**Journal of Artificial Intelligence and Semiconductor Integrated Hardware (JAISIH)**
Volume 1 | Issue 2 | April-June 2026 | Pages 16-22
DOI: 10.5281/zenodo.21342922

**Review Article** OPEN ACCESS https://jaisih.org/index.php/jaisih/issue/archive

# A Review of Machine Learning Approaches for Software Testing in Secure Modern Systems

**Mr. Ram Pratap Singh**

*Department of Computer Science and Engineering, Lakshmi Narnia College of Technology, Bhopal*
rampratap@lnct.ac.in
**Corresponding Author:** *Ram Pratap Singh (rampratap@lnct.ac.in)

**Received:** 15 April, 2026 | **Revised:** 5 May 2026 | **Accepted:** 14 June 2026 |

***Abstract*—Software testing plays a critical role in ensuring the quality, reliability, and security of modern software systems. With the increasing complexity of applications driven by cloud computing, the Internet of Things (IoT), mobile technologies, and distributed architectures, traditional testing approaches often face challenges related to scalability, efficiency, and cost. Machine Learning (ML) has emerged as a promising solution to enhance software testing by automating testing activities and improving decision-making processes. This review paper presents a comprehensive analysis of ML-based approaches in software testing for secure modern systems. It examines key applications of ML, including defect prediction, fault detection, automated test case generation, test optimization, and vulnerability assessment. The paper also discusses the integration of Artificial Intelligence (AI) and Large Language Models (LLMs) in software testing and quality assurance. Furthermore, it explores security testing methodologies and emerging trends such as DevSecOps and continuous security testing. The review highlights the benefits, challenges, and limitations of current approaches while identifying future research directions for developing intelligent, scalable, and security-aware software testing frameworks that improve software quality and testing efficiency.**

**Keywords—*Machine Learning, Software Testing, Defect Prediction, Security Testing, Large Language Models (LLMs)***

## I. INTRODUCTION

Software systems have become increasingly complex due to the rapid growth of cloud computing, Internet of Things (IoT), mobile applications, and distributed architectures[1][2]. As modern secure software applications handle sensitive data and critical operations, ensuring their reliability, quality, and security has become a major concern. Software testing plays an important role in the Software Development Life Cycle (SDLC) by identifying defects, verifying system functionality, and ensuring that applications meet both functional and non-functional requirements[3].

Software testing is a critical activity in the Software Development Life Cycle (SDLC) that tries to check a software product's functionality, performance, and dependability. It involves executing the software under controlled conditions in order to find bugs, failures, and vulnerabilities[4]. If these flaws are not fixed, they can result in malfunctions, crashes, security breaches, and, ultimately, a poor user experience. In this aspect, software testing is a quality control tool with the capability of preventing costly failures, maintaining a company's reputation, and, most importantly, ensuring customer satisfaction.

While software testing is certainly important, it comes at a significant cost because it involves the execution of many test cases at various stages of the SDLC. These test cases must be repeatedly executed, not only during development but also when changes, additions, and updates are made[5]. Therefore, software testing is a resource-intensive process resulting in financial and time restrictions on software development initiatives. In this context, software development teams are always looking for cost-effective testing methods that can ensure defect detection and quality assurance.

Machine Learning (ML) is a process where machines learn from data using algorithms and can further predict or make decisions based on the data [6]. The data-centric learning approach has made Machine Learning powerful and widely accepted across various domains, including the software industry. ML enhances software testing by automating tasks such as test case generation and defect prediction, thereby speeding up the testing process and reducing manual effort. This efficiency allows teams to focus on critical issues more quickly, leading to faster releases and better product quality. ML also enables broader test coverage by exploring more inputs and scenarios, which means fewer bugs go undetected. It makes techniques like mutation testing more feasible, ensuring more thorough testing. Besides improving quality, ML helps cut down costs by minimizing labor-intensive tasks and reducing post-release fixes, making it both an efficient and cost-effective solution for software testing.

### A. Structure of the Paper

This paper is structured as follows: Section II discusses software testing, Section III presents ML-based testing approaches, Section IV covers security testing and emerging trends, Section V reviews recent literature, and Section VI concludes the paper with future research directions.

DOI: 10.5281/zenodo.21342922

## II. Role Of Software Testing In Modern Software Systems

Software testing has a crucial role in software engineering as it is essential for ensuring the quality, performance, security, and reliability of software systems[7][8]. Software testing refers to the process of systematic evaluation, inspection and verification of the software system to detect and correct errors or defects introduced by the software development or maintenance process and ensure that the software system is developed and delivered in accordance with user requirements and specification requirements [9].It is crucial for software to work or perform as per requirements but it is natural having bugs or defects in software. The bugs can be generated during development, bug fixing, feature addition, code refactoring, and even during software maintenance. Therefore, it is obvious for the development team to test the software under different scenarios before releasing it to the client.

### *A. Objectives and Importance of Software Testing*

The objective of testing is to discover the residual design errors before delivering to the customer.

- **Bug:** A software bug may be defined as a coding error that causes an unexpected defect, fault or flaw. In other words, if a program does not perform as intended, it is most likely a bug.
- **Error:** A mismatch between the program and its specification is an error in the program.
- **Defect:** Defect is the variance from the desired product attribute (it can be a wrong, missing or extra data). It can be of two types – Defect from the product or variance from customer/user expectations. It is a flaw in the software system and has no impact until it affects the user/customer and operational system. 90% of all the defects can be caused by process problems.
- **Failure:** A defect that causes an error in operation or negatively impacts a user/ customer.
- **Quality Assurance:** Is oriented towards preventing defects. Quality Assurance ensures all parties concerned with the project adhere to the process and procedures, standards and templates and test readiness reviews.
- **Quality Control:** quality control or quality engineering is a set of measures taken to ensure that defective products or services are not produced and that the design meets performance requirements.
- **Verification**: Verification ensures the product is designed to deliver all functionality to the customer; it typically involves reviews and meetings to evaluate documents, plans, code, requirement, and specifications; this can be done with checklists, issues list, walkthrough, and inspection meetings.
- **Validation:** Validation ensures that functionality, as defined in requirements, is the intended behavior of the product; validation typically involves actual testing and takes place after verifications are completed.
- **Software Reliability Estimation:** The objective of testing is to discover the residual design errors before delivering to the customer[10]. The failure data during the testing process are taken in down to estimate the software reliability. The testing process may function with regular feedback from the reliability analysis to the testers and designers.

### *B. Levels of Software Testing*

A level of testing is the stage at which the software must be tested. There are four recognized levels of testing, each designed to verify different aspects of the software system. These levels ensure that defects are identified systematically, improving software quality, reliability, and overall system performance throughout the Software Development Life Cycle (SDLC).

#### *1) Unit Testing*

In Unit testing, the smallest testable parts of an application called units are tested independently. It focuses on smallest element of the software system called module. That's why, it is also called module testing. It is done by developers. 1) Example Consider the example of Calculator. To conduct unit testing on the calculator system, individual modules like Addition, subtraction, multiplication, division must be tested independently.

#### *2) Integration Testing*

The purpose of integration testing is to expose faults in the interaction between integrated units. It is the process of testing the interface between two software units[11]. It can be done in three ways- Big-bang approach, top-down approach and bottom-up approach.

#### *3) System Testing*

System testing is nothing but the testing of complete system. It is the last test that the developer performs before delivering the software to public for acceptance testing. Different types of system testing are followed like usability testing, stress testing, regression testing etc. First and important step in system testing is to prepare System test plan.

#### *4) Acceptance Testing*

It is a level of testing where a system is tested for acceptability. It has various types' like- Alpha testing, beta testing, user acceptance testing and business acceptance testing[12][13]. It is done by end users. Its outcome provides an important quality indication for the customer to determine whether to accept or reject the product.

### *C. Challenges in Modern Software Testing*

Modern software testing faces several technical, organizational, and operational challenges that can affect software quality, testing efficiency, and timely project delivery. The major challenges are discussed below:

- **Lack of communication:** Communication gaps and misunderstandings between clients and development teams can lead to unclear requirements, resulting in inaccurate test cases and reduced testing effectiveness.
- **Missing documentation:** Incomplete or missing documentation of functional and non-functional requirements may cause developers and testers to overlook essential features[14], leading to unnecessary testing and implementation errors.
- **Diversity in testing environments:** The wide variety of devices, browsers, and platforms makes it challenging to ensure consistent application performance across all user environments.

DOI: 10.5281/zenodo.21342922

- **Inadequate testing:** Insufficient testing and limited regression testing can leave defects undetected, reducing software reliability and increasing the risk of failures after deployment.
- **Identifying the right automation tool or framework:** Selecting an inappropriate automation tool can reduce testing efficiency, increase costs, and create compatibility issues with project requirements.
- **Employing skilled testers or training existing teams:** Effective test automation requires skilled professionals. A lack of expertise or inadequate training can hinder automation implementation and maintenance.

## III. Machine Learning Approaches In Software Testing

Machine learning techniques are effective tools that can enhance the effectiveness of software testing processes. This technology can provide a better understanding of customers. The implementation of machine learning techniques in software testing can offer many opportunities for customers [15]. Invariant testing approaches are maintained effectively through the use of machine learning techniques, which directly impact software testing. Machine learning technology is a common model of software testing. The software testing process is fairly straightforward. Predictive analytical approaches are also used in software testing maintenance. With the help of a few resources, it is noticed that machine learning technology is effective than predictive analytics. An effective layered algorithm is implemented successfully to maintain the efficiency of machine learning technology [16]. The process of machine learning is used to analyze the functions on the level of the output. The function of software testing provides a scenario that is used in implementing the function of machine learning. The approach of software testing consists of different attributes based on the design of the process. The different activities are used in the practices of suitable aspects of the function of different analysis.

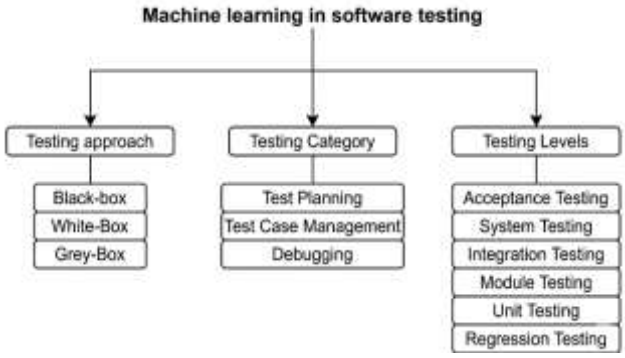


Fig. 1. Machine Learning in Software Testing: Approaches, Categories, and Levels

Figure 1 illustrates the role of machine learning in software testing by categorizing testing approaches, testing categories, and testing levels, providing a structured overview of its applications in modern software testing.

### *A. Machine Learning Techniques*

Machine learning is referred to as one of the integral branches of artificial intelligence. If explained simply, it is a field of research that emphasizes teaching cognitive activities to machines. It helps artificially constructed machines to function as human minds. With the help of machine learning huge amounts of information can be stored within machine databases which can be programmed and developed according to specific necessity[17]. Machine learning consists of two primary categories, one of which is referred to as supervised learning while the other is named unsupervised learning.

#### *1) Supervised Learning*

Supervised learning trains the machine learning algorithm using input samples and labels. Using a training dataset with proper labels to instruct the algorithm is supervised learning. Supervised learning algorithms are classified:

- **(a) Regression:** is determined by the variable that represents the output. If the output variable is continuous, then the job in question is referred to as a regression task. Predicting the price of a home and the price of a stock are both instances of regression problems.
- **(b)Classification:** Classification tasks use categorical variables such as color and form. Most machine learning applications employ supervised learning. Supervised learning methods include logistic regression, linear regression, SVMs, and random forests.

#### *2) Unsupervised Learning*

In Unsupervised learning, we train the machine learning algorithm with only the input examples but no output labels. The algorithm tries to learn the underlying structure of the input examples and discover patterns[18]. Unsupervised learning algorithms can be further categorized based on two tasks, namely Clustering and Association. In clustering, an algorithm such as k-means tries to discover inherent clusters or groups in the data.

### *B. ML-Based Test Case Generation and Optimization*

Machine Learning (ML) has significantly improved software testing by enabling the automatic generation and optimization of test cases. ML algorithms analyze software requirements, source code, historical testing data, and user behaviour patterns to create relevant test cases with minimal human intervention. In addition, ML techniques help prioritize, select, and optimize test cases based on their likelihood of detecting defects, thereby reducing testing time and resource consumption[19]. This approach enhances test coverage, improves fault detection efficiency, and supports faster software delivery in modern development environments[20].

Machine Learning and optimization algorithms play a crucial role in automating test case generation and improving test suite effectiveness. Various UML diagrams serve as input models, while optimization techniques such as Genetic Algorithms, Particle Swarm Optimization, and Firefly Algorithms help generate high-quality test cases with improved coverage and reduced testing effort. Figure.2 illustrates the major techniques involved in ML-based test case generation and optimization.

DOI: 10.5281/zenodo.21342922

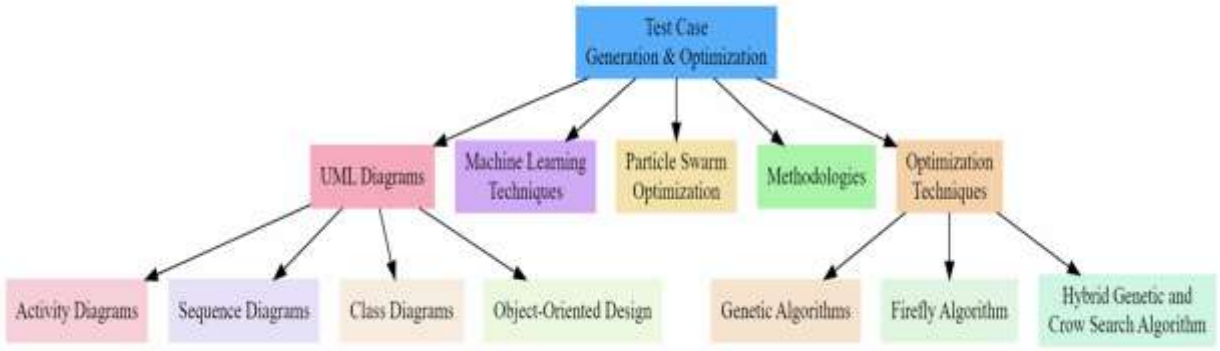


Fig. 2. The process of Test Case Generation and Optimization

### C. *Defect Prediction And Fault Detection Using Ml*

Software Defect Prediction (SDP) improves software quality and reduces testing costs by using machine learning models to predict defects from historical software data[21]. It supports efficient resource utilization by identifying fault-prone modules during the Software Development Life Cycle (SDLC). Machine learning techniques analyze software metrics, source code, execution logs, and defect history to detect potential faults at an early stage[22]. Common algorithms, including Random Forest, Support Vector Machine (SVM), Neural Networks, and K-Nearest Neighbour (KNN), help prioritize high-risk modules for testing. These approaches enhance testing efficiency, improve software reliability, reduce maintenance effort, and support the development of high-quality software systems[23]

## IV. Software Testing In Secure Modern Systems

Software security testing is a specialized form of testing aimed at validating and verifying that a software system meets its security requirements. Two principal approaches can be used: functional security testing and security vulnerability testing. While security functional testing is used to check the functionality, efficiency and availability of the designed and developed security functionalities and security system, security vulnerability (or penetration) testing directly addresses the identification and discovery of previously unknown system vulnerabilities introduced by security design flaws or software defects. Security vulnerability testing uses the simulation of attacks and other kinds of penetration attempts [24][25][26]. Therefore, testing methods including security testing, penetration testing and regression testing are widely adopted to identify flaws and ensure robust system performance. This process supports improved reliability, reduced risks, and maintained trust in security-critical applications

### A. *Security Testing Methodologies:*

Security testing is a multidimensional practice that employs a range of methodologies to identify vulnerabilities, assess risks, and enhance the resilience of software applications against potential cyber threats. software development teams to make well-informed decisions when designing their testing strategies, balancing trade-offs, and optimizing their testing efforts to deliver top-notch software products that meet user expectations and business objectives. This overview delves into some prominent security testing methodologies, each designed to uncover specific types of vulnerabilities and provide insights into application security:

- **Penetration Testing:** In the penetration testing technique, the tester attempts to find vulnerabilities in a system by mimicking the actual attacker's behavior to infect or damage the system.
- **Vulnerability Scanning and Static Code Analysis:** Vulnerability scanning uses automated tools to detect known security weaknesses in software components, while static code analysis examines source code to identify coding flaws such as SQL injection, cross-site scripting (XSS), and insecure authentication before execution.
- **Dynamic and Interactive Application Security Testing (DAST/IAST):** Dynamic testing evaluates application security during execution by analyzing runtime behavior and user interactions. IAST extends this approach by combining runtime monitoring with dynamic analysis to provide more accurate vulnerability detection.
- **Threat Modeling and Security Requirements Analysis:** Threat modeling identifies potential attack vectors during system design, while security requirements analysis ensures that security objectives and regulatory requirements are incorporated throughout the development process.
- **White Box Security Review and Regression Testing:** White box testing performs an in-depth examination of source code to identify architectural and implementation vulnerabilities. Security regression testing verifies that previously resolved vulnerabilities are not reintroduced after software updates.
- **Fuzz Testing:** Fuzz testing supplies unexpected or malformed inputs to an application to identify input validation errors, buffer overflows, and other vulnerabilities that may affect system reliability and security.

### B. *Exploring Emerging Trends in Security Testing:*

The dynamic landscape of security testing continues to evolve as technology advances and threat vectors become more sophisticated. The waterfall approach does not allow the process to go back to the previous phase and allow changes in it. The waterfall model is used for small projects, as there is little room for revisions once a stage is completed. This exploration delves into emerging trends that are reshaping the realm of security testing, focusing on the integration of security into DevOps practices, the adoption of continuous

DOI: 10.5281/zenodo.21342922

security testing, and the significance of threat intelligence sharing.

*1) DevSecOps Integration*

The convergence of development, operations, and security— commonly known as Dev SecOps—has gained traction as organizations recognize the necessity of integrating security throughout the entire software development lifecycle. By embedding security practices early and consistently, Dev SecOps aims to proactively identify vulnerabilities, reduce risk, and accelerate the delivery of secure applications. This trend emphasizes collaboration among cross-functional teams, encouraging security professionals to work closely with developers and operations personnel. This approach enables the detection and remediation of security issues at an accelerated pace, mitigating the impact of potential threats and aligning security efforts with the speed of modern development practices.

*2) Continuous Security Testing*

The traditional approach of conducting security testing as a one-time event is giving way to continuous security testing, which advocates for ongoing, automated assessment of application security. This trend aligns with the principles of continuous integration and continuous deployment (CI/CD), ensuring that security testing is seamlessly integrated into the development pipeline. As code changes are made, continuous security testing tools analyze the codebase, identify vulnerabilities, and provide real-time feedback to developers. This approach minimizes the window of exposure to potential threats, fosters a culture of security awareness, and enables swift remediation of issues.

*3) Threat Intelligence Sharing*

The growing complexity of cyber threats necessitates collaborative efforts among organizations to combat emerging risks effectively. Threat intelligence sharing involves the exchange of information about the latest threats, attack techniques, and vulnerabilities among different entities. This trend empowers organizations to prepare for potential threats, identify patterns across attacks and adopt proactive security measures. Collaboration and information sharing within the cybersecurity community enable collective resilience against a rapidly evolving threat landscape[27]

## V. Literature Review

This section reviews recent literature on machine learning approaches for software testing in secure modern systems. Table I summarizes the reviewed studies by comparing their focus areas, techniques, key contributions, limitations, and future research directions, highlighting existing trends and identifying important research gaps.

N.Varma et al, (2026) develops a literature-grounded research framework for data-driven software quality assurance by integrating machine learning-based risk prediction with intelligent test optimization. It proposes a multi-layer architecture that links feature engineering, risk scoring, test selection, feedback learning, and operational controls for modern quality engineering[28].

R. Madhumita and D. Chandana (2025), explores recent advancements in AI-driven software testing, highlighting AI-enhanced test automation, predictive defect detection, and LLM-based test generation. It also discusses the challenges and opportunities of adopting these technologies while ensuring quality in generative AI systems[29].

H. Chen and M. A. Babar (2024), considers security in machine learning-based software systems by addressing inherent defects and adversarial attacks throughout the software lifecycle. It highlights the lack of comprehensive review work covering this emerging area in software engineering[30].

J. Wang et al, (2024) provides a comprehensive review of LLMs in software testing, focusing on test case preparation and program repair. It also analyzes commonly used LLMs, prompt engineering techniques, and highlights key challenges and future opportunities[31].

S. Pargaonkar, (2023) explores automated security testing using dynamic analysis, static analysis, and IAST for vulnerability detection across the software development lifecycle. It also highlights Dev SecOps integration, continuous security testing, and proactive vulnerability management[32].

M. Abdul Salam et al, (2022) explores the use of machine learning in software engineering for tasks such as defect prediction, bug localization, code completion, and software testing. It highlights that machine learning improves testing efficiency and supports software quality assurance while reducing testing time and cost[33]

TABLE I. Summery Of The Study On Machine Learning Approaches For Software Testing In Secure Modern Systems

| Author | Focus Area | Techniques | Key Contribution | Limitations & Challenges | Future Work |
|---|---|---|---|---|---|
| N. Varma et al. (2026) | Data-driven software quality assurance | ML-based risk prediction, intelligent test optimization | Proposed a multi-layer framework integrating risk assessment and test optimization | Limited practical validation and real-world deployment | Validate the framework on large-scale industrial projects |
| R. Madhumita & D. Chandana (2025) | AI-driven software testing | AI-based automation, predictive analytics, LLMs | Reviewed AI-enhanced testing and generative AI quality assurance | Reliability, explainability, and adoption challenges | Develop trustworthy and explainable AI testing models |
| H. Chen & M. A. Babar (2024) | Security of ML-based software systems | Secure software engineering, adversarial defense | Reviewed security issues across the ML software lifecycle | Lack of comprehensive security frameworks | Design end-to-end secure ML testing methodologies |
| J. Wang et al. (2024) | LLM-based software testing | Large Language Models, prompt engineering | Surveyed LLM applications in test generation and program repair | Prompt dependency and evaluation challenges | Improve robustness and benchmarking of LLM-based testing |

| | | | | | |
|---|---|---|---|---|---|
| S. Pargaonkar (2023) | Automated security testing | Static analysis, dynamic analysis, IAST, DevSecOps | Presented integrated security testing across SDLC | Limited automation for emerging cyber threats | Integrate AI-driven continuous security testing |
| M. Abdul Salam et al. (2022) | Machine learning in software engineering | Defect prediction, bug localization, code completion | Demonstrated ML benefits for software quality assurance | Limited scalability and generalization across projects | Develop scalable and adaptive ML-based testing solutions |

## VI. Conclusion And Future Work

Software testing has become an indispensable component of modern software engineering, ensuring the quality, reliability, and security of increasingly complex software systems. This review examined recent machine learning approaches applied to software testing, including intelligent test case generation, defect prediction, fault detection, software quality assurance, and security testing. The reviewed studies demonstrate that machine learning significantly improves testing efficiency by automating testing activities, enhancing defect prediction accuracy, optimizing test execution, and strengthening vulnerability detection. Furthermore, emerging technologies such as Large Language Models and Dev SecOps have expanded the scope of intelligent and secure software testing. Overall, the survey highlights that machine learning is transforming software testing into a more adaptive, efficient, and security-aware process for modern software systems.

Future research should focus on developing explainable, robust, and scalable machine learning models for software testing. Integrating Large Language Models, reinforcement learning, and secure Dev SecOps frameworks with continuous testing can improve automation, vulnerability detection, and testing reliability. Standardized evaluation benchmarks and large-scale industrial validation remain essential for practical adoption.

**Funding:** This research received no external funding.
**Institutional Review Board Statement:** Not applicable.
**Informed Consent Statement:** Not applicable.
**Data Availability Statement:** The data supporting the findings of this study is available from the corresponding author upon reasonable request.
**Conflicts of Interest:** The author declares no conflict of interest.

**How to cite this article**

R. P. Singh, "A Review of Machine Learning Approaches for Software Testing in Secure Modern Systems," *Journal of Artificial Intelligence and Semiconductor Integrated Hardware*, vol. 1, no. 2, pp. 16–22, Apr.–Jun. 2026, doi: 10.5281/zenodo.21342922.

## References

[1] J. B. Mehta, "AI-Driven Test Engineering for Cloud-Native Systems," *Int. J. Data Sci. IoT Manag. Syst.*, vol. 5, no. 1, 2026, doi: 10.64751/ijdim.2026.v5i1.297.

[2] P. H. Desai, P. Mahalle, and P. Chandre, "Intelligent Access Control Schemes for the Internet of Everything: A Survey of Techniques, Challenges, and Future Directions," in *Information Systems for Intelligent Systems*, 2026, pp. 95–105. doi: 10.1007/978-3-032-13196-6_9.

[3] S. Ajorloo, A. Jamarani, M. Kashfi, M. Haghi Kashani, and A. Najafizadeh, "A systematic review of machine learning methods in software testing," *Appl. Soft Comput.*, vol. 162, p. 111805, Sep. 2024, doi: 10.1016/j.asoc.2024.111805.

[4] C. López-Martín, "Machine learning techniques for software testing effort prediction," *Softw. Qual. J.*, vol. 30, no. 1, pp. 65–100, February, 2022.

[5] A. Mehmood, Q. M. Ilyas, M. Ahmad, and Z. Shi, "Test Suite Optimization Using Machine Learning Techniques: A Comprehensive Study," *IEEE Access*, vol. 12, pp. 168645–168671, 2024, doi: 10.1109/ACCESS.2024.3490453.

[6] M. Islam, F. Khan, S. Alam, and M. Hasan, "Artificial Intelligence in Software Testing: A Systematic Review," in *TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)*, IEEE, Oct. 2023, pp. 524–529. doi: 10.1109/TENCON58879.2023.10322349.

[7] J. B. Mehta, "Securing Test Automation in Zero Trust Architectures: A Framework for Continuous Verification," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–5. doi: 10.1109/ICCA66035.2025.11430950.

[8] V. Chaturvedi, "A Review on AI-Driven Project Management for Transforming Software Engineering Perspectives," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 4, no. 2, pp. 895–908, Dec. 2024, doi: 10.48175/IJARSCT-22800E.

[9] X. Jia, "The Role and Importance of Software Testing in Software Quality Management," *J. Ind. Eng. Manag.*, vol. 1, no. 4, pp. 39–44, 2023, doi: 10.62517/jiem.202303406.

[10] P. Ammann and J. Offutt, "Introduction To Software Testing," *Introd. to Softw. Test.*, vol. 4, no. 3, pp. 1–345, 2016, doi: 10.1017/9781316771273.

[11] R. Palwe, "Three Layers of Trust in AI Interfaces: Interface, Behavior, and Organization," *Int. J. Sci. Res.*, vol. 15, no. 1, pp. 1152–1160, Jan. 2026, doi: 10.21275/SR26112072531.

[12] A. Anand and A. Uddin, "Importance of software testing in the process of software development," *Int. J. Sci. Res. Dev.*, vol. 12, no. 6, p. 1, 2019.

[13] S. K. Somarapu;, A. Srivastava;, A. B. Singhal;, and A. R. R. Bandaru, "Structural Dependency Analysis of IoT Adoption Drivers for Sustainable Business Operations Using ISM-MICMAC Analysis," in *2026 5th International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Noida, India: IEEE, 2026, pp. 1–6, February. doi: 10.1109/ICIPTM69057.2026.11465718.

[14] E. Kesavan, "The Future of Software Testing: A Review of Trends, Challenges, and Opportunities," *Int. J. Innov. Sci. Eng. Manag.*, vol. 4, no. 2, pp. 53–57, Apr. 2025, doi: 10.69968/ijisem.2025v4i253-57.

[15] Q. Zhang *et al.*, "A survey on large language models for software engineering," *Sci. China Inf. Sci.*, vol. 69, no. 4, p. 141102, 2026.

[16] N. Mulla and N. Jayakumar, "Role of Machine Learning \& Artificial Intelligence Techniques in Software Testing," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 6, pp. 2913–2921, 2021.

[17] K. K. Mohammed, "The Future is Cloud : Modernizing Big Data for the Cloud Era," *Int. J. Sci. Res. Eng. Trends*, vol. 11, no. 5, pp. 1–5, 2025, doi: 10.5281/zenodo.17339856.

[18] S. K. Sahu, A. Mokhade, and N. D. Bokde, "An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges," *Appl. Sci.*, vol. 13, no. 3, p. 1956, Feb. 2023, doi: 10.3390/app13031956.

[19] S. R. Kongarana, A. A. Rao, and P. R. Raju, "Review of Automated Test Case Generation, Optimization, and Prioritization using UML Diagrams: Trends, Limitations, and Future Directions," *Scalable Comput. Pract. Exp.*, vol. 25, no. 5, pp. 3651–3673, Aug. 2024, doi: 10.12694/scpe.v25i5.3030.

[20] V. Chaturvedi, "Modern Software Development with Java , Spring Boot , and Python : A Survey of Frameworks and Best Practices," *ESP J. Eng. Technol. Adv.*, vol. 3, no. 4, pp. 188–197, 2023, doi: 10.56472/25832646/JETA-V3I8P121.

[21] A. Khalid, G. Badshah, N. Ayub, M. Shiraz, and M. Ghouse, "Software Defect Prediction Analysis Using Machine Learning

Techniques," *Sustainability*, vol. 15, no. 6, Mar. 2023, doi: 10.3390/su15065517.

[22] S. K. Malaraju and S. K. Madishetty, "AI-Augmented Compiler Optimization for Energyefficient Software Execution on Embedded Systems," in *2025 14th International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, Uttar Pradesh, India: IEEE, 2025, pp. 1–6, November. doi: 10.1109/SMART66937.2025.11389313.

[23] P. Chandra Shekhar, "Driving Agile Excellence in Insurance Development through Shift-Left Testing," *Int. J. Multidiscip. Res.*, vol. 3, no. 6, pp. 1–18, 2021.

[24] O. Bin Tauqeer, S. Jan, A. Omar Khadidos, A. Omar Khadidos, F. Qudus Khan, and S. Khattak, "Analysis of Security Testing Techniques," *Intell. Autom. Soft Comput.*, vol. 29, no. 1, pp. 291–306, 2021, doi: 10.32604/iasc.2021.017260.

[25] S. A. D. Poovaiah, "EVALUATION OF LOGIC LOCKING SCHEMES AGAINST ORACLE-LESS MACHINE LEARNING ATTACKS AT THE RTL LEVEL," *Int. J. Artif. Intell. Mach. Learn.*, vol. 2, no. 1, pp. 273–296, Apr. 2023, doi: 10.34218/IJAIML_02_01_024.

[26] M. H. Hamza Afzal, "Securing AI Systems Against Adversarial Attacks: A Framework for Building Robust and Trustworthy Machine Learning Models," *Comput. Fraud Secur.*, no. 5, pp. 65–74, May 2024, doi: 10.52710/cfs.867.

[27] L. A. Yeruva, D. Singh, S. Suddala, N. Bhatt, and R. Uddin, "Augmented Data Management for Cache Performance, Cybersecurity, and Mobile Integration," *J. Comput. Mech. Manag.*, vol. 5, no. 3, pp. 280–293, May, Jun. 2026, doi: 10.57159/jcmm.5.3.26691.

[28] N. Varma, R. Chandra, S. Iyer, and P. Narayanan, "Data-Driven Software Quality Assurance: Leveraging Machine Learning for Risk Prediction and Test Optimization," *Int. J. Math. Anal. Res.*, vol. 2, no. 1, pp. 01–08, Jan. 2026, doi: 10.64137/3108-2637/IJMAR-V2I1P101.

[29] R. Madhumita and D. Chandana, "Advancing Software Testing: Integrating AI, Machine Learning, and Emerging Technologies," *Int. J. Res. Eng. Sci. Manag.*, vol. 8, no. 1, pp. 93–97, 2025.

[30] H. Chen and M. A. Babar, "Security for Machine Learning-based Software Systems: A Survey of Threats, Practices, and Challenges," *ACM Comput. Surv.*, vol. 56, no. 6, pp. 1–38, Jun. 2024, doi: 10.1145/3638531.

[31] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software Testing With Large Language Models: Survey, Landscape, and Vision," *IEEE Trans. Softw. Eng.*, vol. 50, no. 4, pp. 911–936, 2024, doi: 10.1109/TSE.2024.3368208.

[32] S. Pargaonkar, "Advancements in Security Testing: A Comprehensive Review of Methodologies and Emerging Trends in Software Quality Engineering," *Int. J. Sci. Res.*, vol. 12, no. 9, pp. 61–66, Sep. 2023, doi: 10.21275/SR23829090815.

[33] M. Abdul Salam, M. Abdel-Fattah, and A. Moemen, "A Survey on Software Testing Automation using Machine Learning Techniques," *Int. J. Comput. Appl.*, vol. 183, no. 51, pp. 12–19, 2022, doi: 10.5120/ijca2022921919.