Katalon

A Review of AI-augmented End-to-End Test Automation Tools

Phuoc Pham

Katalon Inc.
Ho Chi Minh City, Vietnam
phuoc.pham@katalon.com

Vu Nguyen

Katalon Inc.
University of Science
Vietnam National University
Ho Chi Minh City, Vietnam
nvu@fit.hcmus.edu.vn

Tien Nguyen

Katalon Inc.
University of Texas at Dallas
Texas, USA
tien.n.nguyen@utdallas.edu

ABSTRACT

Software testing is a process of evaluating and verifying whether a software product still works as expected, and it is repetitive, laborious, and time-consuming. To address this problem, automation tools have been developed to automate testing activities and enhance quality and delivery time. However, automation tools become less effective with continuous integration and continuous delivery (CI/CD) pipelines when the system under test is constantly changing. Recent advances in artificial intelligence and machine learning (AI/ML) present the potential for addressing important challenges in test automation. AI/ML can be applied to automate various testing activities such as detecting bugs and errors, maintaining existing test cases, or generating new test cases much faster than humans.

In this study, we will outline testing activities where AI has significantly impacted and greatly enhanced the testing process. Based on that, we identify primary AI techniques that are used in each testing activity. Further, we conduct a comprehensive study of test automation tools to provide a clear look at the role of AI/ML technology in industrial testing tools. The results of this paper help researchers and practitioners understand the current state of AI/ML applied to software testing, which is the first important step towards achieving successful and efficient software testing.

CCS CONCEPTS

Software and its engineering → Software testing and debugging

ACM Reference Format:

Phuoc Pham, Vu Nguyen, and Tien Nguyen. 2022. A Review of AI-augmented End-to-End Test Automation Tools. In 37th IEEE/ACM International Conference on Automated Software Engineering (ASE '22), October 10–14, 2022, Rochester, MI, USA. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3551349.3563240

1. INTRODUCTION

Software is becoming more complicated, and consumer demand is growing higher than ever [27]. Developers are under increasing pressure to release new features and updates rapidly. Therefore, to deliver high-quality products in a short amount of time, it is essential to use software automation testing methods effectively.

Test automation has been supporting the continuous testing process for many years to automate repetitive and other testing tasks that are difficult to perform manually. It is an essential step in the software development process to detect errors and ensure the system under test (SUT) satisfies the specified requirements by leveraging automated software tools for executing test cases without human intervention. This can be achieved by writing test scripts or using any automation testing tool.

However, even with automated testing, many testing activities are far from satisfactory, such as maintaining a huge amount of automated test cases or selecting test cases to execute. These activities are challenging and not straightforward, even for skilled Quality Automation (QA) engineers. They must consistently monitor and keep the automation suite up-to-date with the application in every release cycle. As a result, it introduces a lot of burdens to the software development process. Today, with the advent of AI in domain software testing, the general test automation scenario is also changing. Many of the above problems can be addressed smoothly, much faster, and more accurately. AI-based testing aims to make the testing process smarter and highly effective, eliminate repetition and ensure refinement.

AI is now transforming the software testing industry in ways that could not have been imagined. Researchers and practitioners recognize the potential for AI/ML to bridge the gap between human and machine-driven testing capabilities. As a result, several companies are developing AI-powered automated testing tools. Since 2014, there has been a spike in the number of vendors offering AI-driven test automation services [19]. Most tool vendors are startup companies targeting system-level testing of web, mobile, and desktop applications. They aim to mimic human behavior, making them highly useful for testers involved in automated and precise continuous testing processes.

There are a lot of industry tools that we can use for automated testing, and each of them may use different intelligent techniques to tackle the same problem and serve different purposes. For example, Parasoft SOAtest [26] uses AI to improve the automation of service testing. Applitools leverages AI to automate visual analysis of web and mobile applications [23]. Thus, selecting a suitable tool for the current project is challenging. Given such a challenge, it is essential to understand and validate different aspects of AI in software testing. In this paper, we conducted a survey about the landscape of AI-powered software solutions to determine what testing activities they can support and what AI techniques are used to automate these activities. Understanding these aspects is the first crucial step towards the success of the test automation plan.

The paper is structured as follows: Section 3 covers test activities in the testing life cycle that AI/ML has significantly impacted, and how AI/ML can fit into these activities will be elaborated in detail in this section. Section 4 covers in-depth AI-powered features that have been used in each testing activity of popular industrial tools and the benefits they can bring to end users. Based on the research and study in this section, we aim to provide good evaluation factors and a good reference for selecting the right test automation tool for a particular project of interest.

2. RELATED WORK

Focusing on AI/ML in software testing, we found several studies closely related to our work that have been published in the past few years. Juristo et al. [14] analyzed the maturity level of the knowledge about testing techniques by examining existing empirical studies on these techniques. Nico Krüger et al. [22] discussed how AI and ML could be leveraged to improve test management, quality, and the impact of changes from design into production. King et al. [20] yielded insightful perspectives on AI in software testing in practice by bringing together six industry experts into panel discussions to figure out future directions for developing AI-powered testing systems. Sugali et al. [30] recognized and explained some of the biggest challenges software testers face in dealing with AI/ML applications. Durelli et al. [5] highlighted the most widely used ML algorithms to automate various software engineering activities and identify several avenues for future research. To our best knowledge, no related work is concerned about how to use AI/ML to automate major testing activities and the current AI-powered industrial tool landscape in leveraging AI/ML to deal with these testing activities. This paper aims to provide a comprehensive overview of this missing part in the software testing community.

3. HOW IS AI/ML CHANGING THE SOFTWARE TESTING PROCESS?

Software testing involves many activities such as requirements analysis, test management, test planning, test generation, test execution, test evaluation, and report [8, 11]. These testing activities are complex, time-consuming, and require skilled engineers to get the task done. Given that many activities can be formulated as learning problems, there has been a growing interest in using AI/ML to automate these activities, improving the testing quality in a minimal amount of time. However, not all stages can be beneficial and possible to automate; for example, humans still play a vital role as decision-makers in testing activities such as planning, management, or reporting. In this paper scope, we focus on main testing activities where AI/ML has enhanced the process to a large extent and made significant impacts; these activities included: test case generation, test data generation, test execution, test maintenance, and root cause analysis.

3.1 Test Script Generation

Creating test cases is an essential step during the software testing process [4]. However, making different test cases for various scenarios is challenging, slow, and inefficient. Each test script needs to be manually written, tested and debugged step-by-step. This is one of the testing activities that greatly benefit applying AI/ML for automating. These AI/ML techniques can help QA engineers to automate creating test cases partially or entirely.

This technique can be applied for visual testing and functional testing. For example, an automated test case generator can take web elements such as a button and create relevant test cases for validation. In this case, AI techniques such as natural language processing or computer vision can be used to understand the SUT and generate multiple test cases whenever there's an update in the product's features to ensure the application functionalities are still working as expected.

Another advantage of generated test cases is its self-maintenance capability [28]. As the machine plays the primary role of generating test cases, it can automatically select and keep records and metadata for web elements that it will interact with. Based on this stored information, it can update the existing test cases accordingly to avoid false positives which may arise. We will elaborate on this advantage in more detail in Section 3.4 about test maintenance.

3.2 Test Data Generation

Test data generation is another promising area in the context of data-driven testing. In some testing scenarios, valid test data must be provided to verify the application functionalities, for example, form logging, registering a new account, entering the receiver's address, etc. Test data can be generated either based on project specification [33] or source code. Many techniques have been proposed, such as using possible combinations based on a previously collected dataset, searched-based data generation [25], or heuristic approaches [13]. Test data generation can be used in conjunction with test scripts to satisfy functional business requirements, improve the overall data coverage and bring more confidence to the automation project. Currently, test data generation using AI and ML is in its infancy in both academic and industry. With more research being carried out and as tools improve, soon. It will help QA engineers scan through large code bases to understand this context in-depth and identify critical areas for test coverage focus.

3.3 Test Execution

Executing test scripts on different browsers and environments always requires significant effort. The testing may vary between browsers and testing environments, and the test automation suite

grows together with the code base. It is challenging and time consuming to handle by humans [12]. Automated managing, prioritizing, and scheduling of which test cases need to be run and which different devices and configurations should be selected is an essential task for smooth test execution. With the support of AI/ML, these problems mentioned above can be addressed efficiently, such as only running regression tests for significant action flows in the application and automatically verifying the application with multiple combinations of devices, operating system, and browsers. This introduction of AI/ML can increase test coverage and execution speed and save significant efforts and resources for executing test cases.

3.4 Test Maintenance

Code-based test automation is often unstable as it uses selectors to interact with web UI. However, these selectors are widely known to be fragile [10]. As a minor UI modification, it can alter every element selector, resulting in many test breakages. Therefore, it is mandatory to constantly update test scripts to keep up-to-date with the application's source code. This vicious cycle has to repeat with new application releases. A new era of tools has evolved to address these problems, where AI/ML assists test maintenance. It can help QA engineers to keep up with application changes by dynamically updating test cases. It is commonly called as a self-healing feature in many research communities and industry tools. In this approach, the ML engine resides in various techniques such as data analytics, visual hints [29], natural language processing [21], or other heuristic approaches [2] to identify objects in a script even after they have changed. When your script fails due to being unable to find the object it expected, the self-healing mechanism provides a complete understanding and analysis of possible alternative options and selects the one most similar to the object previously used.

3.5 Root Cause Analysis

Root cause analysis [15, 34] is a good quality control measure that provides QA engineers with insights on what exactly goes wrong, at what point, and understanding the reason behind it. Then, QA engineers have to check whether this is a real failure, a flaw in design and development, or just a test script problem. Tracking such failures takes a lot of time, even for an experienced QA engineer. In this context, AI/ML can identify the test cases which are impacted due to a feature change and trace them from impacted user stories and feature requirements to determine the impacted test cases and test scripts [9]. By leveraging AI/ML to handle these tasks automatically, QA engineers can avoid wasting time stepping into false positive error reports

4. HOW ARE INDUSTRIAL TOOLS APPLY AI/ML IN THEIR APPLICATIONS?

Today, there are so many test automation tools that are AI-enabled. Each tool has its own characteristic, scope, and target customers. Due to page limitations, in this section, we select eight representative testing tools from Gartner Peer Insights about Software Test Automation [7] for further detailed analysis of how AI/ML techniques can be integrated into their products to improve testing quality, saving both time and cost for QA engineers.

4.1 Katalon Studio

Katalon Studio [16] is an end-to-end test automation solution that leverages Selenium's core engine. It offers AI-enabled visual testing features with three image comparison methods: pixel-based, layout-based, and content-based comparison. These visual testing options significantly improve the speed and efficiency of UI quality testing. For test maintenance, its self-healing mechanism [17] can use different pre-configured locators (XPath, attributes, CSS and Image-based locator) to find the web element that will be used for the remaining execution. In addition, its Smart Wait [18] can automatically wait for all front end processes of the web page loaded before moving on to the next steps, completely preventing unstable outputs happening.

4.2 Applitools

Applitools [3] is an AI-enabled, end-to-end automation tool for visual UI testing and monitoring. For test generation, it can integrate existing tests smoothly and eliminates the need for manual writing. Its Visual AI feature enables a machine to mimic human behavior to recognize functions in a web page and then analyze the entire screen of the application to detect visual bugs (such as overlapping, invisible, off-page, and unexpected features that have appeared). Its comparison algorithms recognize whether the changes are meaningful or just bugs. As the algorithm is entirely adaptive, there is no manual setup or pre-defined configurations needed beforehand.

4.3 Testim

Testim [32] is an end-to-end AI testing tool that mainly focuses on functional testing and UI testing. For test execution, it leverages machine learning to speed up the test automation authority and execution, reducing test creation-execution time by running multiple tests simultaneously. It overcomes the problems of slow authoring and unstable tests that usually result from frequent changes and releases in the UI. For test maintenance, its Smart Locators detect the changes in the app to run automatic tests, reducing flaky tests and test maintenance. For root cause analysis, it integrates seamlessly with CI/CD tools, provides detailed bug reports, and performs root-cause analysis of failed tests for quick remedial action.

4.4 TestCraft

TestCraft [31] is an AI-powered test automation platform for re- gression, monitoring, and constant testing that works on top of Selenium. For test generation, it can generate a test model out of one's test scenario instead of a recording, making it simpler to reuse test steps and implement changes as needed. As a result, any modification to a test step will apply automatically to all other scenarios that use it, reducing test maintenance time. For test execution, it can recognize web elements correctly even when a web application change happens by assembling all possible data on all web page elements at a given time. These attributes are then fed into the algorithm, determining the most feasible part it should look for in any given run. For test maintenance, it leverages ML technologies to dramatically reduce maintenance time and cost, limiting unnecessary test breakages by overcoming UI changes.

4.5 Parasoft SOAtest

Parasoft SOAtest [26] is a testing and analysis tool suite for APIs and API-driven applications. For test execution, it leverages AI/ machine learning to model the relationships between tests and the underlying code. The model is used to perform test impact analysis on new or modified code to provide more intelligent and targeted test automation. The result is more intelligent and accurate, saving both time and effort. For test generation, it leverages AI/ML to process service definitions and recorded traffic and produce maintainable and reusable test assets.

4.6 Mabl

Mabl [24] is a cloud-based software testing tool . For test execution, its testing infrastructure is completely managed in the cloud, scaling tests infinitely and running them all in parallel. Mabl uses ML algorithms to detect threats or issues and improves test execution. For test monitoring, it can identify and surface problems and alert possible impacts. For test maintenance, it can eliminate flaky tests by automatically detecting whether web elements have changed and dynamically updating related tests accordingly. For failure analysis, it can continuously compare test results to test history to quickly detect changes and regressions, resulting in more stable releases. In addition, Mabl can learn from the previous testing processes and provide a list of recommended fixes for QA engineers; they can choose to approve, alter or reject these fix recommendations.

4.7 AccelQ

ACCELQ [1] is a cloud-based codeless AI testing automation tool that focuses on web UI, API, desktop, and mobile platform automation. For test generation, it uses AI/ML to auto-generate test cases with high coverage, capture the structure of test data based on business semantics, and ensure data abstraction is consistently driven across test scenarios. In addition, its AI-powered natural language automation allows users to generate test cases from the natural English description, no coding needed.

4.8 Functionize

Functionize [6] is a cloud-based automated testing for functional and performance. This tool uses AI/ML to speed up test creation, diagnosis, and maintenance. For test generation, its AI-powered smart agent creates tests quickly and uses natural language processing for automated generating test cases. For test maintenance, its SmartFix feature can detect UI changes and test failures. For root cause analysis, its ML engine autonomously identifies the root cause of broken tests and uses computer vision to identify differences in your layout and structure over time.

5. CONCLUSION

Test automation has become an essential part of successful software testing. Many start-ups and companies are using AI/ML extensively to enhance and address remaining issues in software testing. As a result, various automation testing tools are available with different characteristics and serving purposes, making it challenging to choose the most suitable one.

In this paper, we surveyed several main testing activities where AI/ML has made significant impacts. Based on that, we conducted a comprehensive survey to show how AI-powered tools tackle each testing activity. We believe our work will value research and the industry community and give an overall picture of AI/ML applied in software testing. Based on that, researchers can open new directions to meet industry needs effectively, and QA engineers can make better decisions when picking automation tools for a given project.

REFERENCES

[1] AccelQ. 2022. AI-Powered Codeless Test Automation on the Cloud. September 5, 2022 from https://www.accelq.com/
[2] Iñigo Aldalur, Felix Larrinaga, and Alain Perez. 2020. ABLA: An algorithm for repairing structure-based locators through attribute annotations. In the International Conference on Web Information Systems Engineering. Springer, 101–113.
[3] Applitools. 2022. Visually Perfect Applications with Automated Visual Testing - Applitools. Retrieved September 5, 2022 from https://applitools.com/
[4] Fabrice Bouquet, Christophe Grandpierre, Bruno Legeard, and Fabien Peureux. 2008. A test generation solution to automate software testing. In Proceedings of the 3rd international workshop on Automation of software test. 45–48.
[5] Vinicius HS Durelli, Rafael S Durelli, Simone S Borges, Andre T Endo, Marcelo M Eler, Diego RC Dias, and Marcelo P Guimaraes. 2019. Machine learning applied to software testing: A systematic mapping study. IEEE Transactions on Reliability 68, 3 (2019), 1189–1212.
[6] Functionize. 2022. Agile Automation Testing Framework with Machine Learning. Retrieved September 5, 2022 from https://www.functionize.com/
[7] Inc. Gartner. 2022. Reviews for Software Test Automation Reviews 2022. Re- trieved September 5, 2022 from https://tinyurl.com/gartnerReviewsSoftware
[8] David Gelperin and Bill Hetzel. 1988. The growth of software testing. Commun.ACM 31, 6 (1988), 687–695.
[9] Ruchika Gupta. 2021. Test Failure Analysis With AI-Critical To DevOps. Retrieved September 5, 2022 from https://tinyurl.com/webomatesTestFailure
[10] Mouna Hammoudi, Gregg Rothermel, and Paolo Tonella. 2016. Why do record/replay tests of web applications break?. In 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST). IEEE, 180–190.
[11] Mary Jean Harrold. 2000. Testing: a roadmap. In Proceedings of the Conference on the Future of Software Engineering. 61–72.
[12] IttiHoodaandRajenderSinghChhillar.2015.Softwaretestprocess,testingtypes and techniques. International Journal of Computer Applications 111, 13 (2015).
[13] Neetu Jain and Rabins Porwal. 2019. Automated test data generation applying heuristic approaches—a survey. In Software Engineering. Springer, 699–708.
[14] Natalia Juristo, Ana M Moreno, and Sira Vegas. 2004. Reviewing 25 years of testing technique experiments. Empirical Software Engineering 9, 1 (2004), 7–44.
[15] Julen Kahles, Juha Törrönen, Timo Huuhtanen, and Alexander Jung. 2019. Au- tomating root cause analysis via machine learning in agile software testing environments. In 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST). IEEE, 379–390.
[16] Katalon. 2022. Katalon | Simplify Web, API, Mobile, Desktop Automated Tests — katalon.com. Retrieved September 5, 2022 from https://katalon.com/
[17] Katalon. 2022. Self-healing Tests | Katalon Docs. Retrieved September 5, 2022 from https://katalon.com/resources-center/blog/self-healing-object-locator
[18] Katalon.2022.[WebUI]SmartWaitFunction. RetrievedSeptember5,2022from https://katalon.com/resources- center/blog/handle- selenium- wait
[19] Tariq King. 2020. The Current State & Future Trends of AI in Software Testing. Retrieved September 5, 2022 from https://tinyurl.com/perfectoCurrentState
[20] Tariq M King, Jason Arbon, Dionny Santiago, David Adamo, Wendy Chin, and Ram Shanmugam. 2019. AI for testing today and tomorrow: industry perspectives. In 2019 IEEE International Conference On Artificial Intelligence Testing. IEEE, 81– 88.
[21] Hiroyuki Kirinuki, Shinsuke Matsumoto, Yoshiki Higo, and Shinji Kusumoto. 2021. NLP-assisted Web Element Identification Toward Script-free Testing. In 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, 639–643.
[22] Nico Krüger. 2020. AI Testing and Machine Learning in Software Testing | Per- force Software — perforce.com. Retrieved September 5, 2022 from https://www. perforce.com/blog/alm/ai- testing- and-machine- learning- software- testing

[23] George Lawton. 2019. How autonomous software testing could change QA. Retrieved September 5, 2022 from https://tinyurl.com/ techtargetAutonomousSoftware
[24] Mabl. 2022. Intelligent Test Automation for Agile Teams. Retrieved September 5, 2022 from https://www.mabl.com/
[25] PhilMcMinn.2004.Search-basedsoftwaretestdatageneration:asurvey.Software testing, Verification and reliability 14, 2 (2004), 105–156.
[26] Parasoft. 2022. Parasoft SOAtest — parasoft.com. Retrieved September 5, 2022 from https://www.parasoft.com/products/parasoft-soatest/
[27] Zheng Qin, Xiang Zheng, and Jiankuan Xing. 2008. Introduction to software architecture. Springer.
[28] Sina Shamshiri, José Miguel Rojas, Juan Pablo Galeotti, Neil Walkinshaw, and Gordon Fraser. 2018. How do automatically generated unit tests influence soft- ware maintenance?. In 2018 IEEE 11th international conference on software testing, verification and validation (ICST). IEEE, 250–261.
[29] Andrea Stocco, Rahulkrishna Yandrapally, and Ali Mesbah. 2018. Visual web test repair. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 503–514.
[30] Kishore Sugali. 2021. Software Testing: Issues and Challenges of Artificial Intelli- gence & Machine Learning. (2021).
[31] TestCraft. 2022. Introducing TestCraft for Codeless. Retrieved September 5, 2022 from https://www.perfecto.io/blog/introducing-testcraft
[32] Testim. 2022. Automated UI and Functional Testing - AI-Powered Stability - Testim.io. Retrieved September 5, 2022 from https://help.testim.io/docs
[33] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. Software testing, verification and reliability 22, 5 (2012), 297–312.
[34] Carter Yagemann, Simon P Chung, Brendan Saltaformaggio, and Wenke Lee. 2021. Automated bug hunting with data-driven symbolic root cause analysis. In Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. 320–336.