International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 04 | April - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Leveraging AI for Bug Detection and Resolution in Software Testing

Aarti Sanjay Gawai[1]

[1] *Dept. of Information Technology and Computer Science, D.G Ruparel College of Arts, Science and Commerce, Mumbai, Maharashtra, India*
aartigawai1611@gmail.com

Abstract - It primarily demonstrates the importance of combining artificial intelligence with the field of software testing in terms of decreased administrative burden and effective outcomes to raise the caliber of software products.

The biggest challenge in today's software development environment has unavoidably remained the need for efficiency and quality in the delivery of all software engineering artifacts. While the industry follows complex software engineering protocols, the market expects new releases nearly every day. To address evolving market demands, various Artificial Intelligence (AI) techniques have been developed and are now extensively applied within the modern software engineering industry. Our main goals are to identify factors that affect the performance of current software bug detection and determine the efficacy of AI techniques in enhancing those techniques. According to the evidence the software testing field has made use of artificial intelligence methods and tools to simplify the challenging tasks of software bug detection and bug fixing.

Key Words*: *Bug, AI-Powered, Detection, Resolution, Software Testing, Test Cases, test automation.

1.INTRODUCTION

We all understand the significance of software testing in delivering high-quality products. Identifying bugs early in the development process significantly reduces time, cost and frustration leading to smoother releases and more reliable software. Old testing methods are slow, manual and full of human errors. Think about spending hours reviewing through code only to discover that an important bug passed by unnoticed. That's where AI-powered bug detection and resolution steps in as a game-changer. With its capacity to learn, adapt and automate, AI is transforming the traditional approach to software testing and bug detection. As software systems grow in complexity and size, manual bug detection methods become progressively impractical. This problem is addressed by automated bug detection using Artificial Intelligence which makes use of algorithms that can scan enormous amounts of code for anomalies or possible bugs. Artificial Intelligence has had a significant impact on society across various fields, including computer vision, speech recognition, the Internet of Things, neuroscience, language understanding, and healthcare.

In software development, a "bug" refers to an Any inconsistency in an actual and expected result in the functioning of the software detected in the development environment or to behave in unintended ways. Bugs can arise from various sources, including syntax mistakes, logical errors or design flaws. Bugs can occur at any stage of the software development life cycle from design and coding to testing and deployment. Bug detection refers to the process of identifying and locating errors within a system or software application. This is a crucial phase in the software development life cycle, ensuring that the final product is functional, reliable and meets the specified requirements. AI testing involves using artificial intelligence to improve and simplify software testing. Its main goal is to assess the software's performance, efficiency and reliability by automating tasks like running tests, validating data and detecting errors. This paper highlights the challenges of traditional testing and how AI-powered testing tools for bug detection are transforming bug detection and resolution in software testing.

2. THE CHANLLAGES OF TRADITIONAL SOFTWARE TESTING

Several challenges are commonly faced in the field of software testing

2.1 Manual testing is repetitive and time-intensive.

Traditional software testing often feels tedious and time-consuming, progressing slowly and lacking excitement. Since manual testing involves repetitive tasks, it can quickly drain project resources and cause testers to lose focus, increasing the risk of human error.

2.2 Diversity in Testing Environments.

Emulators and simulators can be beneficial in the early stages of testing but they fail to accurately mimic real-world conditions and performance. True application issues often surface only when tested on actual devices, making it essential for teams to have access to device labs that support testing across various device, browser and operating system combinations.

2.3 Identifying the Right Automation Tool and Framework

With numerous automation tools available, selecting the right one that aligns with the project's technology stack, budget and testing needs can be daunting. A poor automation strategy can result in inefficiencies, increased costs and wasted development effort.

2.4 Choosing the Right Test Automation Approaches

Determining what to automate, how extensively to automate and in what order can be a complex task especially when working with large-scale applications. A strategic approach is essential to maximize efficiency, reduce maintenance overhead and ensure long-term test reliability.

2.5 Insufficient Testing

To ensure an application is robust and error-free, it must be thoroughly tested across various environments. This is especially crucial for complex applications, where every part of the code must go through rigorous regression testing a process that can be highly time-consuming.

3. The Role of AI in Enhancing Software Testing

The roles performed by AI in Testing

3.1 Driving More Innovative Testing with AI

AI is revolutionizing the way software testing is approached, bringing innovation and intelligence into every phase of the process. Instead of relying Only on static, rule-based testing, AI enables dynamic and adaptive testing strategies. It can analyze large volumes of data to detect anomalies generate smart test cases and even predict where bugs are most likely to occur based on Previous data. Machine learning algorithms can continuously learn from past test results to improve future testing cycles. With AI, testing becomes more proactive, insightful and aligned with real-world usage leading to faster releases, fewer bugs and a more reliable user experience.

3.2 Speed and Efficiency in Natural Language Processing

Leveraging NLP, AI can effectively process and analyze large volumes of bug reports and documentation for efficient bug detection. This means it can process and analyze large amounts of data written in human language, allowing it to make predictions much faster than a human could. Natural Language Processing (NLP) analyzes previous reports to identify recurring issues and bugs within the system.

3.3. More Accurate Test Case Generation

AI can significantly enhance the process of test case generation by ensuring higher accuracy and efficiency. Creating test cases manually can be both time-consuming and error-prone but AI algorithms can analyze application code, user behavior and past test results to automatically generate detailed and accurate test cases.AI can analyze historical bug data to identify high-risk areas in the application. By prioritizing test cases that are more likely to uncover critical bugs it ensures better resource allocation.

Fig -1: Referenced from TestGrid

4. AI-Based Bug Resolution

- AI- grounded tools like GitHub Copilot and Facebook's Getafix give automated debugging suggestions reducing inventors' workload. These tools dissect error logs, suggest law fixes and indeed induce patches. According to Stack Overflow inventors using AI- powered law sidekicks save up to 40% of their time on debugging and repetitious coding tasks.
- AI- driven tone- mending software can describe runtime issues and apply real- time fixes. For

example, Netflix's Chaos Monkey part of its Simian Army leverages AI to identify weak system points and automatically correct failures to maintain service durability. This visionary strategy minimizes time-out and helps maintain the adaptability of operations indeed when faced with unlooked-for failures.

- Refactoring is essential for maintaining clean and effective codebases. AI-powered tools like Sorcery and Refact.AI analyse code structures and propose improvements helping to reduce technical debt and enhance maintainability. By automating refactoring, AI helps inventors concentrate on writing new features while keeping the being codebase optimized and effective.
- Not all bugs are inversely critical. AI- powered error prioritization tools dissect bug inflexibility, frequence and impact to help development brigades concentrate on fixing the most critical issues first. Tools like Sentry and Bugsnag use AI to classify errors based on their impact on users and provide actionable insights for efficient debugging and resolution.
- AI- Powered Root Beget Analysis relating the root cause of a bug can frequently be time-consuming. AI- driven root cause analysis tools like Over Ops and Rook out help inventors snappily pinpoint the source of issues.

5. AI-Powered Bug Detection

- AI-driven tools like DeepCode and Codiga analyze codebases in real-time identifying syntax errors, logic flaws and security vulnerabilities. By leveraging
- machine learning, these tools can recognize coding patterns and flag potential issues before they escalate into major problems.
- AI models trained on vast amounts of historical bug data can predict potential defects even before they manifest in production. According to a study by Microsoft Research, AI-based bug prediction models can reduce defects in enterprise applications by as much as 30%.
- AI enhances test automation by generating and executing test cases dynamically based on code changes. Tools such as Testim and Applitools leverage AI to spot UI inconsistencies, regression errors and performance problems, helping achieve broader test coverage with less manual intervention.
- AI-powered anomaly detection tools scan vast amounts of log data to identify unusual patterns that may indicate bugs or system failures. Platforms like Splunk and Datadog use machine learning to analyze logs in real time providing early warnings for potential issues.
- Static code analysis tools powered by AI such as SonarQube and CodeQL help detect vulnerabilities and performance issues without executing the program. By testing code structure, dependencies and patterns, these tools can detect sections of code that may lead to bugs in the future.

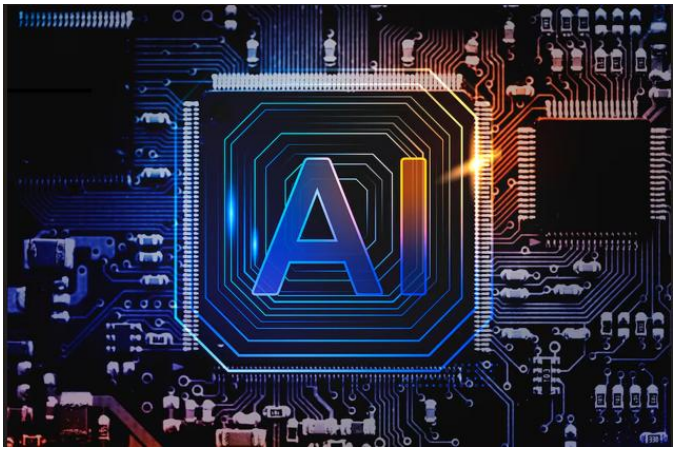

Fig -2: Referenced from ExpressComputer

6. AI Tools for Bug Detection and Resolution in Software Testing

6.1 DeepCode AI

DeepCode is an advanced AI-powered code review tool that uses cutting-edge machine learning models to understand code semantics. It provides real-time feedback to developers, identifying potential bugs, security vulnerabilities and performance issues.

6.2 Testim.io

Testim.io is an AI-powered testing platform designed to help teams efficiently create, run and maintain end-to-end tests, particularly for web applications. When working with single deviations in large datasets or when functionalities under test frequently change with each

release, its smart engine significantly optimizes the test-writing process.

6.3 Sentry.io

Sentry.io is an error tracking and performance monitoring tool that enables developers to monitor application issues and performance in real time. While not an AI tool, Sentry leverages powerful algorithms and integrations to enhance error detection and streamline debugging.

6.4 Testrigor

testRigor is generative AI Powered intelligent no-code test automation platform. Testim.io is designed to empower manual testers, business users and developers to automate testing across web, mobile, desktop API and database platforms all without writing any code. It significantly reduces test flakiness by automatically adapting to UI changes ensuring more stable and reliable test execution. Additionally, it integrates seamlessly with CI/CD workflows making it a powerful asset in modern DevOps pipelines. Users of testRigor have leveraged these advanced capabilities to dramatically increase the efficiency and effectiveness of their QA testing processes.

6.5 Applitools

Applitools represents the next generation of test automation platforms powered by advanced Visual AI technology.
This innovative platform dramatically reduces the time and effort involved in creating, executing and maintaining automated tests. By leveraging AI-driven visual validation, Applitools replaces traditional functional testing approaches with a smarter more efficient alternative.

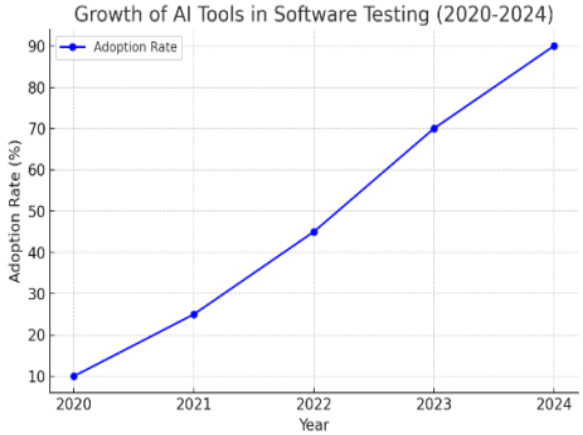

Fig -3: Growth of AI tools in software testing

7. Benefits of AI-Driven Bug Detection and Resolution

Faster Time-to-Market: AI-powered bug detection and resolution significantly reduce debugging time, enabling teams to release software updates faster. By automating time-consuming debugging tasks, development cycles become more efficient enabling businesses to stay ahead of the competition and respond swiftly to market demands.

Cost Savings: AI helps organizations cut costs by reducing the need for extensive manual testing and minimizing the time spent identifying and fixing bugs. According to a report by IBM, fixing a bug in production can cost four to five times more than resolving it during the development phase. AI-driven tools detect issues early, preventing costly production failures and lowering overall maintenance expenses.

Enhanced Software Quality: AI-driven tools detect issues with high precision, resulting in greater software reliability and user confidence. AI utilizes machine learning and historical data to proactively detect and resolve defects, preventing them from affecting the end-user experience. This results in fewer crashes, improved performance and an overall better user experience strengthening customer trust and satisfaction.

Improved Developer Productivity: Developers often spend a significant portion of their time debugging and resolving errors, which can slow down innovation and delivery. AI-powered tools alleviate this burden by automating repetitive debugging tasks, enabling developers to focus on writing high-quality code and building impactful features.

8. CONCLUSIONS

This paper presents an initial review of the approaches used in the integration of artificial intelligence and software testing. It investigates how the software testing domain has utilized artificial intelligence approaches and techniques to facilitate the complex tasks of software bug detection and bug prediction. It mainly demonstrates the significance of merging artificial intelligence with the software testing domain in terms of reduced overhead and efficient results to enhance the quality of software

products. We have also presented an analysis of various artificial intelligence tool that are utilized in the field of software bug detection and resolutions. Using AI in software testing offers numerous advantages including the ability to detect bugs faster, resolve issues more quickly and reduce the need for manual effort. Now is the perfect time for companies to embrace AI-powered solutions and gain a competitive edge in the fast-evolving tech landscape.

REFERENCES

1. https://practicallogix.com/ai-and-software-development-enhancing-bug-detection-and-resolution-efficiency/#:~:text=AI%20in%20Bug%20Resolution&text=For%20example%2C%20tools%20like%20CodeAI,common%20bugs%20and%20suggest%20fixes.
2. https://wizrlabs.ai/resources/ai-driven-bug-detection-and-resolution-transforming-software-maintenance

3. van Leeuwen, J. (ed.): Computer Science Today. Recent Trends and Developments. Lecture Notes in Computer Science, Vol. 1000. Springer-Verlag, Berlin Heidelberg New York (1995)

5. Develop-an-understanding-of-performance-planning-of-Organization.pdf

6. Paterson, A.L., and Ainsworth, R. (2017). Using machine learning for automated software testing. Proceedings of the ACM/IEEE International Conference on Automated Software Engineering

7. Pandey, S. K., Mishra, R. B., & Tripathi, A. K. (2017). BPDET: An effective software bug prediction model using deep representation and ensemble learning techniques. Expert Systems with Applications, 144, 113085.

8. Building a Culture of Quality: How AI Can Improve Collaboration in Bug Resolution, +1 626-217-2650 .

9. INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING, Omar Isam Al Mrayat, 2Malik Jawarneh, 3Dyala Ibrahim, 4Abdallah Altrad, International Journal of Intelligent Systems and Applications in Engineering, ISSN:2147-679921.

10. Advances in AI and Software Testing in 2024: A Comprehensive Review, Saivarun Pinna, IJIRT, 2349-6002.

11. K. K. Ganeeb, V. Jayaram, M. S. Krishnappa, P. Gupta, A. Nagpal, A. R. Banarse, and S. G. Aarella, "Advanced encryption techniques for securing data transfer in cloud computing: A comparative analysis of classical and quantum-resistant methods," International Journal of Computer Applications, vol. 186, no. 48, pp. 1–9, Nov. 2024. doi: 10.5120/ijca2024924135.

12. Ramification Of AI-powered Vibe Coding in Agile Development,
Ms. Mansi M.Rajapurkar1, Ms. Pranjali D. Chaudhari2, International Research Journal of Engineering and Technology (IRJET), p-ISSN: 2395-0072.

13. Holtz, A., & Seitz, R. (2020). AI-Assisted Software Development: A Survey of Tools and Methods. Springer. The role of AI in developer experience: Balancing automation and creativity. International Journal of Software Research, 28(3), 198-215. The Role of Artificial Intelligence in Agile Development: Challenges and Opportunities. Springer

14. https://mverve.com/ai-for-bug-detection-and-resolution-in-software-testing/

15.https://www.practicallogix.com/ai-and-software-development-enhancing-bug-detection-and-resolution-efficiency/

16.Artificial Intelligence for Automated Testing and Quality Assurance in Software Development Lifecycles, Azeezat Raheem1* , Ikeoluwa Kolawole 2 , Adeola Mercy Osilaja 3 and Victor Eyo Essien4, International Journal of Research Publication and Reviews, ISSN2582-7421.

17.AIAutomation and Testing, https://www.browserstack.com/guide/artificial-intelligence-in-test-automation

18. Aleti, A. (2023). Software Testing of Generative AI Systems: Challenges and Opportunities. arXiv preprint arXiv:2309.03554.

19. Software Testing A Craftsman's Approach, Paul C. Jorgensen,
978-1-4665-6069-7.

20. Software Testing and Analysis: Process, Principles, and Techniques, Mauro Pezze` Universita di Milano Bicocca ` Michal Young University of Oregon, ISBN-13 978-0-471-45593-6.

BIOGRAPHIES

Ms. Aarti Sanjay Gawai
M.Sc. Information Technology
Asst. Prof., IT/CS Dept. D G Ruparel College