

# Survey of Artificial Intelligence-Driven Test Engineering for Secure Cloud Native Systems

**Dr. Puneet Garg**<sup>1</sup>

<sup>1</sup> Associate Professor, Department of CSE, AIKIET Group of Institutions, Delhi NCR Ghaziabad, India

**ABSTRACT:** Cloud-native technologies, which include microservices, containers, Kubernetes, and serverless computing, have been growing rapidly in recent times and have revolutionized the world of software development. Although the emergence of these technologies is beneficial in making software more scalable, flexible, and deployable, there exist certain difficulties that come about with them in software testing and security. Traditional methods of testing have found themselves ill-equipped to deal with such complex and dynamic environments, where artificial intelligence emerges as an ideal solution. The survey aims to study the role of AI-enabled test engineering strategies in creating secure cloud-native systems. This includes investigating the use of Machine Learning, Deep Learning, Natural Language Processing, and Reinforcement Learning in planning, testing, and deploying software. This survey will focus on various AI-enabled strategies in automated test case creation, test data creation, defect detection, test priority analysis, and deployment. Security issues such as malware detection, unauthorized access, runtime security threats, and decentralized access control are some other aspects to be covered in this survey. Some of the advantages of using AI-driven strategies for testing include greater automation, accuracy, and quicker deployment. On the downside, there are certain challenges such as the need for high-quality data, explainability, and computational complexity.

**KEYWORDS:** Test Automation, AI-Driven Test Engineering, Cloud-Native Systems, Test Automation, Automated Defect Detection, Kubernetes.

## 1. Introduction

Cloud-native development has transformed the way applications are designed, deployed, and managed, enabling businesses to harness the full potential of cloud computing. Unlike traditional monolithic applications, cloud-native applications leverage microservices, containerization, and dynamic orchestration to achieve scalability, resilience, and rapid deployment cycles [1][2][3]. This shift aligns with the increasing demand for systems that can adapt to fluctuating workloads and evolving business needs.

By adopting principles such as statelessness and automation, organizations can reduce infrastructure management overhead and focus on delivering value through innovation [4]. Additionally, the architecture supports continuous integration and deployment (CI/CD), allowing for faster time-to-market and improved product quality.

Software testing is a critical element of software quality assurance and represents the ultimate review of

specification, design and coding. Software testing is the process of testing the functionality and correctness of software by running it. Software testing is usually performed for one of two reasons: defect detection, and reliability estimation. The problem of applying software testing to defect detection is that software can only suggest the presence of flaws, not their absence (unless the testing is exhaustive) [5]. The problem of applying software testing to reliability estimation is that the input distribution used for selecting test cases may be flawed. The key to software testing is trying to find the modes of failure - something that requires exhaustively testing the code on all possible inputs. Software Testing, depending on the testing method employed, can be implemented at any time in the development process.

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into software testing has introduced transformative methodologies, enhancing efficiency, accuracy, and scalability. This paper explores contemporary advancements in AI-driven software testing, highlighting frameworks, tools, and techniques

that optimize the testing lifecycle [6]. By analyzing innovations such as AI-enhanced test automation, predictive analytics for defect detection, and large language models (LLMs) for test generation, present a comprehensive overview of the current landscape [7].

The landscape of software development has undergone a radical transformation in recent years, with the increasing adoption of Agile methodologies, DevOps pipelines, continuous integration/continuous delivery (CI/CD), and cloud-native architectures [8]. These trends demand faster, more frequent software releases without compromising quality. In this high-pressure environment, software testing has emerged as a bottleneck in the development lifecycle, particularly when relying on traditional manual or semi-automated testing methods. The complexity, volume, and velocity of modern software applications have outpaced the capabilities of conventional test automation, prompting a shift toward more intelligent and scalable solutions powered by Artificial Intelligence (AI) and Machine Learning (ML).

### A. Structure of the Paper

This paper is structured to evaluate the Effectiveness of Zero-Trust-Based IDS Against Advanced Persistent Threats in Cloud Infrastructure: Section II Cloud-Native Systems Overview Section III Security Challenges in Cloud-Native Systems IV Artificial Intelligence in Software Engineering Section V Benefits and Challenges. Section VI provides a survey of the literature. Section VII provides significant conclusions and suggestions for the future.

## 2. Cloud-Native Systems Overview

Cloud computing has transformed the way organizations develop, deploy, and manage modern applications by providing computing resources and services over the internet. Depending on the level of abstraction offered, cloud services are commonly categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS), and Function as a Service (FaaS) [9]. These service models enable organizations to reduce infrastructure complexity, improve resource utilization, and accelerate application delivery [10]

Cloud-native computing extends cloud capabilities by adopting architectural principles and technologies specifically designed for dynamic and distributed environments. Cloud-native systems emphasize scalability, resilience, agility, and automation through the use of microservices, containers, orchestration platforms, and serverless computing. Middleware plays a critical role in cloud-native environments by enabling communication, interoperability, resource management, and fault tolerance across distributed

services [11].

### A. Microservices

Microservices architecture is a foundational component of cloud-native systems, where applications are decomposed into small, independently deployable services that communicate through lightweight APIs [12]. Unlike monolithic architectures, microservices improve modularity, scalability, maintainability, and fault isolation. Each service can be developed, deployed, and scaled independently, enabling organizations to respond rapidly to changing business requirements. Common design patterns such as API Gateway, Service Discovery, and Circuit Breaker further enhance the reliability and efficiency of microservice-based applications.

### B. Containers

Containers provide lightweight and portable environments for packaging and executing applications along with their dependencies. Technologies such as Docker have become widely adopted for containerization, enabling consistent application behavior across development, testing, and production environments [13]. In cloud-native systems, containers support the Single Responsibility Principle, where each container performs a specific function, improving maintainability, scalability, and resource efficiency.

### C. Container Orchestration

As cloud-native applications often consist of numerous containerized services, orchestration platforms are essential for managing their lifecycle. Kubernetes has emerged as the leading container orchestration platform, providing automated deployment, scaling, service discovery, load balancing, and fault recovery [14]. By automating operational tasks, orchestration platforms enhance system reliability and support agile DevOps practices in large-scale distributed environments [15].

### D. Serverless Computing

Serverless computing represents a cloud execution model in which infrastructure management is abstracted from developers, allowing them to focus solely on application logic [16]. Although servers remain part of the underlying infrastructure, their provisioning, maintenance, and scaling are handled automatically by cloud providers. Function as a Service (FaaS) platforms, such as AWS Lambda, Google Cloud Functions, and Azure Functions, execute event-driven functions on demand, reducing operational overhead and improving resource efficiency [17]. Serverless computing further enhances the scalability and flexibility of cloud-native applications by enabling automatic scaling based on workload requirements.

### 3. Security Challenges in Cloud-Native Systems

Cloud-native systems have transformed modern application development through microservices, containers, orchestration platforms, and serverless computing. However, their distributed and highly dynamic nature introduces several security challenges that differ significantly from those of traditional monolithic systems. The large attack surface, complex service interactions, and continuously changing infrastructure make cloud-native environments attractive targets for cyberattacks. Ensuring confidentiality, integrity, and availability therefore remains a critical concern.

#### A. Security Challenges in Cloud-Native Systems

Despite their scalability and flexibility, cloud-native systems introduce significant security challenges due to their distributed and dynamic nature. The use of microservices, containers, and orchestration platforms increases the attack surface and complicates security management. Key challenges include securing inter-service communication, protecting sensitive data, managing access control, detecting threats, and ensuring compliance in rapidly changing environments.

##### 1) Dynamic and Ephemeral Infrastructure

Cloud-native environments are highly dynamic, with containers and microservices being continuously created, scaled, migrated, and terminated according to workload demands. This rapid change makes security monitoring, vulnerability management, and policy enforcement more challenging than in traditional static environments. Conventional security tools often struggle to provide real-time protection in such rapidly evolving systems.

##### 2) Container Security Vulnerabilities

Containers have become the primary deployment unit in cloud-native applications. Although they offer portability and resource efficiency, vulnerabilities may arise from insecure container images, misconfigurations, excessive privileges, outdated software packages, and weak isolation mechanisms. Attackers can exploit these weaknesses to gain unauthorized access or compromise application workloads.

##### 3) Microservices Communication Security

Cloud-native applications rely heavily on communication between multiple microservices through APIs and service meshes. Securing service-to-service communication is essential because attackers may intercept, modify, or spoof messages if proper authentication, authorization, and encryption mechanisms are not implemented. API vulnerabilities

further increase the attack surface of cloud-native systems.

#### 4) Malware and Runtime Attacks

The distributed nature of cloud-native platforms makes malware detection more complex [18][19]. Advanced threats such as ransomware, cryptojacking malware, and container escape attacks can evade traditional security solutions. Runtime attacks targeting containers and Kubernetes environments have emerged as significant security concerns requiring continuous monitoring and behavioral analysis.

#### 5) Unauthorized Access and Data Breaches

Cloud-native systems frequently handle sensitive business and user data across distributed services [20]. Weak authentication mechanisms, misconfigured permissions, and insecure APIs can lead to unauthorized access and data breaches. Protecting sensitive information requires strong identity management, encryption, and access control policies.

#### 6) Decentralized Access Control and Identity Management

Unlike monolithic applications that often rely on centralized security controls, cloud-native systems distribute security responsibilities across numerous services. Managing authentication, authorization, and user identities consistently across microservices becomes increasingly difficult as system complexity grows. Inadequate access control mechanisms may enable lateral movement by attackers after an initial compromise [21].

#### B. Security Defense Mechanism in Cloud-Native Systems

To address these challenges, researchers and practitioners have proposed several security defense mechanisms designed specifically for cloud-native environments.

##### 1) Moving Target Defense (MTD)

MTD is a strategy that modifies specific system components to make it more difficult for attackers to carry out successful attacks, lowering the attack capabilities [22]. The primary objective is to stop attackers from using information they have learned about target systems due to their homogeneous composition and software monoculture.

##### 2) Container Security Issues and Their Defenses

A lightweight and resource-effective deployment has become easy with Docker container integration in microservice and cloud-native applications [23]. Although obtain significant advantages, security issues have arisen due to the heterogeneity and unnecessary

dependencies and components.

### 3) Security-Attack-Based Protections

Researchers have adopted several strategies to protect against several attacks, especially in a cloud-based system. Concerning cross-site scripting (XSS), Bui et al. Found vulnerabilities in the APIs of cloud-based applications where external extensions might result in security flaws [24]. For example, adversaries could exploit cloud services' document sharing and messaging aspects to deliver malicious input. Instead, they provided guidelines, especially with coding practices and security enforcement.

### 4) Defense-In-Depth

Even when using simple-to-manage low-entropy authentication secrets, a flexible communication system with a high level of security may be created by combining standard cryptographic primitives. The method offers encryption, forward secrecy, and protection against replay attacks, even for out-of-order communication [25].

### 5) Identity and Access Management Solutions

Modern cloud-native security frameworks incorporate identity-centric approaches such as OAuth 2.0, multi-factor authentication (MFA), role-based access control (RBAC), and Zero Trust security principles. These mechanisms ensure that users and services receive only the permissions necessary to perform their designated functions.

### 6) Continuous Monitoring and Threat Detection

Continuous monitoring solutions provide real-time visibility into cloud-native environments by collecting logs, metrics, and security events. Advanced threat detection techniques leveraging artificial intelligence and machine learning can identify anomalous behavior, detect attacks early, and support rapid incident response [26][27][28].

## 4. Artificial Intelligence in Software Engineering

Artificial Intelligence (AI) has become an integral part of modern software engineering, enabling automated decision-making, prediction, and vulnerability detection [29]. Techniques such as machine learning, deep learning, data mining, and natural language processing help improve software quality, prioritize testing efforts, and reduce the time required to identify security vulnerabilities throughout the software development lifecycle.

### A. Artificial Intelligence in Planning

AI algorithms have the potential to revolutionize project management by analyzing vast amounts of

historical project data and providing valuable insights to support decision-making processes. These algorithms can identify patterns, correlations, and trends that may not be apparent to human managers, enabling them to make more informed decisions.

Machine learning models, such as decision trees, random forests, and support vector machines, can be trained on past project data to predict project outcomes, identify potential risks, and recommend appropriate mitigation strategies.

Accurate project timeline prediction is crucial for effective project planning and resource allocation. AI techniques, such as long short-term memory (LSTM) networks and recurrent neural networks (RNNs), have shown great promise in this area. These deep learning models can analyze past project data, considering various factors such as task durations, resource availability, and dependencies, to generate reliable timeline predictions.

Requirements gathering is a critical phase in the software development lifecycle, and AI can significantly streamline this process [30]. Natural language processing (NLP) techniques can be applied to analyze user feedback, customer reviews, and project documentation to automatically extract and prioritize requirements.

Maintaining accurate and up-to-date project documentation is essential for effective communication and collaboration among team members. AI algorithms can automate the generation of project documentation, such as user stories and acceptance criteria, based on the collected requirements.

Effective allocation of human resources is crucial for maximizing project efficiency and minimizing costs. AI-based recommender systems can assist in matching project tasks with the most suitable team members based on their skills, experience, and workload.

Distributing tasks among team members in a dynamic and adaptive manner is essential for ensuring smooth project progress. Reinforcement learning algorithms can be employed to dynamically assign tasks to team members based on real-time project progress and individual performance metrics. These algorithms learn from the outcomes of previous task assignments and continuously adapt their strategies to maximize overall project efficiency.[31]

### B. Artificial Intelligence in Testing

Automated test scenario creation: AI algorithms can analyze system requirements and generate test scenarios automatically. These scenarios cover different input combinations, edge cases, and potential failure points.

- **Intelligent test data generation:** Machine learning models can generate realistic test data based on historical

data patterns and domain knowledge [32]. This ensures thorough testing coverage and reduces manual effort in creating test data.

- **Parallel test execution:** AI-based test orchestration frameworks can optimize test execution by intelligently distributing tests across multiple machines or cloud instances. This reduces overall testing time and improves resource utilization.

- **Adaptive test prioritization:** Reinforcement learning algorithms can dynamically prioritize test cases based on their likelihood of revealing defects. This allows for more efficient testing cycles and faster feedback loops.

- **Identifying unexpected behaviors:** Unsupervised learning algorithms, such as autoencoders and clustering techniques, can detect anomalies and unexpected behaviors during testing [33]. These algorithms learn normal system behavior and flag deviations as potential issues.

- **Pinpointing performance bottlenecks:** AI models can analyze system logs and performance metrics to identify performance bottlenecks during testing [34]. These models can correlate performance issues with specific test scenarios or system components.

### C. Artificial Intelligence in Deployment

Intelligent rollout planning: AI algorithms have the potential to revolutionize deployment strategies by considering a wide range of factors, such as user segments, geographic regions, and system dependencies. These algorithms can analyze historical deployment data, user behavior patterns, and system architectures to create optimal rollout plans. By leveraging machine learning techniques, such as clustering and graph analysis, AI models can identify the most suitable user groups or regions for initial deployments, minimizing the risk of adverse impacts.

- **Automated canary releases:** Canary releases are a popular deployment technique that involves gradually rolling out new features or updates to a small subset of users before a full-scale deployment. AI can automate and optimize the selection of users or servers for canary releases [30]. Machine learning models can analyze user profiles, usage patterns, and system metrics to identify the most representative and low-risk subset for testing [35].

- **Continuous Integration/Continuous Deployment (CI/CD) Automated build and release pipelines:** AI can significantly enhance the efficiency and reliability of CI/CD pipelines by automating various tasks and adapting to changes in the codebase. AI-powered CI/CD pipelines can intelligently trigger

builds based on predefined criteria, such as code commits or schedule, and dynamically adjust the build configurations based on the characteristics of the changes. These pipelines can also incorporate machine learning models to predict the likelihood of build failures and proactively take corrective actions.

- **Automated environment setups Infrastructure as Code (IaC) optimization:** Infrastructure as Code (IaC) has emerged as a key practice in modern software deployment, enabling the management of infrastructure through version-controlled code [36]. AI algorithms can optimize IaC configurations by analyzing resource usage patterns and performance metrics. These algorithms can identify inefficiencies, such as overprovisioned resources or underutilized instances, and suggest optimizations to improve resource utilization and cost-effectiveness.

- **Dynamic resource provisioning:** Reinforcement learning in cloud environments enables dynamic and intelligent allocation based on real-time application demands. These algorithms can continuously monitor application performance metrics, such as response times, throughput, and resource utilization, and make autonomous decisions to adjust resource allocations accordingly.

### 5. Benefits and Challenges

The matrix illustrates the relationship between key challenges and strategic imperatives in AI-driven test automation. Data quality requires effective training and governance, while algorithm bias demands continuous improvement and transparency [37]. Tool complexity and integration issues can be addressed through better workflow integration and optimization. Lack of explainability highlights the need for transparency, and cultivating an organizational culture supports collaboration, innovation, and successful AI adoption. Overall, the matrix provides a strategic framework for overcoming challenges and enabling effective AI-driven testing.

- **Test Oracle:** The biggest challenge: The test oracle problem is a companion of every researcher and practitioner working in the field of software testing. It has been there from the inception of the software testing conundrum and the problem is expected to stay with us for a longer duration or maybe forever.

- **Availability of Data:** Any model in AI must be trained and tested before being deployed in production [38]. The efficiency and effectiveness of a model are highly correlated to the amount of data with which a model is trained and tested.

- **Adaptiveness to data:** AI models are highly dependent on the data with which they are trained and tested. An important phase in the production of an AI

model is the collection of robust datasets from real-world scenarios and the use of that data to train a model generalized to fit that data. Such a model assumes future data and historic data (data with which the model is trained) to be from the same distribution

- **Identifying Test Data:** Every AI model must be tested 9 thoroughly before being put into production. Model testing is like a black-box technique where the structural or logical information regarding the model is not a necessity. Rather comprehensive information and understanding regarding the testing data is required.[39]

- **Exhaustive search space leads to generality loss:** For most of the optimization problems in search-based software testing, the AI algorithm has to exhaustively search for the solution or the goal. Although sub-optimal search strategies have been identified and implemented so far, they work for a particular class of problems.

- **Exploitation of Multicore Computation:** A lot of AI techniques are highly computationally expensive making them potentially incompatible with large-scale problems faced by software testers [40]. With the recent advancements in computing infrastructure, Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) have been incorporated at scale for these techniques.

## 6. Literature Review

This section provides a survey of previous work that has been done on a comparable subject and keywords that are relevant to it. Table I summarizes recent studies on AI-driven test engineering for secure cloud-native systems, highlighting proposed frameworks, methodologies, and outcomes. The reviewed research demonstrates improvements in testing efficiency, security, resilience, automation, and operational performance.

Mr. Jay Bharat Mehta (2026) tests architecture patterns for Distributed Systems In the modern cloud-native environment, the architecture of testing is changing to accommodate distributed systems. These systems are made up of a number of services that could span across different geographies, platforms or clouds. The test engineering designs of this type of distributed system should be flexible, resilient, and efficient to sustain the quality of the software at scale [41]

Sree et al. (2025) the incorporation of Artificial Intelligence (AI) in cloud-native environment is revolutionizing the process of software engineering, upkeep, and evolution. Automation, resource optimization, and decision-making processes are all greatly improved by AI-driven tools and techniques like

machine learning, automated code creation, predictive analytics, and intelligent monitoring. The merging of cloud-native architectures and AI approaches is examined in this study, which offers a systematic examination of how AI improves the design, testing, deployment, and operations phases of the software development lifecycle.[42]

Kale (2025) model establishes a smart fault injection engine that can emulate realistic and adaptive failure modes according to past system dynamics and run-time environment. Simultaneously, a predictive analytics engine is used to predict possible failures, high-risk components, and prioritize test execution using machine learning models. Experimental evidence shows that it has a large improvement in fault-detection rates, prediction accuracy, and system resilience, without incurring a large performance overhead. The study helps to improve intelligent DevOps in the future by facilitating self-adaptive and scalable efficient testing pipelines, which lead to the development of robust and reliable delivery of cloud-native applications [43].

K. Jayabalan al. (2025) the cloud-native approach utilizes a microservices architecture combined with integrated DevSecOps practices, AI-enabled fraud detection, and predictive autoscaling to achieve increased performance and security. Objective evaluations against four state-of-the-art fintech frameworks demonstrate massive improvements, with 0.5 vulnerabilities per KLOC (72% reduction), 120 ms p95 latency (65% lower), an SLA adherence of 99.96%, and an F1-score of 0.945 for fraud detection (5–9% improvement). Additionally, overall compliance with PCI-DSS and GDPR was 99.2%, and usability tests using the SUS could achieve a score of 88, compared with existing solutions achieving 10–15 points lower [44]

Srikanth Chakravarthy Vankayala (2024) synthesizes foundational work in software testing (2000–2010), advances in AI-driven testing (2015–2023), and modern cloud-native engineering practices to propose a unified, adaptive testing paradigm. Experimental insights drawn from prior research and empirical studies indicate that such intelligent, feedback-driven approaches can significantly enhance defect detection rates, optimize resource utilization, improve test coverage, and increase overall system resilience in complex, large-scale software ecosystems.[45]

Panyala (2023) presents a framework of continuous integration, continuous deployment, security automation, observability, and self-healing cloud-native platforms by building on existing theories, practices and research in relevant contemporary literature until 2022. Based on the analysis conducted, can conclude that AI-based DevOps can greatly decrease the time spent at the production stage and enhance the productivity and

stability of the system.[46]

Shah (2022) how Artificial Intelligence (AI), particularly machine learning (ML), anomaly detection, and reinforcement learning, can augment cyber resilience across cloud-native platforms. A layered AI-

augmented architecture is proposed, covering telemetry ingestion, behavioral feature engineering, ML-based detection, and automated response orchestration. Key evaluation metrics and datasets are discussed, alongside limitations such as adversarial AI, data imbalance, and explainability concerns [47]

**TABLE I. RESEARCH ON SURVEY OF ARTIFICIAL INTELLIGENCE DRIVEN TEST ENGINEERING FOR SECURE CLOUD NATIVE SYSTEMS**

| Author(s) & Year                       | Key Contribution                                                                                      | Methodology / Approach                                                                                                                                    | Findings / Outcomes                                                                                                                          |
|----------------------------------------|-------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Mr. Jay Bharat Mehta (2026)            | Proposed testing architecture patterns for distributed systems in cloud-native environments.          | Flexible, resilient, and scalable test engineering frameworks for geographically distributed services.                                                    | Improved software quality assurance and testing efficiency for large-scale distributed systems.                                              |
| Sree et al. (2025)                     | Explored AI integration in cloud-native software engineering.                                         | Systematic review of AI-driven tools including machine learning, predictive analytics, automated code generation, and intelligent monitoring.             | AI significantly enhances software design, testing, deployment, resource optimization, and operational decision-making.                      |
| Kale (2025)                            | Developed an AI-driven adaptive testing architecture for cloud-native systems.                        | Smart fault injection, predictive analytics, centralized test orchestration, observability, and feedback-driven learning integrated with CI/CD pipelines. | Improved fault detection, prediction accuracy, resilience, and enabled self-healing testing ecosystems with minimal performance overhead.    |
| Jayabalan et al. (2025)                | Proposed a cloud-native fintech framework with AI-enabled security and performance optimization.      | Microservices architecture, DevSecOps integration, AI-based fraud detection, and predictive autoscaling.                                                  | Achieved lower vulnerabilities, reduced latency, higher SLA compliance, improved fraud detection accuracy, and better regulatory compliance. |
| Srikanth Chakravarthy Vankayala (2024) | Presented a unified adaptive testing paradigm combining traditional and AI-driven testing approaches. | Synthesis of software testing research, AI-driven testing techniques, and cloud-native engineering practices.                                             | Enhanced defect detection, resource utilization, test coverage, and system resilience in large-scale software ecosystems.                    |
| Panyala (2023)                         | Investigated AI-based DevOps for cloud-native platforms.                                              | Framework incorporating CI/CD, security automation, observability, and self-healing mechanisms.                                                           | Reduced production deployment time while improving system productivity, reliability, and stability.                                          |
| Shah (2022)                            | Proposed an AI-augmented cyber resilience architecture for cloud-native systems.                      | Machine learning, anomaly detection, reinforcement learning, telemetry analysis, and automated response orchestration.                                    | Improved threat detection speed, reduced false positives, enhanced incident response, and promoted proactive cloud security.                 |

## 7. Conclusion and Future Work

Artificial Intelligence (AI) has been identified as one of the most revolutionary technologies that can be used in software testing, facilitating smart automation and predictive analysis for cloud-native applications. The survey investigated how the use of AI-based test engineering can be leveraged to improve the quality, security, and performance of software in cloud-native application environments that leverage microservices, containers, Kubernetes, and serverless computing platforms. Technologies associated with AI, such as machine learning, deep learning, natural language processing, and reinforcement learning have exhibited

significant promise in tasks related to automated test case generation, defect prediction, anomaly detection, test case prioritization, and deployment. Nevertheless, a number of challenges need to be overcome to fully realize the benefits of AI-driven test engineering frameworks.

Further research efforts are required for creating explainable and trustworthy models of AI. The use of Large Language Models and Generative AI in automated test generation, requirement analysis, and security testing would be an ideal way forward. The development of self-healing and autonomous testing tools for dynamic cloud-native applications would also be an excellent approach. Studies involving AI security testing to detect zero-day attacks, container vulnerabilities, and

cyberattacks would be another area of interest. Moreover, the establishment of standard datasets and benchmarking measures would help achieve efficient test engineering solutions via AI.

## REFERENCES

1. M. Amaral, J. Polo, D. Carrera, I. Mohamed, M. Unuvar, and M. Steinder, "Performance Evaluation of Microservices Architectures Using Containers," in 2015 IEEE 14th International Symposium on Network Computing and Applications, IEEE, Sep. 2015, pp. 27–34. doi: 10.1109/NCA.2015.49.
2. H. N. Dholariya, "Regulatory-Grade Autonomous Data Modernization: The RAMA Framework for Compliance-Aware AI-Native Cloud Architectures," J. Comput. Anal. Appl., vol. 35, no. 1, pp. 883–898, Jan, Jan. 2026, doi: 10.48047/jocaaa.2026.35.01.40.
3. J. B. Mehta, "An Autonomous Dependency And Failure-Propagation Modeling System For Security-Critical Distributed Enterprise Cloud Computing Environments," 202641001713, 2026
4. G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," IEEE Access, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.
5. S. S. R. Ahamed, "Studying the Feasibility and Importance of Software Testing: An Analysis," vol. 1, no. 3, pp. 119–128, 2010.
6. R. Madhumita and D. Chandana, "Advancing Software Testing: Integrating AI, Machine Learning, and Emerging Technologies," Int. J. Res. Eng. Sci. Manag., vol. 7, no. January, p. 9, 2025.
7. B. Krishnan, S. Perla, S. Maddela, and R. Lingam, "Adaptive Multi-Cloud Infrastructure for CRM Analytics: Real-Time ML and Data Sync with LLMs," in 2025 IEEE 3rd Global Conference on Wireless Computing and Networking (GCWCN), Lonawala, Maharashtra, India: IEEE, Nov. 2025, pp. 1–8. doi: 10.1109/GCWCN66157.2025.11448404.
8. H. V. Pandhare, "Future of Software Test Automation Using AI/ML," Int. J. Eng. Comput. Sci., vol. 14, no. 05, pp. 27159–27182, 2025, doi: 10.18535/ijecs.v14i05.5139.
9. H. Jiang, J. Nagra, and P. Ahammad, "SoK: Applying Machine Learning in Security - A Survey," 2016.
10. M. R. C. Mukkolakkal, "InfraLLM: A Generic Large Language Model Framework for Production-Grade Microservice Auto-Scaling in Cloud Infrastructure," Int. J. Sci. Res. Mod. Technol., vol. 4, no. 11, pp. 113–123, 2025, doi: 10.38124/ijrsmt.v4i11.1023.
11. J. Kurma, "A Review of Security , Compliance , and Governance Challenges in Cloud-Native Middleware and Enterprise Systems," vol. 5, no. 1, 2022, doi: 10.15662/IJRAI.2022.0501003.
12. O. C. Oyeniran, A. O. Adewusi, A. G. Adeleke, L. A. Akwawa, and C. F. Azubuko, "Microservices architecture in cloud-native applications: Design patterns and scalability," Comput. Sci. IT Res. J., vol. 2, no. 1, pp. 92–106, 2024, doi: 10.51594/csitrj.v5i9.1554.
13. Z. Shen et al., "X-Containers," in Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, New York, NY, USA: ACM, Apr. 2019, pp. 121–135. doi: 10.1145/3297858.3304016.
14. P. Naayini and S. Kamatala, "Automating Infrastructure Platforms with Cloud, Kubernetes, and Site Reliability Engineering," Int. J. Comput. Tech., vol. 8, no. 6, pp. 1–9, November, 2021.
15. H. B. Dama, "A Survey of MySQL Database Administration Techniques and Best Practices," ESP J. Eng. Technol. Adv., vol. 6, no. 1, pp. 89–98, February, 2026.
16. N. Kodakandla, "Serverless Architectures: A Comparative Study of Performance, Scalability, and Cost in Cloud-native Applications," Iconic Res. Eng. Journals, vol. 5, no. 2, pp. 136–150, 2021.
17. T. Shah, "Cloud-Based Data Warehousing for Marketing Agility: Lessons from FinTech Migrations to Snowflake and AWS," Int. J. Adv. Res. Sci. Commun. Technol., vol. 4, no. 4, March, pp. 642–652, 2024, doi: 10.48175/IJARST-16000B.
18. K. Subramaniam, "Cloud Network Security: A Survey on Threats, Existing Solutions and the Path Forward," Sch. Comput. Augment. Intell. Arizona State Univ., pp. 1–7, 2022.
19. R. Keshava, S. K. Pandurangan, M. Sakthivanitha, S. G. S. Parmsivan, and R. Maruthi, "AI-Powered Algorithms for the Prevention and Detection of Computer Malware Infections," in 2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India: IEEE, 2025, pp. 1673–1680, September. doi: 10.1109/ICESC65114.2025.11212519.
20. B. Matthew, S. Gupta, and A. Sen, "Migrating Legacy MES System Data Containing BOM,

- Routing, and Serialization Records to a Cloud-Native Lakehouse," 2024.
21. T. Arif, B. Jo, and J. H. Park, "A Comprehensive Survey of Privacy-Enhancing and Trust-Centric Cloud-Native Security Techniques Against Cyber Threats," *Sensors*, vol. 25, no. 8, p. 2350, Apr. 2025, doi: 10.3390/s25082350.
22. H. Jin, Z. Li, D. Zou, and B. Yuan, "DSEOM: A Framework for Dynamic Security Evaluation and Optimization of MTD in Container-based Cloud," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–1, 2019, doi: 10.1109/TDSC.2019.2916666.
23. C. Patel, "A Review of Multi-Channel CRM Strategies Using Big Data and Cloud Integration," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 1, pp. 577–588, 2022.
24. T. Bui, S. Rao, M. Antikainen, and T. Aura, "XSS Vulnerabilities in Cloud-Application Add-Ons," in *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*, New York, NY, USA, NY, USA: ACM, Oct. 2020, pp. 610–621. doi: 10.1145/3320269.3384744.
25. K. Jander, L. Braubach, and A. Pokahr, "Defense-in-depth and Role Authentication for Microservice Systems," *Procedia Comput. Sci.*, vol. 130, pp. 456–463, 2018, doi: 10.1016/j.procs.2018.04.047.
26. O. Safaryan, E. Pinevich, E. Roshchina, L. Cherkesova, and N. Kolennikova, "Information system development for restricting access to software tool built on microservice architecture," *E3S Web Conf.*, vol. 224, p. 01041, Dec. 2020, doi: 10.1051/e3sconf/202022401041.
27. M. K. Shah, "Artificial Intelligence (AI)- Based Threat Detection Models for Privileged App Abuse in Modern Systems," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 6, no. 4, p. 11, 2026.
28. R. Lingam, S. Nagi, M. Chigurupati, and B. T. Myneni, "Resilient DevSecOps: Self-Healing Cloud-Native Systems via SRE-Driven AI Threat Detection and Response," in *2026 International Conference on Smart Futuristic Technology*, IEEE, Jan. 2026, pp. 1–6. doi: 10.1109/ICSFT66733.2026.11508049.
29. H. Sofian, N. A. M. Yunus, and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," *IEEE Access*, vol. 10, pp. 51021–51040, 2022, doi: 10.1109/ACCESS.2022.3174115.
30. S. Kumar, "Artificial Intelligence in Software Engineering: A Systematic Exploration of AI-Driven Development," *Int. J. Innov. Res. Sci. , Eng. Technol.*, vol. 13, no. 6, pp. 11903–11913, 2024, doi: 10.15680/IJIRSET.2024.1306220.
31. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second. The MIT Press, 2018.
32. E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, no. 1, pp. 1393–13696, 2020, doi: 10.1609/aaai.v34i09.7123.
33. M. Cinque, D. Cotroneo, and A. Pecchia, "Event Logs for the Analysis of Software Failures: A Rule-Based Approach," *IEEE Trans. Softw. Eng.*, vol. 39, no. 6, pp. 806–821, Jun. 2013, doi: 10.1109/TSE.2012.67.
34. S. Kumar, "Artificial intelligence in software engineering: A systematic exploration of AI-driven development," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 13, no. 6, pp. 11903–11909, 2024.
35. N. Dragoni et al., "Microservices: Yesterday, Today, and Tomorrow," in *Present and Ulterior Software Engineering*, M. Mazzara and B. Meyer, Eds., Cham: Springer International Publishing, 2017, pp. 195–216. doi: 10.1007/978-3-319-67425-4\_12.
36. S. Choi, J. Goo, E. Jeon, M. Yang, and M. Jang, "ELIS: Efficient LLM Iterative Scheduling System with Response Length Predictor," 2025. doi: 10.48550/arXiv.2505.09142.
37. M. H. Moghadam, "Machine learning-assisted performance testing," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA, NY, USA: ACM, Aug. 2019, pp. 1187–1189. doi: 10.1145/3338906.3342484.
38. K. Kamaraj, B. Lanitha, S. Karthic, P. N. Senthil Prakash, and R. Mahaveerakannan, "A Hybridized Artificial Neural Network for Automated Software Test Oracle," *Comput. Syst. Sci. Eng.*, vol. 45, no. 2, pp. 1837–1850, 2023, doi: 10.32604/csse.2023.029703.
39. J. Čegić, "Machine learning-based test data generation for safety-critical software," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, New York, NY, USA: ACM, Nov. 2020, pp. 1678–1681. doi: 10.1145/3368089.3418538.
40. A. Muhammad, "AI-Driven Testing Automation: Harnessing Machine Learning for Intelligent Test Case Creation and Predictive Defect Analysis," 2023.

41. J. B. Mehta, "AI-Driven Test Engineering For Cloud-Native Systems," *Int. J. Data Sci. IoT Manag. Syst.*, vol. 5, no. 1, Jan. 2026, doi: 10.64751/ijdim.2026.v5i1.297.
42. M. S. Sree, C. K. K. Reddy, K. Vaishnavi, and V. Harika, "AI-Powered Software Engineering for Cloud-Native Environments," 2025, pp. 57–86. doi: 10.4018/979-8-3693-9356-7.ch003.
43. P. Kale, "Architecting Autonomous Testing Pipelines for Cloud-Native Systems Using AI-Driven Fault Injection and Predictive Analytics," *Int. J. AI, Big Data, Comput. Manag. Stud.*, vol. 6, pp. 207–216, 2025, doi: 10.63282/3050-9416.IJAIBDCMS-V6I1P121.
44. K. Jayabalan, S. Parmsivan, G. Sunkara, M. Parikh, P. Challa, and V. Nutalapati, "Innovative Framework for Secure and Scalable Web and Mobile Application Development in Fintech: A User- Centric and AI - Driven Approach," in 2025 5th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS), IEEE, Nov. 2025, pp. 1499–1505. doi: 10.1109/ICUIS67429.2025.11380645.
45. S. C. Vankayala, "Intelligent Quality Assurance in Cloud-Native Systems: A Deep Learning and Reinforcement Learning Approach to Adaptive Test Coverage Optimization," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 11, no. 6, pp. 546–553, Nov. 2024, doi: 10.32628/IJSRSET2512555.
46. V. R. Panyala, "International Journal of Advanced Research in Education and Technology ( IJARETY ) AI-Augmented DevOps Frameworks for Accelerating Cloud-Native Platform Engineering at Scale," vol. 10, no. 1, 2023, doi: 10.15680/IJARETY.2023.1001028.
47. J. K. Shah, "AI-driven resilience in cloud-native big data platforms against cyberattacks," *J. Comput. Sci. Technol. Stud.*, vol. 4, no. 2, pp. 191–199, 2022, doi: 10.32996/jcsts.2022.4.2.23.