INTERNATIONAL JOURNAL OF MODERN TECHNOLOGY AND ENGINEERING RESEARCH
Volume-3, Issue-1, (2025)
https://caarnjournals.com/index.php/ijmter/index
ISSN (Online): 2581-5792
A Peer Reviewed (Refereed) International Journal

ARTIFICIAL INTELLIGENCE-DRIVEN SOFTWARE EVALUATION: TECHNIQUES, CHALLENGES, AND FUTURE DEVELOPMENTS

[1] BENJAMIN, UBOKOBONG.E, [2] ISHARA BARHOSON GALADIMA.

[1] Department of Computer Science, Akwa Ibom State Polytechnic
[2,] Nuhu Bamalli Polytechnic, Zaria
Email: [1]; ubokobong.benjamin@akwaibompoly.edu.ng **[2];** Isharag873@gmail.com

ABSTRACT

This paper examines the impact of artificial intelligence (AI) on software evaluation, highlighting how AI-driven techniques have influenced the assessment and optimization of software systems. It discusses key methodologies, including machine learning algorithms for predictive analytics and automated testing frameworks. Additionally, it addresses challenges such as ensuring high-quality input data, reducing bias in AI models, and bridging skill gaps among developers. Furthermore, it considers future changes that could improve software development, such as generative models and real-time monitoring tools. By covering current uses and future possibilities, this study offers insights into how organizations can use these technologies to increase efficiency while handling ethical concerns.

Keywords*: Artificial Intelligence, AI-Driven, Evaluation, Software, Machine Learning, Algorithm*

1. Introduction

The integration of Artificial Intelligence (AI) into software evaluation has revolutionized the way developers assess and improve the efficiency and reliability of their applications. AI-driven tools leverage machine learning algorithms to analyze past performance data, predict future outcomes, and identify potential bugs before they reach production (Verocol, 2023). This transformative shift enables real-time monitoring, predictive analytics (Bird et al,2023), and automated testing processes, significantly enhancing the accuracy of performance evaluations while streamlining the development lifecycle (Verocol, 2023).In addition to its role in software development, AI has also made significant impacts in other sectors such as pharmaceuticals by accelerating drug discovery and development through advanced data analysis techniques (PMC,2023). Similarly, in software testing, AI automates repetitive tasks with remarkable speed and precision compared to manual methods. However, despite these advantages, challenges persist due to issues like data complexity and scalability (Davoudian et al,2020).

This paper aims to explore the techniques employed in AI-driven software evaluation along with its challenges. It will also delve into future trends that are likely to shape this

field further.

Software Evaluation: Software evaluation is a systematic process that assesses and compares software solutions to ensure they meet specific requirements. This process analyzes various aspects of software, such as functionality, reliability, usability, efficiency, maintainability, and portability.

Key stages in software evaluation include:

- **Defining objectives**: Identifying the specific needs that the software must address, including what the software will be used for and which teams will use it.
- **Researching options**: Exploring available solutions through online reviews, vendor websites, and peer recommendations to create a shortlist.
- **Assessing vendor reputation**: Checking vendor credibility through customer reviews and third-party analyst reports.
- **Using free trials/demos**: Testing software capabilities in real-world scenarios using free trials or demo versions.
- **Gathering user feedback**: Engaging end-users to evaluate usability and gather feedback during the selection process.
- **Comparing cost vs. value**: Comparing the software's price with its operational value while considering ROI, security, and compliance.

Criteria for software evaluation include functionality, reliability, usability, efficiency, maintainability, and portability, integrity, cost/benefit hardware/software requirements and the Vendor/Developer. These criteria ensure that selected software meets current needs and future scalability requirements.

It is important to involve multiple stakeholders during the evaluation process. These stakeholders include:

- IT leaders, who assess infrastructure compatibility
- CIOs (Chief Information Officers), who focus on strategic alignment
- Customer service leaders, who evaluate customer experience impacts
- ITSM (IT Service Managers), managers, who assess usability
- End-users, who provide practical perspectives on user experience.

The software evaluation process helps organizations avoid costly mistakes by aligning software capabilities with their unique needs.

- **Artificial Intelligence**: AI: Artificial Intelligence (AI) is a wide-ranging field focused on developing systems capable of performing tasks that typically require human intelligence. These tasks include reasoning, learning, problem-solving, perception, and understanding language. AI involves various technologies and techniques, such as machine learning, deep learning, and natural language processing, which enable computers and machines to mimic human-like intelligence, Forbes, 2018, IBM, 2024 and Britannica, 2024.
- **Machine Learning**: A branch of AI that involves training algorithms on data to make predictions or decisions without explicit programming for specific tasks.
- **Deep Learning**: A specialized form of machine learning that uses artificial

neural networks with multiple layers to process information, similar to the human brain's structure and function.

Techniques in AI-Driven Software Evaluation: AI employs various techniques to assess and enhance software quality, performance, and reliability. Key methods include:

1. **Automated Test Case Generation**: Automated Test Case Generation (ATCG) is an AI Algorithm that revolutionizes software testing by utilizing AI and machine learning algorithms to create test cases automatically. This process enhances efficiency, accuracy, and defect detection by analyzing software requirements and user interactions without manual efforts (Bu et al, 2021).

Key Features of ATCG

1. Requirement Analysis: AI interprets software specifications to identify potential test scenarios.

2. User Interaction Study: Simulates real-world usage patterns based on user behavior.

3. Historical Data Analysis**: Prioritizes tests using past data to focus on high-risk areas.

4. Adaptability: Continuously improves as the software evolves.

5. Comprehensive Coverage: Ensures thorough testing across various functionalities and environments.

Advantages of ATCG

1.Efficiency Boost: Reduces manual effort significantly.

2.Accuracy Improvement: Minimizes errors through automation.

3.Enhanced Coverage: Includes diverse scenarios often overlooked by humans.

4.Early Defect Detection: Identifies issues early in development for timely fixes.

5.By integrating AI-driven automation into traditional testing methods, ATCG transforms quality assurance processes, making them more efficient and effective than ever before.

2. Self-Healing Test Scripts. Self-healing test automation is an advanced approach that utilizes artificial intelligence (AI) and machine learning (ML) to enhance the resilience and efficiency of software testing. Traditional automated tests often depend on static identifiers to locate elements within an application's user interface (UI). When these identifiers change due to software updates or modifications, test scripts can fail, requiring manual maintenance to update the affected scripts. This maintenance can be time-consuming and may delay the development cycle.

In contrast, self-healing test automation systems are designed to automatically detect and adapt to changes in the application under test (Keysight Blog, 2022). When a self-healing system encounters a change, such as an updated button identifier that the test cannot recognize, it doesn't simply fail. Instead, it initiates alternative strategies to identify the element and continue the test execution. For instance, if an element's ID has changed, the system might use other attributes like name, CSS selector, XPath, or even the element's relative location to other elements to locate it. Once the correct element is identified, the system updates the test script with the new identifier, effectively "healing" itself to accommodate the change.

This self-healing capability significantly reduces the need for manual intervention, allowing test scripts to adapt dynamically to changes in the application's UI. As a result, the testing process becomes more robust and efficient, enabling teams to focus on developing new features rather than constantly updating test scripts. Moreover, by minimizing test failures due to minor UI changes, self-healing test automation ensures more reliable test outcomes and accelerates the overall development cycle. Several tools and frameworks have incorporated self-healing capabilities to enhance test automation. For example, integrating AI language models with Playwright can create a powerful self-healing automation framework. This integration enables the language models to analyze code within the context of browser interactions, leading to more accurate verification and eliminating issues such as hallucination. Additionally, AI-driven tools like Testsigma and testRigor offer self-healing mechanisms that assist testers in saving time and money, all the while accelerating the overall application performance. Self-healing test automation represents a significant advancement in software testing. By automatically adapting to changes in the application, it reduces maintenance efforts, ensures the robustness of the testing process, and allows development teams to deliver high-quality software more efficiently.

3. Performance Gap Analysis: Performance gap analysis is a crucial process in software development that involves identifying areas where a software application underperforms compared to its potential or desired state. AI frameworks play a significant role in this process by leveraging advanced technologies like machine learning (ML) and artificial intelligence (AI) to monitor performance metrics, detect gaps, and provide actionable recommendations for optimization (Novichkov et al, 2022).

Key Components of AI-Driven Performance Gap Analysis

1. Data Collection: Collect relevant performance metrics such as response time, throughput, resource utilization, etc.
2. Tools: Utilize performance monitoring tools and instrumentation to gather comprehensive data.
3. Benchmarking: Compare collected metrics against established benchmarks or historical data.
4. Statistical Techniques: Employ statistical hypothesis testing (e.g., t-tests), anomaly detection algorithms, and trend analysis to identify deviations.
5. Machine Learning Techniques: Use reinforcement learning or genetic algorithms to explore the configuration space of the software application.
6. Domain Knowledge Integration: Incorporate domain-specific knowledge and expert rules for contextually relevant recommendations.
7. Evaluation and Validation: Conduct experiments or simulations in real-world scenarios to measure the impact of generated recommendations on performance metrics.

By integrating these capabilities into their workflows, organizations can significantly enhance their ability to optimize software application performance, establish benchmarks or desired states against which current performance will be measured,

use statistical methods supported by ML/AI techniques to identify discrepancies between actual and desired states, generate actionable recommendations based on identified gaps using domain-specific knowledge where applicable and this structured approach ensures that organizations can systematically address underperformance issues while maximizing efficiency gains from leveraging advanced technologies like ML/AI in their operations.

Explainable AI (XAI) Techniques: In software analytics, XAI methods like PyExplainer and LIME are used to interpret machine learning model predictions, enhancing transparency and trust in AI-driven evaluations (Russo, 2024).

Ethical AI Considerations: Evaluating AI applications involves assessing data handling practices to ensure ethical standards, such as avoiding inherent biases and ensuring responsible data usage.

Evaluation-Driven Development: This approach integrates rigorous, data-driven testing with techniques like prompt engineering to improve AI models, ensuring they deliver real value in applications like AI-powered code analysis. These techniques collectively contribute to more efficient, reliable, and ethical software evaluation processes in AI-driven development.

Challenges in AI-Driven Software Evaluation

AI-driven software evaluation faces several challenges that impact its effectiveness and reliability. Here are some of the key issues:

1. Data Quality and Availability:

AI models require high-quality, extensive datasets for training. However, many organizations lack sufficient historical data, especially in new or niche projects, poor data quality leads to inaccurate predictions and unreliable defect detection (Athur Jozef et al, 2009).

2. Model Training:

Overfitting (performing well on training data but poorly on new data) and underfitting (failing to capture essential patterns) are common issues due to insufficient or imbalanced datasets (Magabaleh et al, 2024).

3. Lack of Domain Knowledge: AI models often lack contextual understanding of business logic or domain-specific requirements, leading to missed critical defects, this reduces trust in AI outputs when they fail to address key business concerns (Herald Papp, 2021).

4. Interpretability and Transparency: Many AI algorithms operate as "black boxes," making it difficult for developers to understand decision-making processes (Herald Papp et al, 2021).

This results in lack of transparency which can lead to mistrust and reluctance in adopting AI-driven solutions.

5. Integration Challenges: The absence of standardization complicates tool interoperability, requiring custom scripts or middleware for integration with existing systems (Durelli et al, 2019).

6. Bias and Fairness Concerns: Bias in algorithms can lead to unfair evaluations or decisions if not addressed through diverse training data sets.

7. Human Expertise Requirement: Despite advancements, human judgment remains crucial for effective testing due to the need for creativity, intuition, and adaptability that

current AI lacks.

Addressing these challenges requires strategic planning, investment in high-quality data management practices, development of explainable models, continuous testing validation processes, fostering a supportive environment within organizations, ensuring diverse training datasets and integrating robust oversight mechanisms.

Future Developments in AI-Driven Software Evaluation

1. **Automated Code Generation and Review**: Currently tools like OpenAI Codex and GitHub Copilot are transforming code generation by producing high-quality code snippets from natural language descriptions. However, future Development expects improvements in the reliability and ethical considerations of AI-generated code, addressing concerns about intellectual property rights (Daniel Rodriguez, 2024).
2. Enhanced Bug Detection and Debugging: Current Trend has it that Machine learning algorithms are being used to predict bugs early in the development cycle, reducing testing time but Future Development points to Integration of more sophisticated ML models to enhance predictive capabilities, ensuring higher software reliability (Durelli et al, 2009).
3. Advanced Testing with AI: At the moment, we have AI-powered testing instruments generate adaptive test cases, prioritizing critical tests to improve software quality.But we expect Further sophistication in identifying intricate issues that might be overlooked by human testers alone in the future (Zarour et al, 2024).
4. NLP for Requirement Management: NLP facilitates converting user inputs into actionable tasks, enhancing requirement clarity through inconsistency detection.
5. Integration with Emerging Technologies: Trends include integrating blockchain for secure data sharing (decentralized applications) and leveraging quantum computing for faster data processing. Despite these advancements, challenges such as ethical implications (bias), data integrity issues, and integration complexities need to be addressed through robust frameworks (Zarour et al, 2024).

Examining other Software Engineering Processes

It is important to examine other software engineering processes alongside evaluation which is the focus of this research to draw a distinction amongst them, the table below tries to clearly define software engineering processes like Verification, Validation, Testing and Evaluation as well as showing the other of performing them:

Table 1: Definitions amongst software engineering processes

PROCESS	DEFINITION	OBJECTIVES	FOCUS	WHEN	PERFORMED BY	TECHNIQUE	EXAMPLE
VERIFICATION	Ensuring that the software meets specified requirements	To ensure correctness in design and implementation	Consistency, completeness, correctness of product e.g. design and code	Early in the development cycle	Developers and Reviewers	Reviews ,walkthroughs	Checking of design, match coding standards and requirements
VALIDATION	This process seeks the assurance that the software meets user needs and intended use	To ensure the software meets business and users requirements	User expectations, real world scenarios and intended purpose	After verification and before Deployment	End-users, Business Analyst and Testers	User acceptance Testing(UAT) ,System Testing in Real World Condition	Confirming that the Login Function meets User expectations.
TESTING	This is the process of executing programs to find if there is any defect(s)	To identify bugs and defects in the product	Functional and non-functional Testing	During Development and before Release	Testers and QA Engineers	Unit Testing, Integration Testing, System Testing and Acceptance Testing	Running test cases to check if login function works correctly
EVALUATION	Assessing the software's Overall effectiveness, quality and impact	Assess software performance, usability, maintainability and business impact	Overall system quality, user satisfaction and business value	After Deployment or periodically during use	User, Stakeholder, Auditors, and Reviewers	Surveys, feedback analysis, performance monitoring and case studies	Measuring the effectiveness of the login system based on user's feedback and performance metrics

SOFTWARE ENGINEERING PROCESS ORDER:
Verification → Validation → Software Testing → Evaluation

Comparison of Software Evaluation Techniques

Software evaluation has to do with assessing the quality, performance, and reliability of software. Two outstanding approaches are used: AI-driven techniques and traditional methods.

Key features, advantages, and tools associated with AI-driven and traditional software testing techniques.

AI-Driven Techniques:

- Key features include advanced algorithms for automated testing, predictive analytics, and real-time feedback.
- Advantages include high levels of automation that increase productivity by focusing on crucial aspects and predictive analytics that can identify potential problems early in development.
- Tools include TensorFlow, PyTorch, and Katalon Studio.

Traditional Techniques:

- Key features include manual testing which relies on human judgment and static analysis tools that examine code without executing it.
- Advantages include human intuition, which is especially valuable for exploratory testing and static tools, which provide immediate feedback on code structure.
- Tools include manual testing frameworks and SonarQube.

Table 2: Comparison table for software evaluation techniques

Feature	AI-Driven Techniques	Traditional Techniques
Automation Level	High automation of repetitive tasks.	Low automation; relies heavily on manual effort.
Predictive Capabilities	Can predict potential defects using machine learning.	Limited predictive capabilities; relies on past experiences or static rules.
Speed & Efficiency	Faster development cycles due to real-time feedback and automated processes.	Slower due to manual intervention in most stages.

Source : Ayman Odeh et al. 2024

while AI-driven techniques offer efficiency and predictive power through automation and machine learning algorithms, traditional methods give control over decision-making processes that need human interpretation. This comparison shows the strengths of each approach: AI do well in speed and accuracy through automation but may lack the capability needed for some complex decisions or exploratory tests where human reasoning is crucial. Traditional methods ensure thoroughness but are slower and more prone to errors due to manual involvement. However, while both approaches have their merits, AI-driven techniques are increasingly preferred for their efficiency and predictive capabilities in modern software evaluation scenarios.

while AI-driven techniques offer efficiency and predictive power through automation and machine learning algorithms, traditional methods give control over decision-making processes that need human interpretation. This comparison shows the strengths of each approach: AI do well in speed and accuracy through automation but may lack the capability needed for some complex decisions or exploratory tests where human reasoning is crucial. Traditional methods ensure thoroughness but are slower and more prone to errors due to manual involvement. However, while both approaches have their merits, AI-driven techniques are increasingly preferred for their efficiency and predictive capabilities in modern software evaluation scenarios.

Conclusion

AI-driven software evaluation is transforming how we develop, test, and deploy software by automating processes like coding, debugging, testing while enhancing productivity. As technology advances further into areas like decentralized applications using blockchain or quantum computing's computational power—software engineering education must comply accordingly to ensure responsible use of these powerful tools while maintaining a balance between automation efficiency benefits versus human contributions needed across various stages within the software development process.

Recommendations

Based on current trends and future directions, here are some key recommendations for advancing AI-driven software evaluation:

1. Enhanced Test Coverage and Predictive Testing: Implement AI tools that can identify edge cases and generate test cases automatically, enhancing predictive capabilities to detect potential issues before they arise.
2. Automated Code Generation and Review: Use tools like GitHub Copilot for automated code generation while addressing ethical implications and reliability concerns.
3. Integration with Emerging Technologies (DevOps, Edge Computing) : Integrate AI into DevOps practices to enhance continuous testing within CI/CD pipelines, leveraging edge computing for localized testing scenarios
4. Ethical Considerations in AI Adoption: Develop robust ethical guidelines to mitigate bias in AI models and ensure responsible adoption across software engineering processes.
5. AI-Driven Personalization: Use AI to offer personalized user experiences by adapting softw

References

Arthur Jozsef, 2009. Evaluation of software product quality metrics

Ayman Odeh, Nada Odeh ,2024. Comparative review of AI Techniques for automated code generation in

software Development.

Bird ,2023. Taking flight with copilot

Bu L.,2021. Machine learning steered symbolic execution framework for complex software code

Daniel Rodriquez, 2024, Introducing Evaluation Driven Development; building AI assistants that deliver real Value

Davoudian,A.2020 . Big data systems:A software engineering perspective.

Durelli, 2019. Machine learning applied to software testing: A systematic mapping study

Durelli V.H,2019 . Machine learning applied to software testing: A systematic mapping study

Herald Papp, 2021. A Methodology for delivering Evaluation criteria for software solutions

Herald Papp,2012. A Methodology for deriving Evaluation Criteria for software Solutions

Keysight Blogs, 2022. 5truths about AI-Driven software testing

Magabaleh ,2024. Machine learning applied to software testing: A systematic mapping study

Novichkov,P.S., 2022. A framework for rigorous self-validated data modeling and integrative,reproducible data

analysis.

PMC, 2023. Artificial Intelligence indrug discovery and Development

Russo, D.,2024. Navigating the complexity of generative ai adoption in software engineering

Verocol,2023: The role of AI in enhancing software performance evaluation

Zarour ,2024. Enhancing DevOps Engineering Education Through System- Based Learning Approach

Zarour, Akour.M ,2021. Enhancing DevOps Engineering Education Through System-Based Learning