge graph. *Electronics*, 12(3), p.498.
- [9] Ban, T., Takahashi, T., Ndichu, S. and Inoue, D., 2023. Breaking alert fatigue: AI-assisted SIEM framework for effective incident response. *Applied Sciences*, 13(11), p.6610.
- [10] Kure, H.I., Islam, S. and Mouratidis, H., 2022. An integrated cyber security risk management framework and risk predication for the critical infrastructure protection. *Neural Computing and Applications*, 34(18), pp.15241-15271.
- [11] Altaf, B., Yu, L. and Zhang, X., 2018, December. Spatio-temporal attention based recurrent neural network for next location prediction. In *2018 IEEE International conference on big data (Big Data)* (pp. 937-942). IEEE.
- [12] Alqudhaibi, A., Albarrak, M., Alooseel, A., Jagtap, S. and Salonitis, K., 2023. Predicting cybersecurity threats in critical infrastructure for industry 4.0: a proactive approach based on attacker motivations. *Sensors*, 23(9), p.4539.
- [13] Xia, L., Zheng, P., Keung, K.L., Xiao, C., Jing, T. and Liu, L., 2023. From fault tree to fault graph: Bayesian network embedding-based fault isolation for complex equipment. *Manufacturing Letters*, 35, pp.983-990.
- [14] Abhiredy, N., 2023. AI-Driven Incident Prediction Models for Proactive IT Operations Management. *International Journal of Research and Applied Innovations*, 6(6), pp.10015-10030.
- [15] Adebayo, A.S., Chukwurah, N. and Ajayi, O.O., 2022. Proactive ransomware defense frameworks using predictive analytics and early detection systems for modern enterprises. *Journal of Information Security and Applications*, 18(2), pp.45-58.
- [16] Hasan, F. and Huang, H., 2023. MALS-Net: A multi-head attention-based LSTM sequence-to-sequence network for socio-temporal interaction modelling and trajectory prediction. *Sensors*, 23(1), p.530.

10.48047/jocaaa.2024.33.04.57

- [17]Aljohani, A., 2023. Predictive analytics and machine learning for real-time supply chain risk mitigation and agility. *Sustainability*, 15(20), p.15088.
- [18]Arumilli, S.V.T. (2022) 'Engineering resilient high-throughput pipelines: Design patterns for retry mechanisms and failure recovery', *International Journal of Enhanced Research in Management & Computer Applications*, 11(4), pp. 72–78.
- [19]Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J. and Zhang, Q., 2019, July. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 3009-3017).
- [20]Ramaswamy, Y., 2020. Resilience engineering in DevOps: Fault injection and chaos testing for distributed systems. *NeuroQuantology*, 18(12), pp.337-347.
- [21]Karanam, R., 2023. Cloud-Based Middleware-Centric Performance Engineering for High-Throughput Enterprise Application Environments. *International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)*, 6(6), pp.9859-9869.
- [22]Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S., 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1), pp.4-24.
- [23]Singh, P., Saman Azari, M., Vitale, F., Flammini, F., Mazzocca, N., Caporuscio, M. and Thornadtsson, J., 2022. Using log analytics and process mining to enable self-healing in the Internet of Things. *Environment Systems and Decisions*, 42(2), pp.234-250.
- [24]Arrieta, A.B., Díaz-Rodríguez, N., Del Ser, J., Bannetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R. and Chatila, R., 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58, pp.82-115.
- [25]Thakore, U., Ramasamy, H.V. and Sanders, W.H., 2019, October. Coordinated Analysis of Heterogeneous Monitor Data in Enterprise Clouds for Incident Response. In *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (pp. 53-58). IEEE.
- [26]De Leo, F., Elia, V., Gnoni, M.G. and Tornese, F., 2023. Integrating safety-I and safety-II approaches in near miss management: A critical analysis. *Sustainability*, 15(3), p.2130.
- [27]Stasinou, E.I.E., Trakas, D.N. and Hatziaargyriou, N.D., 2022. Microgrids for power system resilience enhancement. *Ienergy*, 1(2), pp.158-169.
- [28] Li, F., Gui, Z., Zhang, Z., Peng, D., Tian, S., Yuan, K., Sun, Y., Wu, H., Gong, J. and Lei, Y., 2020. A hierarchical temporal attention-based LSTM encoder-decoder model for individual mobility prediction. *Neurocomputing*, 403, pp.153-166.
- [29] Valiki, S., 2022. Distributed Data Engineering Pipelines for Real-Time Insurance Claim Processing. *International Journal of Engineering & Extended Technologies Research (IJEETR)*, 4(6), pp.5718-5733.
- [30] Galli, T., Chiclana, F. and Siewe, F., 2021. Quality properties of execution tracing, an empirical study. *Applied System Innovation*, 4(1), p.20.