International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor 2024: 7.101

AI - Based Root Cause Analysis of Test Failures Using Allure Reports

Alex Thomas Thomas

Saransh Inc
Email: *alexthomaslive[at]gmail.com*

Abstract: ***With increasingly complex and larger systems, automated testing is nowadays an embedded element of modern development practices. However, root cause analysis (RCA) of test failures remains a time - consuming, knowledge - intensive, and error - prone activity that typically requires considerable domain expertise. This paper explores the use of an AI - powered RCA on test failure data extracted from Allure reports a standard test framework to generate rich, structured test outcomes. The approach utilizes machine learning methods, natural language understanding, and causality to enable automated diagnosis and classification of failures through the analysis of logs, error messages, and execution traces. It not only accelerates fault localization but also improves the accuracy and uniformity of diagnosis. The review synthesizes current research trends and technology enabling intelligent RCA, reviews current frameworks and tools, and documents the incorporation of AI models into CI/CD pipelines. The paper also addresses significant issues such as interpretability of the model, data quality, and managing flaky tests. By bridging the gap between test result visualization and automated diagnosis, AI - based RCA with Allure presents a smart and scalable solution for improving software reliability and development efficiency.***

Keywords: Artificial Intelligence (AI), Root Cause Analysis (RCA), Software Testing, Allure Test Reports, Test Automation, Failure Diagnosis

1. Introduction

The expanding complexity and size of modern software systems have made automated testing a necessity to quality and dependability. DevOps pipelines and agile development increasingly rely on Continuous Integration and Continuous Delivery (CI/CD), where automated testing is a supporting pillar. However, although test automation successfully determines failure, determining the root cause of these failures remains a manual very much, time - consuming, and error - probable effort. This manual Root Cause Analysis (RCA) process slows down the delivery cycle, increases operational cost, and slows down development efficiency. Conventional RCA methods typically rely on human know - how and heuristic debugging, which are difficult to scale and may yield unreliable results. With the success of AI technology in industrial systems following Industry 4.0 [1], the latest research has begun investigating AI approaches to automating RCA, especially using ensemble learning and statistical models that have proven effective in identifying fault patterns in large - scale manufacturing contexts [2] [4]. These approaches are increasingly applicable to software systems, especially for minimal - intervention test failure diagnosis.

Allure is a widely used reporting tool that provides detailed test run visualizations, including logs, screenshots, and error traces. While powerful in terms of data presentation, it lacks built - in intelligence to reason or diagnose test failure. This provides a good opportunity to integrate Allure with AI - driven RCA methods so that it can automatically deduce patterns, group similar failures, and infer likely causes from test artifacts. It is now simpler to incorporate AI in Allure reports due to better explainable AI [3] [5], natural language processing (NLP) [11], and causal inference models like RADICE [7].

Software engineering research has also demonstrated the ability of causal graphs [6] [7], graph - based incident extraction [8], and microservice - oriented RCA frameworks [9] [10] to identify root causes for system failures. NLP - based log analysis and deep learning - based anomaly detection [12] [17] also make it possible for systems to process the structured output obtained by test reports such as Allure's. The significance of this study is its capacity to transform test failure diagnosis by integrating smart, AI - based RCA capability into existing tools like Allure. This not only minimizes the need for human intervention but also accelerates root cause discovery, improves release speed, and assists in building more resilient and autonomous testing environments. By comparing existing literature, techniques, and software, this study aims at reviewing the state - of - the - art of AI - facilitating RCA and determining the direction for its effective utilization in software testing procedures.

2. Problem Statement

Automated testing is central to software quality and reliability assurance in modern software development pipelines. Automated testing frameworks like Allure Test Reports are mainstream in their adoption because of their ability to aggregate and present test results in a pictorially intuitive and structured format. However, despite all their reporting capability, Allure lacks intelligent intrinsic mechanisms for diagnosing the root causes of test failure, which it offloads to manual effort. This kind of manual analysis is time - consuming, error - prone, and increasingly unscalable in complex, microservices - based, and continuous integration (CI) - driven setups [5], [6], [10]. As industries are heading towards Industry 4.0 and zero - defect manufacturing principles [1], [4], there is a need for intelligent, interpretable, and automated debugging approaches. Earlier work also demonstrated the potential of Artificial Intelligence (AI) and Machine Learning (ML) methodologies to identify patterns and anomalies for root cause analysis in manufacturing and software systems [2], [3], [13]. However, these methodologies are not generally integrated with existing test reporting frameworks such as Allure.

Moreover, causal reasoning research [7], graph - based incident extraction [8], and log and error message analysis using NLP [11] have been promising but not yet successfully applied to Allure - generated test artifacts. The lack of integration of structured test result visualization and intelligent RCA methods holds back proactive fault detection, incident resolution slowing, and system downtime increasing. Therefore, there is an urgent need to develop an AI - powered framework that can analyze Allure test failure reports to automatically infer failure root causes, give explainable insights to developers, and enhance debugging procedures for large - scale and dynamic software systems.

3. Overview of Test Failure Analysis

3.1 Causes and Types of Test Failures

Automated test environment failures are usually categorized into three broad types: flaky tests, environment problems, and code bugs. Flaky tests are deterministic in nature but display non - deterministic results—they pass at one time and fail at another without making any change to the code. Such behavior is usually due to timing dependencies, asynchronous calls, or resource sharing conflicts, and greatly hinders root cause identification [6], [10]. Environment - related failures are due to instability or misconfiguration in the test infrastructure, for example, unstable network environments, incompatibility of software dependencies, or improperly set up environments. They occur particularly in distributed systems and CI/CD pipelines, where the test environment is continuously provisioned and destroyed [1], [2], [10]. Lastly, code defects represent genuine faults in the application under test, for example, logical faults, buggy implementations, or integration faults. These are genuine quality issues that require developer intervention to resolve [3], [5].

3.2 Popular Methodologies for Diagnosing Test Failures

Traditional techniques of root cause analysis (RCA) rely on manual inspection of test logs, stack traces, and error messages—laborious and error - prone approaches. More recent methodologies, though, are beginning to automate and simplify this process. Statistical and ensemble techniques correlate signals from groups of flunked tests to make educated guesses about probable causes, a strategy commonly employed in manufacturing systems and software reliability engineering [2], [4]. Machine learning models, especially those developed based on explainable AI frameworks, can be trained on historical patterns of failure and produce intelligent predictions for future failures and offer developers interpretable explanations for their reasoning [3], [5], [13]. Graph - based and causal inference models are tackling the problem structure - wise by examining intercomponent dependencies and tracing how failure spreads through systems [7], [8], [9]. NLP also makes logs and error messages easy to automatically parse, extracting relevant failure markers from text diagnostics and reducing drudgery involved in manually interpreting text diagnostics [11]. Despite such progress, these features have not yet been systematically integrated in most reporting tools such as Allure that focus on visualization and less on automated RCA.

3.3 Impact of Unresolved Test Failures on CI/CD Pipelines and Product Quality

Unclosed test failures will negatively impact continuous integration and deployment pipelines if they are not closed. Among the most direct impacts is the slowing down of feedback loops, where developers take longer to find and fix errors, thus slowing down releases and slowing overall development pace [5], [6]. Flaky tests, on the other hand, produce false positives, leading to spurious alerts that cause "test fatigue" and lead to teams either ignoring important signals or wasting time chasing non - issues [6]. An accumulation of repeated test failures also erodes confidence in the automated test suite, sometimes even compelling teams to fall back again to slower and more expensive manual testing approaches [10]. Above all, a failure to correctly and in a timely manner detect defects jeopardizes release of faulty software, which then may result in system downtime, complaints by customers, and financial losses [1], [4]. In summary, the coupling of AI - driven RCA techniques with tools like Allure would significantly enhance the process of fixing test failures, allowing for faster resolution, improved quality of software, and maintaining the integrity of CI/CD pipelines.

4. Test Reporting Tools - Allure Reports

4.1 Allure Reports Features and Architecture

Allure Reports is a popular open - source tool designed to produce clear, comprehensive, and visually rich reports that summarize test execution results. It is a modular - architecture tool with three main components. Its data collection layer collects XML or JSON - based result files from test frameworks at runtime; these files contain detailed metadata, including test steps, parameters, attachments such as logs and screenshots, and test status. The processing engine then gathers and processes these raw test results, forming a structured model containing detailed test results, history, and trends. Finally, the presentation layer offers an interactive and responsive web - based UI that displays test suites, test cases, step - by - step execution flows, and detailed failure information, making debugging and navigation more straightforward [5], [6]. Additionally, the design of Allure supports extensibility and simple integration with other programming languages and testing frameworks, which enables homogeneous reporting in heterogeneous testing environments [3].

4.2 Mechanism of Test Result Collection and Presentation in Allure

Allure has a close integration with popular test automation frameworks such as JUnit, TestNG, PyTest, and Mocha through the possibility to hook into their life cycle events during test execution. While it runs the test, it collects fine - grained information like each test's execution outcome (pass, fail, skip, or broken), elaborate logs at the step and sub - step level, in addition to attachments like screenshots and error logs. It also records metadata like test parameters, execution environment, and historic trend performance. All this information is gathered and stored in a structured manner and published in the form of an HTML report that can be browsed. The report shows tests hierarchically by suites and classes, indicates failed tests with their corresponding stack traces and

error messages, and allows trend analysis by comparing the most recent results with the last [5], [6].

4.3 Advantages of Allure Reports for Test Failure Analysis

Allure Reports provide a comprehensive overview of test runs by offering step - by - step execution traces of tests along with comprehensive attachments, which collectively aid testers and developers in recognizing failure points immediately and accelerating root cause analysis [5]. It provides a clean and intuitive interface with minimal cognitive load, such that one can recognize issues at first glance without manually digging into raw logs or complex data [3], [6]. Moreover, Allure's trending of test results over time offers valuable history, making it less difficult to identify flaky tests and recurring environment - related problems. This kind of longitudinal visibility is especially handy when taking advantage of more sophisticated AI - based analysis methods [5]. Furthermore, the plug - and - play extensible design of Allure accommodates custom annotations and integrations, providing a flexible framework for incorporating AI - based failure diagnosis tools into its reporting ecosystem [3], [5].

4.4 Native Integration with Test Automation Frameworks of Allure Reports

Allure has native integration support with a variety of popular automation frameworks from a variety of programming languages, e. g., Java frameworks JUnit, TestNG, and Cucumber, as well as Python frameworks PyTest. Such broad compatibility allows Allure to generate consistent, platform - neutral reports regardless of the underlying test technology. Allure collects rich execution data by listening to test lifecycle events at minimal configuration overhead, making integration into heterogeneous test pipelines simple [5], [6]. More importantly, Allure's JSON and XML result files, being formally structured, can be consumed by AI - powered root cause analysis engines for extraction of significant failure patterns, and therefore, Allure is a great pick in next - generation CI/CD pipelines for explainable AI diagnostics [3], [5].

5. AI and Machine Learning Techniques in Root Cause Analysis (RCA)

5.1 Summary of AI/ML Methods Useful for RCA

Machine learning and artificial intelligence are now a necessity in automating Root Cause Analysis (RCA) of test failures, especially in big and complex systems [1] [2] [4] [5]. The primary AI/ML methods employed are classification methods such as decision trees, support vector machines, and neural networks, comparing failure types based on features extracted from test results or logs [4] [13]. K - means and DBSCAN cluster algorithms are used when label data is not present, clustering associated failures to detect new or infrequent root causes [4] [5]. Anomaly detection techniques, like Isolation Forest and autoencoders, identify unusual patterns in test behavior or performance metrics for issues that indicate underlying issues [4] [12]. Causal inference models based on Bayesian networks and models like RADICE help to determine cause - and - effect relationships between code changes, testing environments, and failure occurrences [7]. Additionally, graph - based learning employs dependency graphs to monitor cascading faults in distributed systems and microservices architecture, hence delivering a better fault propagation understanding [7] [8] [9].

5.2 Data Preprocessing and Feature Extraction from Test Reports

Before presenting the test data to machine learning algorithms for feeding, test data from Allure Reports and similar ones need to undergo proper preprocessing. Structured parsing of primary information such as test status, test case hierarchical structure, long execution steps, error messages, and supporting logs from Allure's XML or JSON reports is called initial parsing. Feature engineering then creates insightful inputs for ML models from such aspects as time to fail, frequency of failure, failed components or modules, stack trace tokens, and previous sequences of pass/fail outputs [5] [6]. Data normalization and cleaning are also required to avoid duplicate records, deduplicate redundant log lines, and efficiently encode categorical metadata [2] [4]. The structured nature of Allure's output greatly simplifies these preprocessing tasks, and hence it is an excellent source to provide input to ML models [5].

5.3 Natural Language Processing (NLP) Application in Log and Error Message Analysis

Logs and exception messages typically arrive as long and unstructured data streams and therefore are problematic to analyze. NLP techniques offer intelligent means of processing the data. Tokenization and embedding methods transform text messages into vector representations such as TF - IDF or BERT embeddings that enable similarity analysis or classification [11, 12]. Error template mining captures repeated patterns in error messages collected from multiple runs of the same test and facilitates constant failure characterization [6, 10]. Named Entity Recognition (NER) is employed to identify crucial error - specific words, like error codes, file names, and stack trace functions, which enable the identification of failure sources [11]. Topic modeling and clustering algorithms, for instance, Latent Dirichlet Allocation (LDA), cluster logs with similar topics to assist in root cause analysis [5] [11]. These NLP - based approaches are especially important in microservice and CI/CD setups in which logs could be the sole indicator of failure [8] [10].

5.4 Pattern Detection and Past Failure Analysis

Levying the Allure's ease to track past trends, AI systems can identify recurring patterns of failures and evolving signatures of faults. Time - series pattern mining examines a series of test results to find regressions or occasional (flaky) test behaviors that would otherwise be difficult to detect [6] [10]. Signature learning enables machine learning models to discern the "fingerprints" of known failure modes and to predict potential new or incipient problems accurately [5] [9]. Through root cause analysis tag learning, AI can develop root cause templates to indicate likely root causes for current failures [4] [5] [6]. These pattern recognition skills not only assist in the diagnosis of existing problems but also facilitate forecasting potential test failure and assist in the automation of preventative measures.

6. Applying AI to Test Failure Data from Allure Reports

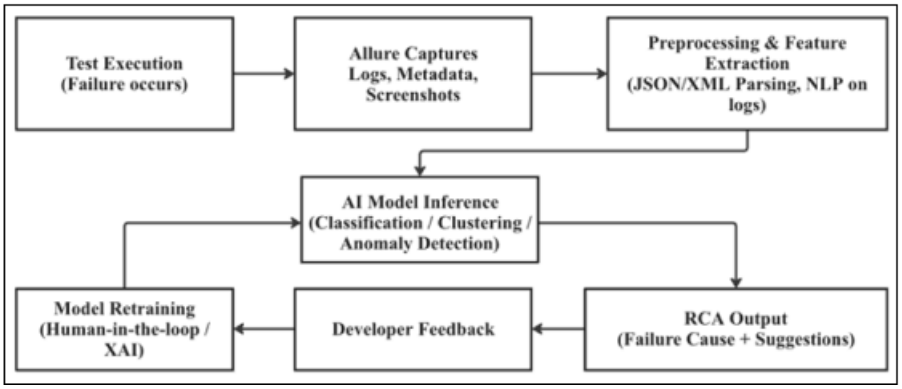

Figure 1: RCA Process flowchart

6.1 Structure of Allure Report Data

Allure Reports are largely employed in testing software for their ability to provide a visual representation and organization of test run data in an easy - to - view and complete format. An Allure report is composed of several structural components. Test steps are described in hierarchical form, keeping tabs on each action in a test case such as navigation commands or assertions. The report also includes status metadata, which captures the outcome of every test (passed, failed, broken, or skipped) along with accompanying timestamps. Logs and error messages are useful runtime information that capture standard output, stack traces, and exception messages. In UI automation, screenshots are commonly attached to failed test steps to capture the visual state of the interface at the moment the error occurred. Besides, attachments like HTTP responses, configuration files, or other artifacts may be included to add richness to the context. Historical data is also optionally displayed by Allure through plugins so that the testers can see trends over builds. The integration of structured data (such as XML/JSON) and semi - structured data (like logs and messages) makes Allure a prime choice for training AI models.

6.2 Methods to Extract and Prepare Data from Allure for AI Models

In order to utilize AI and machine learning algorithms on Allure data properly, raw test report information must be parsed and transformed into a consumable model format. This begins by parsing XML or JSON files found in the allure - results directory. These reports contain formatted test result data and can be extracted using command - line tools like allure - python - commons or by creating custom parsers with libraries like Pandas and Python's json module to display the data in tabular form. Feature engineering is then required after extraction to represent meaningful metadata like test suite names, environments, number of retries, and test durations. Logs can be processed using natural language processing (NLP) techniques like tokenization and vector embeddings like TF - IDF, Word2Vec, or BERT. Screenshots can be processed to extract image features using convolutional neural networks (CNNs) for UI testing. Labeling is then performed based on the learning paradigm: supervised learning can employ known failure types like timeouts, assertion failures, or UI element not found, whereas unsupervised learning can employ clustering of logs or test metadata. Data normalization and balancing are also necessary steps—numeric features such as duration or failure number are normalized, and techniques such as SMOTE or under - sampling may be employed to address class imbalance because test failures would be more typically less than passes. As noted in references [4], [5], and [6], input data preparation would typically be the most time - consuming but necessary phase of AI - root cause analysis.

6.3 Few Examples of AI Algorithms Used

Depending on the desired goal—classification, clustering, or anomaly detection different AI algorithms are utilized during root cause analysis of test failures. Decision trees and random forests are used when explainability is essential since they allow developers to trace down rational paths that result in test failures, and this aligns with explainable AI principles [13] [3]. Support Vector Machines (SVMs) are most effective in dealing with binary classification issues, such as determining whether a test is passing or failing based on structured metadata. Neural networks are applied to solve more complex patterns, especially sequential data or imagery data. Convolutional neural networks (CNNs) are most effectively applied in screenshot analysis, whereas Long Short - Term Memory (LSTM) networks are suited for pattern learning on sequential logs [14] [17]. Clustering methods such as K - means or DBSCAN help to partition similar types of failures together, normally uncovering underlying causes by analyzing logs and metadata [4] [5]. Finally, anomaly detection algorithms such as autoencoders and Isolation Forests detect abnormal test behaviors not conforming to expected patterns and mark them as possible issues even in the absence of labeled data [12]. Such AI models, being trained with well - structured and preprocessed data from Allure reports, offer rich capabilities for automated, scalable, and insightful root cause analysis.

7. Evaluation Metrics for AI - Based Root Cause Analysis (RCA)

Evaluation metrics for AI - based Root Cause Analysis (RCA) of test failures by Allure Reports are decisive factors for establishing the predictive power and actual - world usability of the system. Main performance measures such as accuracy, precision, recall, and F1 - score measure how accurately the

AI discriminates true root causes from formatted and unformatted test data, according to paradigms in [2], [4], and [5]. In addition, Mean Time to Resolution (MTTR) is a significant operational measure that reflects the speed with which failures are diagnosed and resolved, thereby improving CI/CD efficiency as described in [1] and [6]. Integrated human - in - the - loop feedback mechanisms and explainable AI (XAI) — as contended in [3] and [5]—enhance model transparency and responsiveness to allow engineers to authenticate, customize, and iteratively train models from real - world wisdom. These steps collectively ensure that AI - driven RCA not only operates precisely as intended in theory but also accelerates resolution and enables continuous quality improvement in complex test environments.

8. Challenges and Limitations

Artificial intelligence - based Root Cause Analysis (RCA) by means of Allure Reports is also faced with limitations and issues, which can affect its performance in real software testing contexts. One such limitation is the quality and completeness of information in Allure reports, which may rely on the way tests have been written and integrated with logging or monitoring systems, affecting the richness of input features for AI models as mentioned in [4] and [6]. In addition, noisy, redundant, or missing data especially in logs or step definitions can degrade accurate root cause prediction and necessitate robust preprocessing and anomaly handling techniques ([2], [5], [11]). Interpretability of the model is also a significant limitation because developers typically must understand AI decisions in order to trust and act upon them, and thus explainable AI (XAI) frameworks are essential ([3], [5]). Finally, scalability is a concern when applying these AI techniques to big test suites or distributed test environments, where efficiency in performance, memory management, and inference time should be optimized in a way that maintains responsiveness and relevance in CI/CD pipelines ([1], [7], [10]).

9. Conclusion

Application of Artificial Intelligence - based Root Cause Analysis (RCA) in test failures using Allure reports is one of the prime innovations in intelligent software quality assurance, aligning with the broader trend towards Industry 4.0 and intelligent automation advocated by Lee et al. [1]. Through the use of structured and semi - structured data such as test steps, logs, attachments, and screenshots from Allure, AI models can examine failure patterns suitably, detect defect origins, and assist in remediation early. Ensemble learning methods [2] through to causal discovery methods [7, 9] and natural language processing methods [11] have been promising in the diagnosis of system failure, especially when applied to the finesse needs of software testing. These models do not just enhance accuracy and reduce mean time to resolution (MTTR), as noted by Kathiresan [5], but also enhance explainability of fault diagnosis utilizing methods proposed in explainable AI frameworks [3]. Data interpretability and quality determine the efficiency of these systems. Challenges such as noisy or absent Allure information, difficulty scaling to large, distributed CI/CD pipelines, and the "black - box" nature of deep learning models are tangible limitations [4, 6, 10]. Despite these challenges, current research on causal inference, graph - based diagnostics [8], and human - in - the - loop approaches can provide solutions to intrinsic gaps. Together, AI and Allure - based RCA provide an active, data - driven testing regime, eliminating debugging activities and opening the door to more autonomous, fault - resilient, and scalable quality engineering systems in line with future trends in software and manufacturing applications [13, 15].

References

[1] J. Lee, H. Davari, J. Singh, and V. Pandhare, "Industrial Artificial Intelligence for Industry 4.0 - based manufacturing systems, " *Manufacturing Letters*, vol.18, pp.20–23, 2018.
[2] Z. Yang, J. Wang, and T. Chen, "An ensemble approach to root cause analysis in complex manufacturing systems, " *IEEE Trans. Reliab.*, vol.68, no.4, pp.1263–1279, 2019.
[3] C. Tantithamthavorn, J. Jiarpakdee, and J. Grundy, "Explainable AI for software engineering, " *arXiv preprint arXiv: 2012.01614*, 2020.
[4] J. Chen, C. Zhao, and Y. Sun, "A systematic review on machine learning methods for root cause analysis towards zero - defect manufacturing, " *Front. Manuf. Technol.*, vol.8, 2022.
[5] G. Kathiresan, "AI - Augmented Root Cause Analysis: Enhancing Debugging Efficiency in Large - Scale Software Systems, " *Well Testing J.*, vol.32, no.2, pp.130–145, 2023.
[6] H. Jiang, X. Li, Z. Yang, and J. Xuan, "What causes my test alarm? Automatic cause analysis for test alarms in system and integration testing, " *arXiv preprint arXiv: 1703.00768*, 2017.
[7] A. Tonon *et al.*, "RADICE: Causal graph - based root cause analysis for system performance diagnostics, " *arXiv preprint arXiv: 2501.11545*, 2025.
[8] Z. He *et al.*, "Graph - based incident extraction and diagnosis in large - scale online systems, " in *Proc.37th IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE)*, 2022, pp.1–13.
[9] A. Ikram *et al.*, "Root cause analysis of failures in microservices through causal discovery, " in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol.35, pp.31158–31170, 2022.
[10] J. Shi, S. Jiang, B. Xu, and Y. Xiao, "Failure diagnosis in microservice systems: A comprehensive survey and analysis, " *ACM Trans. Softw. Eng. Methodol.*, vol.32, no.4, pp.1–35, 2023.
[11] M. Nakata, Y. Watanabe, K. Fujiwara, and Y. Suwa, "Improving incident management process by natural language processing techniques, " *IEEE Access*, vol.8, pp.163984–163999, 2020.
[12] M. Vinayakumar *et al.*, "Deep learning approach for intelligent intrusion detection system, " *IEEE Access*, vol.7, pp.41525–41550, 2019.
[13] A. Chigurupati and N. Lassar, "Root cause analysis using artificial intelligence, " in *Proc. Annu. Rel. Maintainability Symp. (RAMS)*, 2017, pp.1–6.
[14] J. D. Crocco and J. R. P. R. O'Hern, "Manufacturing quality improvement through statistical root cause analysis using convolution neural networks, " *U. S. Patent* US20180293722A1, Oct.11, 2018.

[15] J. Penrose, "ARC Advisory Group: Proactive asset management with IIoT and analytics, " *ARC Advisory Group*, 2017.
[16] J. D. Crocco and J. R. P. R. O'Hern, "Manufacturing quality improvement through statistical root cause analysis using convolution neural networks, " *U. S. Patent* US20180293722A1, 2018.
[17] R. Vinayakumar *et al.*, "Deep learning approach for intelligent intrusion detection system, " *IEEE Access*, vol.7, pp.41525–41550, 2019.
[18] A. Chigurupati and N. Lassar, "Root cause analysis using artificial intelligence, " in *Proc.2017 Annu. Rel. Maintainability Symp. (RAMS)*, pp.1–6.
[19] J. D. Crocco and J. R. P. R. O'Hern, "Manufacturing quality improvement through statistical root cause analysis using convolution neural networks, " *U. S. Patent* US20180293722A1, 2018.
[20] J. Penrose, "ARC Advisory Group: Proactive asset management with IIoT and analytics, " *ARC Advisory Group*, 2017.

Author Profile

With 15+ years in software development, quality engineering, and agile leadership, I specialize in driving quality and efficiency across enterprise systems. As QE Lead, I lead cross - functional teams delivering robust solutions across AppleCare for Enterprise, Secure Messaging, and Digital Legacy. My expertise spans cards & payments, capital markets, and customer service platforms. I bring strong skills in SDLC/STLC, test automation, agile project management, and process optimization. I've led QA transformations at State Street and American Express, improving test efficiency by up to 70% and deploying automation frameworks that significantly reduce manual effort. Certified PMP and Scrum Master, I'm passionate about bridging business and tech, mentoring teams, and leveraging emerging tech like Generative AI to enhance innovation and delivery. **Core Skills**: Quality Engineering | Test Automation | Agile Leadership | Project Management | Financial Systems | Mainframe Testing | Generative AI in QA