AI models in software performance testing

Artur Dovbysh, Andrii Holovko, Vladyslav Holdovanskyi, Vladyslav Alieksiev
Lviv Polytechnic National University, Ukraine
artur.v.dovbysh@lpnu.ua,
andrii.i.holovko@lpnu.ua, https://orcid.org/0009-0001-2380-5050
vladyslav.a.holdovanskyi@lpnu.ua,
vladyslav.i.alieksiev@lpnu.ua https://orcid.org/0000-0003-0712-0120

Abstract: AI models are reshaping software performance testing. Machine Learning and Deep Learning algorithms automate test scenario generation, enable real-time monitoring, and predict performance issues. They facilitate dynamic load balancing, anomaly detection, testing automation, and offer performance optimization recommendations.

Keywords: AI, SOFTWARE TESTING, SECURITY TESTING, OWASP.

1. Introduction

In the dynamic landscape of software development, the integration of Artificial Intelligence (AI) has emerged as a transformative force, reshaping the conventional paradigms of software testing. With the exponential growth in software complexity, the demand for efficient, reliable, and rapid testing methodologies has become paramount. AI-driven approaches present a promising avenue to address these challenges, offering innovative solutions that redefine the very fabric of software testing.

The integration of Artificial Intelligence (AI) into the domains of performance and security testing stands as a defining frontier in fortifying software applications against vulnerabilities and performance bottlenecks. In the contemporary software landscape, where user expectations for both seamless experiences and robust security measures are non-negotiable, the convergence of AI-powered methodologies with performance and security testing offers a promising avenue to meet these demanding standards [1].

The primary goals of software testing are to ensure reliability, functionality, usability, security, and performance. Testing helps in identifying defects early in the development cycle, which reduces the cost of fixing issues and enhances the overall quality of the software product.

We will provide a detailed case studies overview of some modern AI-based tools for performance and security testing such as Testim, Chat GPT and Sonar. Sonar is not something fully dedicated to the testing process; however, it is able to predict some key code vulnerabilities after just a preliminary code scan.

Through an in-depth exploration of AI's integration into performance and security testing, this article aims to shed light on the current landscape, the challenges to overcome, and the promising avenues that AI-driven methodologies present, thereby contributing to the discourse on advancing software reliability in an era shaped by artificial intelligence.

2. Terminology Prerequisites.

2.1. Software testing.

Let’s start from explanation what is software testing in a nutshell.

Software testing is a crucial process in the development of software that involves evaluating a system or application to identify any discrepancies between expected and actual results. It's done to ensure that the software meets specific requirements, works as intended, and is free from bugs or errors before it's released to users.

Testing involves various methods and techniques; however, we are interested the most in the following types:

- Performance Testing: This type of testing focuses on evaluating how a system performs under specific conditions. It includes assessing the speed, responsiveness, stability, and scalability of the software under various workloads. Performance testing techniques like load testing, stress testing, and scalability testing help identify bottlenecks, measure response times, and ensure the system can handle expected levels of user activity without performance degradation.

- UI (User Interface) Testing: UI testing is centered around examining the graphical user interface of the software. It ensures that the user interface elements like buttons, menus, screens, and interactions work as intended. Testers check for consistency in design, proper alignment of elements, responsiveness to user inputs, and adherence to usability principles. The goal is to provide users with an intuitive, visually appealing, and functional interface.

- Security Testing: Security testing aims to uncover vulnerabilities, weaknesses, or threats in the software that could compromise its integrity, confidentiality, or availability. Testers simulate potential attacks or breaches to assess the robustness of the system's defenses. This involves techniques such as penetration testing, vulnerability scanning, and security audits to identify and address security risks, ensuring the software remains resilient against potential threats [6].

I believe it is crucial to have security testing included in your SDLC, due to the high risks of data compromising. According to the recent investigation by OWASP group the Top 3 compromised data types in 2022 are [7]:

Top 3 Most Compromised Data Types in 2022

![Fig. 1 Top 3 most compromised data types in 2022.](image)

2.2. Artificial Intelligence and Machine Learning.

AI is the overarching field that encompasses the development of computer systems capable of performing tasks that typically require human intelligence. This field aims to create intelligent machines that can simulate human cognitive functions such as learning, problem-solving, reasoning, perception, and language understanding. AI techniques can broadly be categorized into two types:
This type of AI is designed to algorithms and models that allow machines to learn from data and improve their performance on a task over time without being explicitly programmed. It’s based on the idea that systems can learn from data, identify patterns, make decisions, and improve their performance with experience\cite{1}.

Machine Learning (ML) is a subset of AI that focuses on developing algorithms and models that allow machines to learn from data and improve their performance on a task over time without being explicitly programmed. It’s based on the idea that systems can learn from data, identify patterns, make decisions, and improve their performance with experience\cite{1}.

3. Application of performance and security testing in nowadays development process.

Today, it’s hard to imagine the software development process without testing. Testing directly influences the quality of the created product, especially when it comes to security, which is crucial in our time. The problem of cyber security is very important now, because according to OWASP statistics, more and more new types of attacks and vulnerabilities are appearing.

![Fig. 2 OWASP top 10 changes dynamic.](image)

Testing is a rather expensive process in terms of resources and time. Hence, the relevance of integrating AI into the software testing process becomes indisputable. AI has undeniable advantages: it doesn't get tired, lacks the 'human factor,' and operates quickly, among other things\cite{8}.

For quite some time, programmers have been researching and developing AI models to aid in testing. Currently, there are ready-made tools built on this concept, such as:

- **Testim**
- **Chat GPT**
- **Sonar**

Briefly, the use of AI in SDLC would look like this:

![Fig. 3 SDLC with AI based testing.](image)

As we can see, for the end user (in this case, the development team), nothing changes. It’s just that a significant part of the testing process is transferred to AI to simplify and speed up the testing process\cite{1-2}.

Therefore, let’s delve into some of the existing tools in more detail.


Examining case studies is a valuable way to gain insights into the efficacy of various tools in enhancing software performance and security testing. ChatGPT, Testim, and Sonar stand as significant players, each offering unique contributions to this critical process. Through detailed case studies, we can explore how these tools have revolutionized software testing methodologies, improved efficiency, and elevated the overall quality of software products. Their innovative approaches and distinct functionalities have reshaped the landscape of performance testing, showcasing the power of technology in optimizing software development workflows.

4.1. ChatGPT.

General information:

- **Owner:** OpenAI.
- **Released in:** GPT-3 model released in June 2020, updated model – GPT-4 in January 2022.
- **Key features:** Natural language understanding, text generation, context comprehension, content summarization, language translation, and more.
- **Software testing areas:** Unit, Functional, Regression, Usability, Compatibility, Performance, Security, Integration, API, Acceptance.

**Advantages:**

- Efficiency in Test Scenario Generation: ChatGPT can swiftly generate diverse test scenarios. For performance testing, it can simulate various user interactions and loads, aiding in assessing how the software handles different usage patterns. For security testing, it can create scenarios to check for vulnerabilities, helping identify potential entry points for attacks.
- Rapid Prototyping for Testing Environments: ChatGPT can assist in quickly creating prototypes or test environments for performance and security testing. It can generate test data, simulate interactions, or create scenarios that might stress or reveal vulnerabilities in the software.
- Automated Test Case Generation: ChatGPT can help in generating automated test cases based on specific parameters or requirements. It can produce a wide range of test cases to cover various functionalities, aiding in comprehensive testing.
- Scalability Testing Support: With its ability to generate diverse loads and scenarios, ChatGPT can assist in scalability testing, determining how the software performs with varying levels of users or data, helping identify potential bottlenecks or performance issues under different scales.
- Identifying Language-Based Security Vulnerabilities: ChatGPT can assist in identifying language-based security vulnerabilities by generating queries or inputs that might trigger unexpected behaviour in the software. This helps in fortifying against potential linguistic exploitation or manipulation-based security breaches.
- Continuous Testing and Validation: ChatGPT’s ability to continuously generate new test scenarios can aid in...
ongoing testing efforts. This is particularly beneficial in identifying potential performance or security issues as the software evolves or new features are added.

Disadvantages:
- Limited Technical Expertise: ChatGPT lacks deep technical understanding, which might hinder its ability to uncover intricate software issues.
- Inability to Mimic Real User Behavior Fully: It may not replicate all complexities of real user behaviors, potentially missing certain edge cases.
- Risk of Incomplete Coverage: It might not cover all possible scenarios or vulnerabilities, leaving blind spots.
- Handling Complex System Interactions: Testing intricate systems might challenge ChatGPT’s comprehension and simulation abilities.
- Dependency on Training Data and Bias: Biases in training data could influence generated test cases, overlooking certain vulnerabilities.
- Resource Intensiveness: Generating numerous test scenarios might be time and resource-intensive for larger systems [3].

4.2. Testim.

General information:
- Owner: Testim.io Ltd.
- Released in: was first introduced in 2015, aiming to revolutionize test automation by leveraging machine learning and AI capabilities.
- Key features: AI-Powered Test Authoring, Self-Maintenance, Parallel Execution, Integrations.
- Software testing areas: Functional, Regression, Cross-Browser and Cross-Platform testing, UI/UX Testing, Security testing.

Advantages:
- Faster Test Creation: Its AI-powered test authoring capabilities enable faster test creation by learning from user interactions, reducing the need for extensive scripting or manual input.
- Reduced Maintenance Efforts: The self-maintenance feature allows tests to adapt to changes in the UI automatically. When UI elements change, Testim updates the tests accordingly, minimizing the effort needed for maintenance.
- Enhanced Test Coverage: With parallel execution capabilities, Testim allows tests to run simultaneously across various browsers, devices, or environments, thereby increasing test coverage and efficiency.
- Improved Reusability: Testim encourages the creation of reusable test components, making it easier to maintain a modular testing structure and promote consistency across tests.
- Integrations and Ecosystem Support: It integrates well with various CI/CD tools, issue trackers, and test management platforms, fitting into existing development ecosystems seamlessly.
- AI-Driven Insights: Testim provides insights and analytics based on test results, offering data-driven feedback on test performance and potential areas for improvement.

Disadvantages:
- User-Friendly Interface: Its user interface is designed to be intuitive, making it accessible for testers with varying levels of technical expertise.
- Learning Curve: Requires time to master its AI-driven features.
- Dependency on UI: Changes in UI elements can impact test maintenance.
- Complex Scenarios: Handling intricate test cases might be challenging.
- AI Accuracy: Potential inaccuracies in AI interpretation.
- Cost: Expense could be a concern for some teams.
- Limited Non-UI Testing: Focus primarily on UI testing.
- Tool Ecosystem Lock-in: Dependency on Testim's ecosystem [4].

4.3. Sonar.

General information:
- Owner: SonarSource SA.
- Released in: The initial release was in 2007.
- Software testing areas: Sonar primarily focuses on static code analysis and code quality inspection. It helps in identifying issues and potential bugs in the codebase without executing the code.

Advantages:
- Code Quality Improvement: It helps improve code quality by identifying bugs, vulnerabilities, and code smells early in the development process.
- Customizable Rulesets: Users can tailor the rules to fit their specific project needs, ensuring that the analysis focuses on the most relevant issues.
- Support for Multiple Languages: It supports various programming languages, allowing teams working with diverse tech stacks to benefit from its analysis.
- Integration and Automation: Integrates seamlessly with CI/CD pipelines and development environments, automating code analysis as part of the development workflow.
- Visualizations and Reports: Provides comprehensive reports and visual representations of code quality metrics, making it easier for teams to track improvements or identify areas for enhancement.
- Security Vulnerability Detection: It identifies security vulnerabilities, aiding in the creation of more secure software.
- Community and Support: Being an open-source tool, it has a vibrant community that contributes to its improvement, and it benefits from ongoing updates and support.
- Ease of Use: SonarQube offers an intuitive interface that simplifies the process of identifying and understanding code issues.
Disadvantages:
- Resource Intensive: Analyzing large projects can strain system resources.
- Customization Complexity: Customizing rules may be complex.
- False Positives/Negatives: It can produce false results, requiring manual review.
- Limited Dynamic Analysis: Primarily focuses on static code analysis.
- Learning Curve: New users might face a learning curve.
- Open Source vs. Enterprise Features: Advanced features may require the paid version.
- Tool Overload: Managing multiple tools can be complex.
- Continuous Configuration Updates: Requires regular updates and maintenance for coding standards [5].

5. Prospects and challenges.

I believe that the use of such tools based on AI models has a high potential for further application and research in the field of software testing. Let’s consider the following prospects and challenges of it:

- Predictive Analysis: AI predicts performance issues using historical data.
- Automated Test Generation: AI creates diverse test scenarios automatically.
- Real-time Monitoring: AI detects anomalies and performance issues instantly.
- Root Cause Analysis: AI swiftly identifies performance bottlenecks.
- Resource Optimization: AI dynamically adjusts system resources.
- Vulnerability Detection: AI scans code to find potential vulnerabilities.
- Behavioral Analysis: AI monitors and flags abnormal system behavior.
- Automated Penetration Testing: AI simulates cyberattacks to find weaknesses.
- Threat Intelligence: AI gathers and analyzes security threat data.
- Security Patching: AI assists in applying security patches promptly.
- Machine Learning Optimization: ML optimizes test suites based on outcomes.
- AI-Driven Test Maintenance: AI updates test scripts as software evolves.
- Anomaly Detection: AI identifies anomalies in testing, aiding security.

So, as far as you can see, there are plenty of possible improvements and research directions.

6. Conclusions.

In conclusion, the integration of AI models into software performance testing represents a transformative shift in testing paradigms. These AI-driven approaches offer precision in identifying bottlenecks, predicting system behaviors, and optimizing resource usage. This overview underscores the pivotal role of AI in enhancing testing accuracy and scalability, leading to accelerated development cycles and improved software quality.

This exploration highlights the predictive capabilities of AI, enabling proactive issue identification and risk mitigation, ultimately ensuring greater software stability and user satisfaction. As the industry evolves, embracing AI-powered testing frameworks becomes crucial for competitiveness and efficiency in software development.

In summary, AI models in software performance testing usher in an era of innovation, where intelligent systems elevate testing precision and adaptability. Continued research and implementation promise to reshape software quality assurance, offering unmatched efficiency and accuracy in meeting the demands of the digital age.

7. References