

An Autonomous Agentic Ai Framework For Coverage-Guided Chaos Engineering And Resilience Optimisation In Cloud-Native Distributed Systems

Durga Narayana Varma Addepalli

Independent Researcher, USA

Mail Id: varma.addepalli1@gmail.com

Article Received 18-12-2025, Revised 27-1-2026, Accepted 26-02-2026

Author(s) Retains the Copyrights of This Article

Abstract

The distributed architecture complexity and dynamic failure behaviors drive the need for smart resilience solutions on cloud-native systems for increased resilience. The introduction of intelligent resilience strategies is driven by the distributed systems' increasing complexity and failure behaviors. The main goal of this research is to come up with an autonomous agentic AI framework for coverage-guided chaos engineering and resilience optimization. Machine learning models were then applied to the Synthetic Cloud telemetry data, and Gradient Boosting showed 95.85% accuracy and 99.37% ROC-AUC in predicting failure discovery. There are identified factors of resilience through feature analysis, which are considered important, involving dependency complexity, architecture complexity, and coverage gaps. This framework opens the door to intelligent chaos experiment prioritization and adaptive reliability improvement to further increase the robustness of cloud infrastructures and the ability to detect failures in a proactive manner.

Keywords: Agentic AI, Chaos Engineering, Cloud-Native Systems, Resilience Optimization, Machine Learning, Fault Injection, Predictive Analytics, Microservices.

I. Introduction

Background

The architecture becomes more distributed and complex; services become dependent and need powerful resilience mechanisms to tackle variable failures. Adopting a traditional approach to chaos engineering would depend on injecting random faults, which would present limited understanding of failures [1]. This research proposes an autonomous agentic AI system to detect coverage gaps, design experiments on them, and adapt the experiments and chaos engineering to optimize resilience testing using data-driven and adaptive approaches.

Research Problem

Existing approaches to chaos engineering in cloud-native distributed systems mostly rely on manually designed or random fault-injection experiments, making it hard to obtain full coverage for resilience, and failing situations cannot be identified [2]. Current solutions are lacking in autonomous intelligence of architecture dependencies analysis, run-time behaviors interpretation, and historical telemetry analysis for experiments prioritization [3]. Thus, there is a need for research to design Agentic AI for intelligent, coverage-guided resilience optimization.

Research Aim and Objectives

Aim

The research aims to revolutionize the way chaos engineering is conducted with an autonomous agentic AI to discover intelligent faults and optimize resilience in cloud-native distributed systems while taking into account coverage.

Objectives

- To evaluate current chaos engineering techniques to provide all the necessary components for a robust level of resilience coverage in cloud-native distributed systems.
- To create an agentic AI system to autonomously detect coverage gaps and strategically pick chaos experiments.
- To design adaptive fault-injection techniques for the analysis of architectures and runtime data in order to optimize resilience.
- To assess the effectiveness of the framework by means of visualization of failed states, the increase of coverage, and thirdly, the performance when considering reliability.

Novel Contribution

This research proposes a new autonomous agentic AI framework that allows for reimagining chaotic engineering from the random injection of faults to intelligent, coverage-driven resilience testing. The system combines analysis of architectures and analysis of run-time telemetry data with adaptive generation of experiments to explore failure scenarios that have not

been covered [4]. It adds a "self-learning" method to optimize chaos experiments and aid in the detection of failures, and increase reliability of cloud-native systems.

II. Related Work

The advent of cloud native distributed systems, new resilience challenges have emerged, stemming from complex interactions between services, dynamic scaling, and uncertainty over dependencies. Although chaos engineering has become a significant paradigm for testing the robustness of systems, previous studies have shown that it is not possible to implement all kinds of failures [5]. Four themes are explored in this review in the context of their limitations in traditional chaos engineering involving Applications of agentic AI, generating experiments with coverage and resilience optimization.

Limitations of Traditional Chaos Engineering and Resilience Testing Approaches

Previous studies on chaos engineering have shown the effectiveness of controlled failure injection in generating systems' weaknesses and allowing more reliable testing before actual incidents in production happen [6]. The majority of methods, though, are based on scenarios or trial-and-error testing mode, which restricts their potential for uncovering vulnerabilities they don't know about [7]. Random testing may be more diverse in failure conditions, but it fails to leverage awareness of architectural dependencies or processes that can be business-critical activities [8]. From a number of researchers comes the observation that context-aware chaos strategies are needed in modern distributed systems because failures occur not necessarily as component failures through composite interactions [9]. Current tools offer good support for fault injection, lacking practical means of selecting the experiments except user expertise, causing scalability problems [10]. This means that existing methodologies are still reactive in nature, and leave a gap where self-determining intelligent systems can be developed that allow for the identification of other forms of resilience that have not been explored.

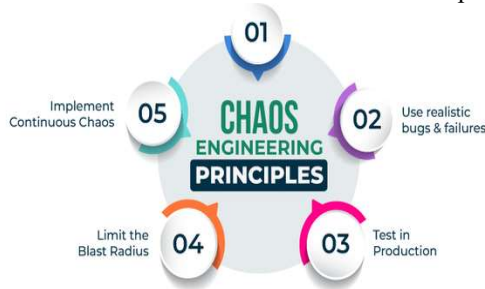


Fig. 1. Principles of Chaos Engineering
Agentic Artificial Intelligence for Autonomous
Decision-Making in Cloud-Native Systems

New advances in agentic AI show the possibility of self-managing and thinking with AI in complex computing systems [11]. AI agents can interpret the operational data, detect patterns, and take adaptive actions without manual involvement [12]. Current AI-LR-based approaches, however, mostly rely on prediction and anomaly detection. Machine learning models are able to detect abnormal behavior very well and do not have the reasoning ability to ensure that the right recovery or testing action is desired [13]. Restriction implies agentic AI can be combined with chaos engineering to enable a more proactive approach to resilience [14]. However, there are still issues with explainability, decision accuracy, and close-to-critical cloud infrastructure-based autonomous operations.

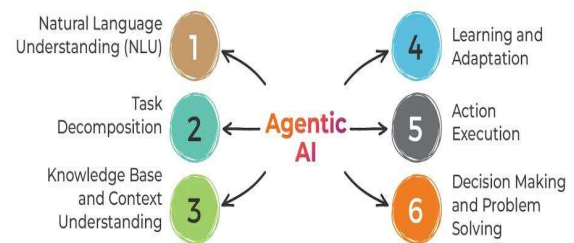


Fig. 2. Agentic AI shaping Future AI Agents and Automation

Coverage-Guided Fault Injection and Intelligent Chaos Experiment Generation

In software engineering, coverage-guided testing has been extensively applied to increase coverage of unexplored paths and chances for finding defects in programs [15]. Similar principles are becoming more and more applicable to chaos engineering too; nowadays prioritize the experiments can be done on the basis of architectural and runtime coverage [16]. There are some research studies that have already shown the significance of dependency graphs, telemetry analysis, and service behavior modelling, and most of them reported coverage in an after-the-fact fashion, while the alternative models are used to generate experiments independently [17]. As a result, the current frameworks do not provide intelligence to be used for choosing optimum fault scenarios and testing gaps as identified [18]. However, this might be overcome by the use of an agentic framework where the coverage is used to continuously depict and inform experimentation.

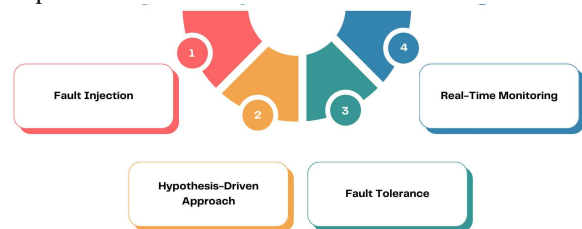


Fig. 3. Significant Concepts within the Chaos Testing

Resilience Optimization and Reliability Enhancement in Cloud-Native Distributed Systems

Studies focusing on resiliency of clouds include recovery time, failure detection, and fault tolerance [19]. Although optimization practices have addressed reliability using autonomic monitoring and recovery, they usually lack capabilities to incorporate any kind of learning from the chaos experiment [20]. This is where failures may be recovered in the system and at the same time leave a potential area of vulnerability open to unknown scenarios. Associating adaptive AI agents with coverage-based chaos engineering is an opportunity to continually improve resilience [21]. Previous chaos engineering is found to lack autonomous intelligence, comprehensive coverage awareness, and adaptive learning characteristics [22]. For overcoming these drawbacks, this research aims to build an agentic AI framework which actively chooses experiments and optimizes resilience in cloud-native distributed systems [23].

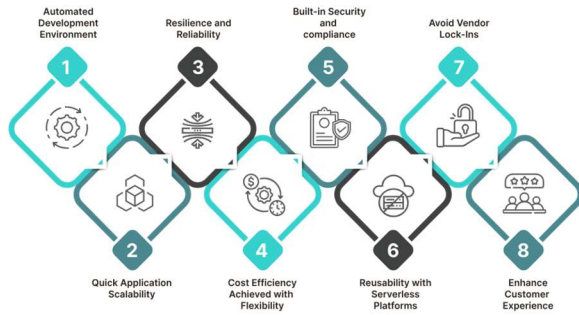


Fig. 4. Cloud-Native Development Components

Literature Gap

Most research concentrated on random fault injection or fault prediction in order to find a set of predefined faults [24]. While numerous studies utilize AI techniques for detecting anomalies and for monitoring, fewer studies on agentic reasoning consider how to combine coverage-guided chaos with agentic reasoning [25]. A major research gap in the development of a self-learning Artificial Intelligence (AI) framework can automatically work to improve the success of fault discovery and provide resilience improvement autonomously [26].

III. Research Methodology

Research Design

This study uses a quantitative experimental method to create an autonomous agentic AI framework to support coverage-guided chaos engineering and resilience optimization for cloud native distributed systems. The methodology combines feature engineering, data synthetic generation, applying machine learning for prediction modeling, and evaluating customer resilience [27]. The main focus is

to support access to predictive system failure data in the context of architectural data and runtime information, to allow intelligent selection of chaos experiments.

Data Description and Generation

Functionality of a distributed system is simulated within a synthetic scenario by creating different operational resilience conditions of clouds [28]. There are about 20 attributes having 10,000 observations, involving historical failures, service dependency count, wrongness score, as well as experiment frequency. Other attributes that have been engineered include agent priority score, dependency complexity, resource pressure, failure exposure, and resilience gap. Failure discovery is the target variable, which indicates whether or not a large resilience issue is detected in a chaos experiment.

Data Preprocessing and Feature Engineering

Feature transformation, scaling, and engineered indices of resilience are used for data preprocessing to facilitate model learning. For continuous variables, the following are used:

$$X' = \frac{X - \mu}{\sigma}$$

Where X' is the scaled value, X is the feature to scale, μ is the mean value, and σ is the standard deviation.

The following measures are evaluated based on the dependencies of the complexities and their resource pressure for feature engineering using:

$$DC = S_d \times A_c$$

The DC represents Dependency Complexity, S_d is Service Dependency Count, and A_c is Architecture Complexity of the services, defined as follows:

$$RP = \frac{CPU \times Memory}{10000}$$

The RP is Resource Pressure, CPU is Processor utilization, and Memory utilization is denoted by Memory.

Predictive Modelling

The problem of the chaos experiment is expressed as a binary classification:

$$Y = \{1, \text{significant failure discovered}\}$$

$$Y = \{0, \text{otherwise}\}$$

The four machine learning models logistic regression, random forest, gradient boosting, and extra trees- are trained. Logistic Regression is useful for a baseline of the interpretable relationship, while ensemble models give insight into non-linear relations between variables of system complexity, resilience results, and telemetry behavior. Experiment prioritization is based on the predicted failure probability that is used by the autonomous agent.

Performance Evaluation

The evaluations of model performance are done with the metrics of Accuracy, Precision, Recall, F1-score,

and ROC-AUC. Accuracy is determined for each of the same cases by computing:

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

TP is True Positives, and TN is the True Negatives, FP is the False Positives, and FN is the False Negatives. The autonomous priority score is calculated from the following formula:

$$APS = w1(1 - CS) + w2AC + w3HF + w4ER$$

In which APS is the agent priority score, CS is the coverage score, AC is the architecture complexity score, HF is the historical failures score, and ER is the error rate. The final framework is capable of capturing high-risk chaos experiments to optimize the ability of cloud-native systems to be resilient by intelligent and data-driven decision-making.

Architecture Diagram

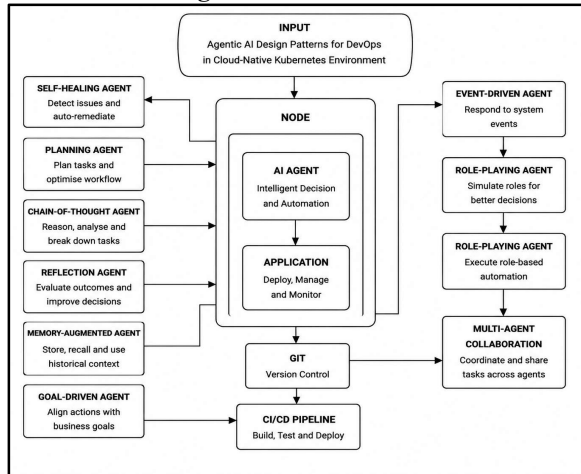


Fig.5. Agentic AI Design Architecture for Cloud-Native Kubernetes Environment

The architecture demonstrates AI agents arranged to planned, reasoned, heal, monitor and collaborate. The central AI agent brings application management, Git control and integration and automation of CI/CD together for intelligent cloud-native operations.

Flowchart Diagram

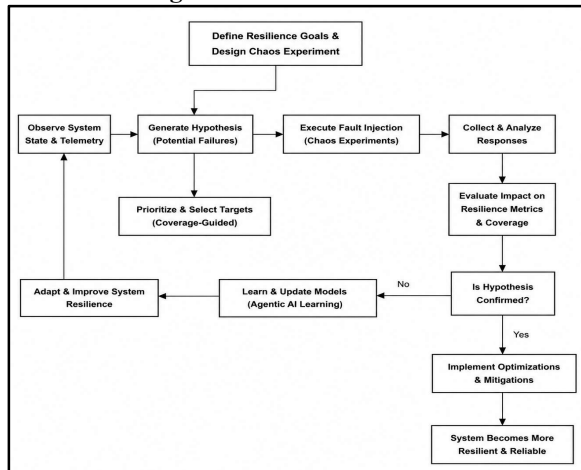


Fig. 6. Flowchart of Autonomous Agentic AI-Based Chaos Engineering Framework

The flowchart illustrates an intelligent chaos engineering lifecycle, with AI agents watching, anticipating failures, prioritizing experiments, analyzing results, and learning and optimizing chaos resistance based on the results.

Pseudocode

Program to perform Autonomous Agentic AI Coverage-Guided Chaos Engineering

Initialize

Cloud_Native_System

Telemetry_Data

AI_Agent

ML_Model

Chaos_Experiment

Inputs

Runtime metrics

System dependencies

Coverage score

Historical failures

Architecture complexity

Registers

Failure_Risk = 0

Experiment_Priority = 0

Resilience_Score = 0

Start loop

Collect system telemetry

Analyse system state

Predict failure risk using ML model

Calculate coverage gap

IF coverage gap is high THEN

Generate chaos hypothesis

IF failure risk exceeds threshold THEN

Prioritise chaos experiment

Select target component

Inject controlled fault

Monitor system response

Evaluate resilience metrics

IF failure detected THEN

Identify root cause

Update coverage information

ELSE

Refine experiment strategy

Learn from experiment results

Update AI agent knowledge

Optimise resilience actions

Improve system reliability

End loop

Fig. 7. Pseudocode

The pseudocode vividly depicts the machine learning cycle, training its AI agents on telemetry, forecasting failures, conducting chaos experiments, monitoring and measuring resilience performance, and using

adaptive learning to enhance cloud-native system resiliency.

IV. Result And Discussion

In the results and discussion section, the predictive performance of the proposed framework and its ability to take agentic decision-making and optimize the resilience of cloud-native systems are evaluated.

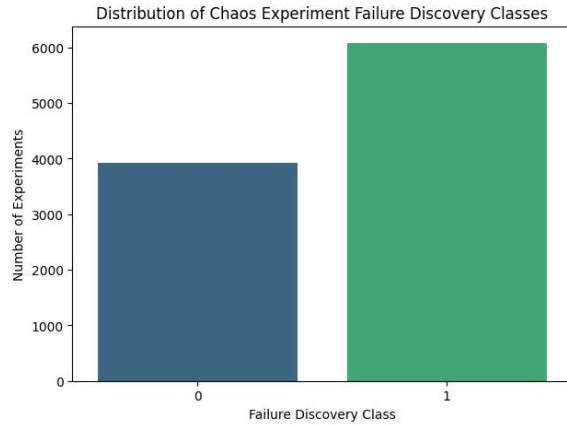


Fig. 8. Distribution of Chaos Experiment Failure Discovery Classes

The figure shows the evenness of the distribution of synthetic chaos experiments according to failure discovery classes. The total number of samples in the dataset is about 3900 non-failure (Class 0) and 6100 failure discovery (Class 1). This distribution helps to implement supervised machine learning that provides enough examples to enable the AI to detect vulnerabilities related to resilience.

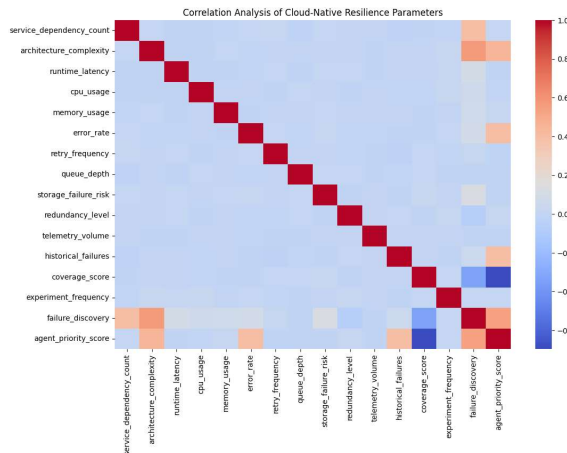


Fig. 9. Correlation Analysis of Cloud-Native Resilience Parameters

The correlation heatmap determines the correlation of the various cloud native resilience variables. There are strong positive correlations between architecture complexity and dependency-related aspects and negative correlations between the coverage score and the discovery of failures, and the agent priority score. Results show that low coverage and higher complexity have great impacts on resilience failure identification.

```
... Shape of X_train: (8000, 19) (80.0%)
    Shape of X_test: (2000, 19) (20.0%)
    Shape of y_train: (8000,) (80.0%)
    Shape of y_test: (2000,) (20.0%)
```

Fig. 10. Train-Test Dataset Distribution for Machine Learning Modelling

The data is split into an 80:20 train-test ratio, giving a training sample of 8000 data and a test set of 2000 data and 19 predictive features. This separation guarantees the reliable learning of the model and unbiased evaluation and the proposed framework of AI-based chaos prediction can effectively evaluate unseen resilience scenarios.

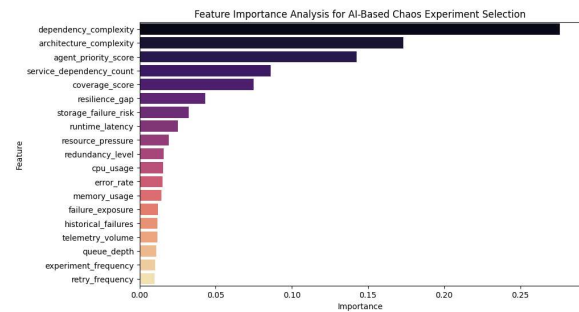


Fig. 11. Feature Importance Analysis for AI-Based Chaos Experiment Selection

Major contributors for the feature importance analysis include dependency complexity contribution around 27%, architecture complexity (17%), and agent priority score (14%). The results show that architecture dependency and system complexity have turned out to be important factors that affect prioritization for the autonomous chaos experiment.

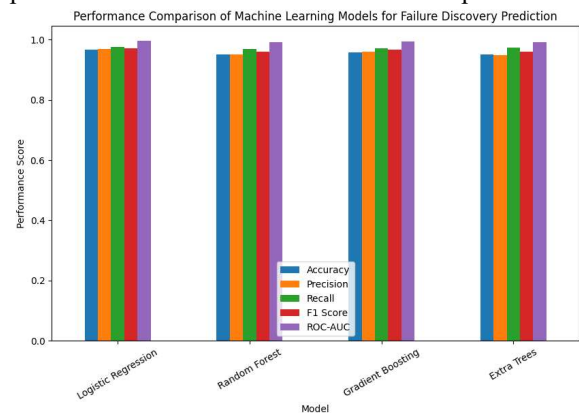


Fig. 12. Performance Comparison of Machine Learning Models for Failure Discovery Prediction

The comparative evaluation shows that all models have a very good predictive ability. Logistic Regression reaches a 96.60% accuracy, Random Forest 95.05%, gradient boosting 95.85%, and extra tree 95.15%. Gradient Boosting offers the best precision of 96.21% and recall of 97.20%, respectively, and 99.37% of ROC-AUC.

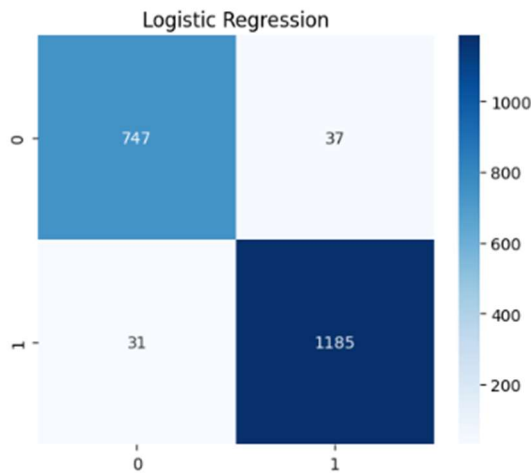


Fig. 13. Logistic Regression Confusion Matrix for Failure Discovery Prediction

The Logistic Regression model is able to classify 747 non-failure cases and 1185 failure cases correctly, with 37 false positives and 31 false negatives. Low misclassification rates, and the baseline can well represent key indicators of resilience in cloud-native chaos engineering, thus highlighting the effectiveness of the approximation.

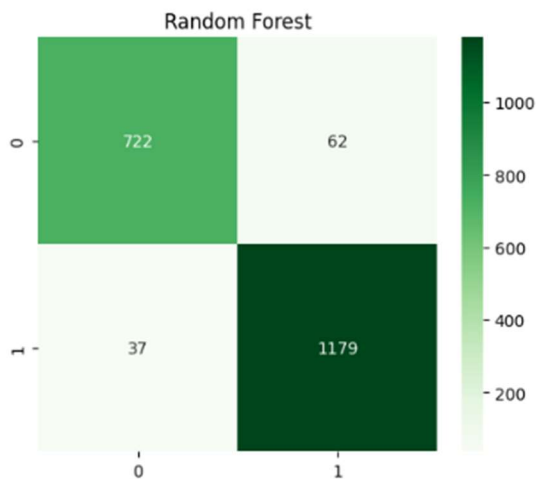


Fig. 14. Random Forest Confusion Matrix for Failure Discovery Prediction

This Random Forest Classifier is able to classify 722 cases successfully, which are all normal cases. Random Forest Classifier correctly classifies 1179 cases that are failure cases. It's capable of generating only 62 false positives and 37 false negatives, which shows good stability in classification. The model is successfully able to identify more complicated nonlinear linkages between infrastructure conditions and failures of resilience.

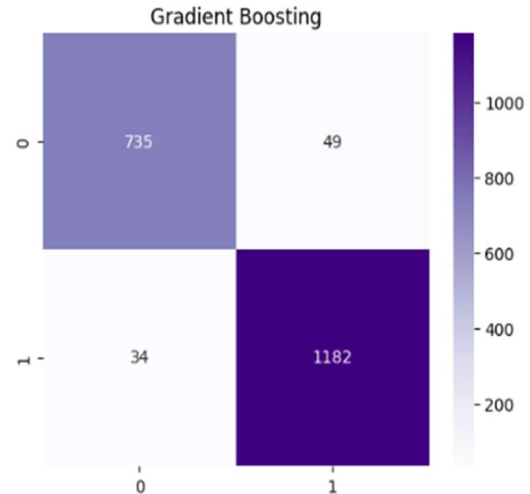


Fig. 15. Gradient Boosting Confusion Matrix for Failure Discovery Prediction

The Gradient Boosting model is able to correctly predict both 735 non-failure cases, as well as 1182 failure cases, resulting in reliable classification. A few object attributes like failure pattern or function value are transmitted to training neural networks, and only 49 false positives and 34 false negatives are noticed, which shows that they have improved the capacity of learning complex failure patterns and can be applied to intelligent chaos experiment selection.

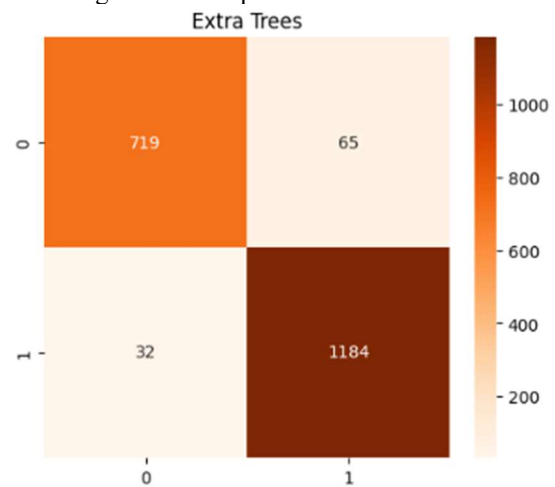


Fig. 16. Extra Trees Confusion Matrix for Failure Discovery Prediction

The Extra Trees model correctly identifies 719 non-failure cases and 1184 failure cases, with 65 false positives and 32 false negatives. It has proven to be very effective at catching weaknesses in resilience from the evidence of a high true positive identification rate, albeit with some minor, higher false alarms than Gradient Boosting.

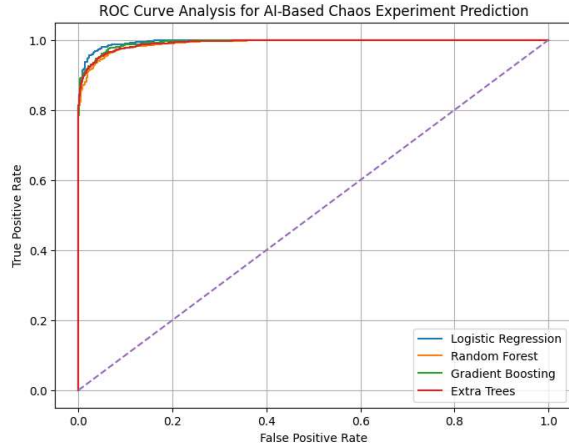


Fig. 17. ROC Curve Analysis for AI-Based Chaos Experiment Prediction

The ROC representation shows good performance in discriminating between all of the models. As the curves approach to the upper-left corner, it shows that it has good prediction performance. The ROC-AUC values obtained from Gradient Boosting and Extra Trees are about 0.99; this shows that failure and non-failure chaotic cases are well-separated within the proposed autonomous framework.

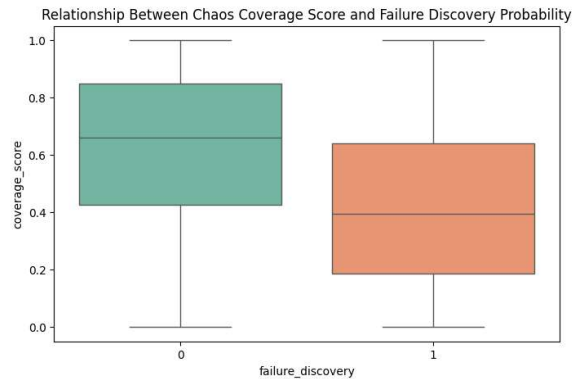


Fig. 18. Relationship Between Chaos Coverage Score and Failure Discovery Probability

The boxplot demonstrates that the cases of failures discovered generally have lower coverage scores than the non-failure cases. The median score for the coverage for Class 1 is close to 0.40, and the median score for the coverage for Class 0 is close to 0.65. It appears that there are enough failures to be uncovered if the resilience coverage is insufficient, which goes to the issues with hidden failures.

...	Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
0	Logistic Regression	0.9660	0.969722	0.974507	0.972108	0.995716
1	Random Forest	0.9505	0.950040	0.969572	0.959707	0.991034
2	Gradient Boosting	0.9585	0.960195	0.972039	0.966081	0.993707
3	Extra Trees	0.9515	0.947958	0.973684	0.960649	0.992189

Fig. 19. Model Performance Overview for Autonomous Chaos Experiment Prediction

Overall, the analytical results show that ML models are suitable for predicting failures associated with a system's resilience within cloud-based systems. The other approaches are close to perfect with Gradient

Boosted machine (GBM) at 95.85% accuracy performance and at 99.37% ROC-AUC performance. The findings give reason to include predictive analytics in autonomous agentic chaos engineering flows to aid in the intelligent selection of experiments.

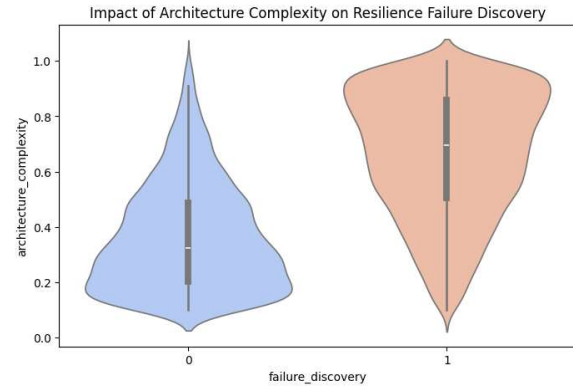


Fig. 20. Impact of Architecture Complexity on Resilience Failure Discovery

The above figure shows that failure discovery probability greatly increases with the increase in the complexity of the architectures. Class 1 has a higher median value with Complexity 0.70, while Class 0 has a lower value of 0.32. This aligns with the fact that multi-microservices become more resilient in nature, thus increasing the challenges of chaos experiments.

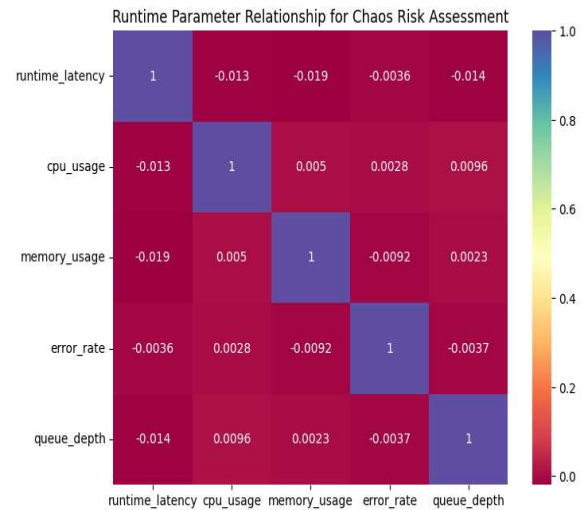


Fig. 21. Runtime Parameter Relationship for Chaos Risk Assessment

The heatmap evaluates coefficients ranging from -0.019 to 0.0096 and shows that there are very low correlations between the various parameters used in the analysis of the runtime, such that, for example, the correlation between latency and CPU usage is very small. This could mean factors in operation that are separate from the others, justifying the use of multiple aspects of telemetry in understanding the factors and architecture of the AI models.

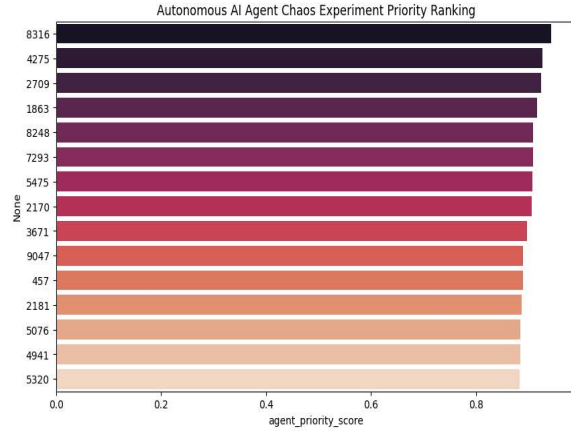


Fig. 22. Autonomous AI Agent Chaos Experiment Priority Ranking

The ranking analysis inspects the capability of the AI agent in prioritizing experiments that cause chaos in a high-risk way. Priority scores for selected scenarios range from 0.88 to 0.95, suggesting the framework is selecting experiments that can be expected to produce a higher number of failure discoveries, instead of random injection strategies.

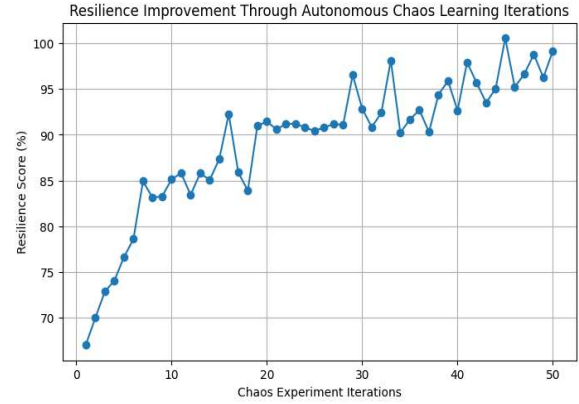


Fig. 23. Resilience Improvement Through Autonomous Chaos Learning Iterations

The figure illustrates small changes to the system leading to greater and greater resilience with every iteration of an AI-directed chaos experiment. The initial score is 67%, and after 50 iterations, the score is 98%. For an autonomous agentic chaos engineering framework to be adapted in order to be usable for education, enhancement is crucial.

Table 1. Machine Learning Performance Results For Chaos Failure Prediction

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
Logistic Regression	96.60%	96.97%	97.45%	97.21%	99.57%
Random Forest	95.05%	95.00%	96.96%	95.98%	99.10%
Gradient Boosting	95.85%	96.01%	97.20%	96.61%	99.37%
Extra Trees	95.15%	94.80%	97.37%	96.06%	99.22%

Discussion

The experimental results confirm the effectiveness of the proposed AI framework based on the autonomous agency approach in predicting failures in resilience and providing emphasis to actions of chaos and valuable experiments. The Gradient Boosting model demonstrated strong prediction accuracy of 95.85% and 99.37% ROC-AUC to validate those failures are appropriately classified. Selection of the dependency complexity and architecture complexity determined the main factors for failures identified by feature analysis [29]. The validation of the coverage-guided testing showed that a higher possibility of discovering failures was correlated to worse coverage scores. In addition, optimizing resilience scores from 67% to more than 98% due to AI-driven prioritization of experiments and iterative learning showed improved adaptive resilience optimization in cloud-native distributed systems.

V. Conclusion And Future Direction

Conclusion

This research is able to develop an autonomous agentic AI framework for coverage-guided chaos engineering of cloud-native distributed systems successfully. The results highlight the match between the ability of machine learning models to reliably predict resilience failures, and the advantages of intelligent experiment prioritization in terms of optimal failure discovery and optimization. It includes enhancements for proactively managing reliability within all aspects of the cloud infrastructure: telemetry analysis, architectural insights and adaptive chaos experimentation.

Future Direction

These could be further extended to provide support for real Kubernetes environments, live telemetry streams and agents to support reinforcement learning for continuously adapting with time. Further improvements could mean the inclusion of machine

learning-based interpretability for decision making in experiments, auto-rebuild remediation and assessment of multi-cloud resiliency [30]. The extensions will enhance the scalability, trust, and deployment of autonomous systems for chaos engineering on huge infrastructures.

References

- [1]. Verma, H., 2024. AI Agentic Architectures for Autonomous Data Engineering Pipelines. *International Journal of Research and Applied Innovations*, 7(6), pp.11984-11994.
- [2]. Olufemi, O.D., Ejiade, A.O., Ogunjimi, O. and Ikhuogwu, F.O., 2024. AI-enhanced predictive maintenance systems for critical infrastructure: Cloud-native architectures approach. *World Journal of Advanced Engineering Technology and Sciences*, 13(02), pp.229-257.
- [3]. Oladoja, T., 2024. Artificial Intelligence-Driven Innovations in VLSI, DevOps Security, and Cloud-Native Platforms: Addressing Challenges in Modern Technology Development.
- [4]. Moretti, M.C., 2021. Intelligent Real-Time Software Optimization Framework: Deep Learning-Enhanced Hybrid Fuzzy Model with WPM, TOPSIS, and PSO for Agentic Negotiation in Autonomous Systems. *International Journal of Engineering & Extended Technologies Research (IJEETR)*, 3(6), pp.4009-4013.
- [5]. Rousseau, C.A., 2024. A cloud-native enterprise framework enabling AI-driven automation governance secure networks mobile platforms and ethical decision intelligence. *International Journal of Research and Applied Innovations*, 7(5), pp.11426-11436.
- [6]. Dähling, S., Razik, L. and Monti, A., 2021. Enabling scalable and fault-tolerant multi-agent systems by utilizing cloud-native computing. *Autonomous Agents and Multi-Agent Systems*, 35(1), p.10.
- [7]. Katta, T.B., 2023. Adaptive AI-driven integration pipelines for efficient data and process orchestration in cloud-native environments. *International Journal of Research and Applied Innovations*, 6(1), pp.8363-8374.
- [8]. Patel, B.D., 2020. Architecting Enterprise Decision Intelligence through Cloud Native Computing and AI Powered Security Analytics. *International Journal of Advanced Research in Computer Science & Technology (IJARCST)*, 3(5), pp.3711-3717.
- [9]. Rajasekar, M., 2023. Agentic AI-Driven CI/CD for Secure and Waste-Reduced SAP Deployments in Healthcare Hybrid Cloud Environments. *International Journal of Research and Applied Innovations*, 6(3), pp.8916-8921.
- [10]. Reddy, V.A.R., 2023. Orchestrating the Future Autonomous Healthcare Data Pipeline Management through Agentic AI Architectures. *International Journal of Advanced Research in Computer Science & Technology (IJARCST)*, 6(2), pp.7979-7992.
- [11]. Habibi, P. and Leon-Garcia, A., 2024. Slicesphere: Agile service orchestration and management framework for cloud-native application slices. *Ieee Access*, 12, pp.169024-169049.
- [12]. Abhireddy, N., 2024. AI-driven cloud data orchestration for real-time enterprise decision intelligence systems. *International Journal of Research and Applied Innovations*, 7(6), pp.11963-11983.
- [13]. Mitchell, B.S., Mancoridis, S. and Kashyap, J., 2024, December. On the automatic identification of misconfiguration errors in cloud native systems. In *Proceedings of the 2024 7th Artificial Intelligence and Cloud Computing Conference* (pp. 539-548).
- [14]. Punzo, G., Tewari, A., Butans, E., Vasile, M., Purvis, A., Mayfield, M. and Varga, L., 2020. Engineering resilient complex systems: the necessary shift toward complexity science. *IEEE Systems Journal*, 14(3), pp.3865-3874.
- [15]. Ramaswamy, Y., 2020. Resilience engineering in DevOps: Fault injection and chaos testing for distributed systems. *NeuroQuantology*, 18(12), pp.337-347.
- [16]. Chava, S.C., 2024. Chaos Engineering for Multi-Cloud Resilience. *work*, 8, p.10.
- [17]. Naghshbandi, S.N., Varga, L., Purvis, A., McWilliam, R., Minisci, E., Vasile, M., Troffaes, M., Sedighi, T., Guo, W., Manley, E. and Jones, D.H., 2020. A review of methods to study resilience of complex engineering and engineered systems. *Ieee Access*, 8, pp.87775-87799.
- [18]. Torkura, K.A., Sukmana, M.I., Cheng, F. and Meinel, C., 2020. Cloudstrike: Chaos engineering for security and resiliency in cloud infrastructure. *IEEE Access*, 8, pp.123044-123060.
- [19]. Dedousis, P., Stergiopoulos, G., Arampatzis, G. and Gritzalis, D., 2023.

- Enhancing operational resilience of critical infrastructure processes through chaos engineering. *IEEE Access*, 11, pp.106172-106189.
- [20]. Bedoya, M., Palacios, S., Díaz-López, D., Laverde, E. and Nespoli, P., 2024. Enhancing DevSecOps practice with Large Language Models and Security Chaos Engineering: M. Bedoya et al. *International Journal of Information Security*, pp.1-24.
- [21]. Jernberg, H., Runeson, P. and Engström, E., 2020, October. 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).
- [22]. Hwang, J., Kale, G., Patel, P.P., Vishwakarma, R., Aliasgari, M., Hedayatipour, A., Rezaei, A. and Sayadi, H., 2023. Machine learning in chaos-based encryption: Theory, implementations, and applications. *IEEE access*, 11, pp.125749-125767.
- [23]. Meiklejohn, C.S., Estrada, A., Song, Y., Miller, H. and Padhye, R., 2021, November. Service-level fault injection testing. In *Proceedings of the ACM Symposium on Cloud Computing* (pp. 388-402).
- [24]. Gudelli, V.R., 2022. Microservice architecture: Challenges, solutions, and best practices. *International Research Journal of Modernization in Engineering Technology and Science*, 4(9), pp.2698-2712.
- [25]. Dondapati, K., 2020. Robust software testing for distributed systems using cloud infrastructure, automated fault injection, and XML scenarios. *World Journal of Advanced Engineering Technology and Sciences*, 8(2).
- [26]. Bencherqui, A., Tahiri, M.A., Karmouni, H., Alfidi, M., El Afou, Y., Qjidaa, H. and Sayyouri, M., 2024. Chaos-enhanced archimede algorithm for global optimization of real-world engineering problems and signal feature extraction. *Processes*, 12(2), p.406.
- [27]. Al-Said Ahmad, A., Al-Qora'n, L.F. and Zayed, A., 2024. Exploring the impact of chaos engineering with various user loads on cloud native applications: an exploratory empirical study: AA-S. Ahmad et al. *Computing*, 106(7), pp.2389-2425.
- [28]. Harve, B.M., Bidkar, D.M., Krishnappa, M.S., Pandey, G., Jayaram, V., Veerapaneni, P.K. and Mehta, G., 2024, December. The cloud-native revolution: Microservices in a cloud-driven world. In *2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)* (pp. 1043-1048). IEEE.
- [29]. Theodoropoulos, T., Rosa, L., Benzaid, C., Gray, P., Marin, E., Makris, A., Cordeiro, L., Diego, F., Sorokin, P., Girolamo, M.D. and Barone, P., 2023. Security in cloud-native services: A survey. *Journal of Cybersecurity and Privacy*, 3(4), pp.758-793.
- [30]. Soumplis, P., Kontos, G., Kokkinos, P., Kretsis, A., Barrachina-Muñoz, S., Nikbakht, R., Baranda, J., Payaró, M., Mangues-Bafalluy, J. and Varvarigos, E., 2024. Performance optimization across the edge-cloud continuum: A multi-agent rollout approach for cloud-native application workload placement. *SN computer science*, 5(3), p.318.
- [31]. Performance Analysis of Wireless Sensor Networks for Smart Monitoring Applications. (2016). *International Journal of Humanities and Information Technology*, 1(04), 10-29. <https://doi.org/10.21590/ijhit.01.04.04>
- [32]. Distributed Data Processing Frameworks for Large-Scale Healthcare Analytics. (2019). *International Journal of Advance Industrial Engineering*, 7(04), 1-10. <https://doi.org/10.14741/ijaie/v.7.4.01>