AI Applied to Testing Planning

Alexia Jennings, Jenan Qasem, Sanskriti Gyawali, Diwashna Poudel, Arushi Chaturvedi

April 28, 2025

Table of contents

Introduction

Making a testing plan is everything in the software life cycle. It ensures the accuracy of the software before handing it over to users. While planning, a developer has to consider the testing parts to be checked, the methods of testing, and the anticipated outcomes. Effective planning for tests increases efficiency, identifies many problems early, and strengthens the software's reliability and quality. Lack of proper procedure causes a test to be unstructured and critical test issues can be overlooked, which is bad for the future.

Several important ideas deal with test planning like test cases, test coverage, and testing strategies. These steps include putting in some action or doing something you intend to perform in the software with a view to verifying whether it responds to you correctly. Every test case comes with an output as well as an input. This deals with the amount of the software checks that you subject him to in terms of tests and the objective is to manage as many vital areas as possible. An efficient test plan is one that finds the balance between time allocated versus the testing to be done. Diverse approaches to testing such as unit tests which test smaller portions of code or system tests which evaluate the whole system, help structure work and ensure all aspects are covered. Detecting issues at the earliest possible stage can be achieved by planning in advance which helps streamline the entire testing procedure.

State-of-the-Art Methods

Test.ai is an AI-powered automated test tool primarily designed for web and mobile applications. Its most striking feature is that it learns UI objects itself and then will re-adapt itself to changes in the application layout using machine learning. This essentially reduces human test scripting. It will continue to learn over time as it gains experience. Test.ai is particularly well-suited for teams that want to automate tests with very limited human interaction. However, it does have

some drawbacks. The steep learning curve, particularly with tuning the AI models to the specifics of the application, is not beginner friendly. Additionally, its mobile and web app focus limits its potential use within more complex or legacy environments. The cost is also unreasonable, particularly for smaller projects or teams.

Appvance IQ is very efficient for automated test creation and concurrent test execution, making it highly appropriate for high-volume, enterprise-type applications. It will provide testing for web, mobile, and APIs as well as execute concurrent tests, across different devices and platforms. This will often considerably shorten the time required to do the test. It is adequately scalable and efficient at managing complexity. It may work for organizations like HHS, which will be executing it in a complex system. Conversely, Appvance IQ is a very resource-heavy tool and needs a lot of computing power, particularly for running parallel test executions. Installation is also resource-heavy and requires significant expertise, not only in the tool but also in the subject application under test. In addition, even when it's possible to run with automated reports, the automation may not provide the detail necessary for proper diagnostics.

Functionize is unique from the rest because it leverages natural language processing (NLP), whereby testers can create tests in natural language (plain English). This makes it very user-friendly for non-technical users, it is possible to create tests very easily and without any programming knowledge. Functionize excels at intelligent test maintenance, whereby tests are dynamically automatically updated as the application changes, thus minimizing the human intervention required to keep the test suites fresh. However, Functionize is not without its flaws. The initial training of the AI may take time and if you use poor-quality training data, you can also receive incorrect test results. Additionally, although Functionize is cloud-based and relatively user friendly the platform may slow down with more complex or larger test cases. Similarly, the ease of use of the tool may limit customization for teams with unique testing needs.

From a software testing perspective, the role of Artificial Intelligence (AI) techniques can be of tremendous benefit, as they automate and improve accuracy and intelligence. Several Machine Learning (ML) models examine past testing data to predict the possible location of defects,

subsequently tailoring the testing processes. All these approaches minimize effort by reducing the amount of redundant testing streams. Most importantly, Natural Language Processing (NLP), a defined technique, automatically transforms predefined requirements into processes that can be executed, which reduces the need to generate manual test cases.

Reinforcement Learning (RL) employs a reward-based mechanism for effective test execution, increasing AI's ability to identify bugs, leading to faster and more economical critical issue resolution for testing teams.

Comparison of Different Techniques & Real World Applications

Emerging forms of AI are becoming increasingly commonplace. For instance, Genetic Algorithms (GA) uses evolution-based techniques to form particular sets of test cases which are especially effective during stress testing and greatly aid in covering edge cases that were previously thought to be impossible to cover, using traditional methods. On the other hand, Deep Learning (DL) models are capable of finding hidden bugs that would otherwise be impossible to find, by recognizing complex patterns and behaviors in large scale software systems. Like any method, these AI approaches also have their disadvantages: ML and RL require good training data, GA can demand high computational resources, and DL models often lack explainability. As with any approach, Artificial Intelligence systems' techniques should always be tailored to specific requirements, taking into consideration the complexity of the system and resources at hand.

This excerpt illustrates the application of AI in software testing by renowned companies. For example, Google utilizes Machine Learning to optimize the order of test case execution for Android applications, increasing the likelihood of early defect detection and minimizing the duration of testing. Facebook (Meta) utilizes Natural Language Processing (NLP) tools that create unit tests from descriptions contained in pull requests, automating a significant part of the software development

and review process. Uber employs reinforcement learning to mobile testing of its apps, ensuring that critical path testing is done earlier in the CI/CD pipeline. Microsoft designed DeepTest, a deep learning-based tool that generates virtual driving scenarios to test the software of autonomous cars, subjecting it to rigorous tests under potentially dangerous conditions that occur infrequently.

Enterprise service providers have also implemented AI tools into their business operations. A case in point is Accenture, which uses genetic algorithms to devise optimization test suites with maximal code coverage and reliability for enterprise software testing in domestic industries like healthcare and finance. Such cases prove that AI techniques can speed up the testing process, reveal additional levels of automation, and lower the workload on human testers. The importance of AI within software testing will continue to increase, making it invaluable as software engineering advances due to the rising complexity of systems.

Works Cited

"AI Testing Automation Platform." Testers.ai, https://testers.ai/

"Appvance IQ Platform." Appvance AI, https://appvance.ai/aiq-platform

Arbon, Jason. "AI-Informed Test Teams." Medium, 8 Mar. 2025, https://jarbon.medium.com/ai-informed-test-teams-1efbdc004ebb

Arbon, Jason. "The Agentic Engineering Loop." Medium, 13 Apr. 2025, https://jarbon.medium.com/the-ai-engineering-loop-e4064f2e1c4c

Bakar, Mohammad, and Rajat Khanda. "AI-Powered Test Case Generation and Validation." arXiv, https://arxiv.org/pdf/2409.05808

"DeepTest 2025 - ICSE 2025." Researchr, https://conf.researchr.org/home/icse-2025/deeptest-2025

"DragonCrawl: Generative AI for High-Quality Mobile Testing." Uber Blog, 23 Apr. 2024, https://www.uber.com/blog/generative-ai-for-high-quality-mobile-testing/

"Functionize Overview." Functionize, https://www.functionize.com/overview

"Introduction to Appvance IQ." Appvance AI, 25 Oct. 2023, https://docs.appvance.net/Content/AIQ/AIQ_Intro/Introduction_to_Appvance_IQ.htm

Johnson, Elly. "Functionize Overview, Features, and Advantages." Test Automation Tools, 30 Jan. 2024, https://testautomationtools.dev/functionize-overview-features-advantages/

Priyanka. "Functionize vs Selenium: A Detailed Comparison." Testsigma, https://testsigma.com/blog/functionize-vs-selenium/

Sogeke, Bosun. "The Role of AI and Machine Learning in Software Testing." Medium, 4

Nov. 2024, https://medium.com/@sogekebosun/the-role-of-ai-and-machine-learning-in-software-testing-636ea6a76cb5

"Quality Engineering | Software Testing Services and Solutions." Accenture, https://www.accenture.com/us-en/services/testing-index