Are You Ready for AI to Take Over Your Automation Testing?

Lisette ZOUNON, CSP- CSM, CALI, DTM

zzlisette@gmail.com

Abstract

We all know that Artificial Intelligence is here and here to stay. Every industry leader has been waiting for how and when AI will disrupt their work. Most teams are wondering how to leverage AI for their quality automation. Indeed, Artificial intelligence has made some major strides in recognizing patterns in software testing and the opportunity for QA engineers to leverage some features that can benefit testing activities and software delivery teams. Although there are several QA automations tools out there, there is a clear advantage in the modern software automation tool leveraging AI.

Attendees in this session will take away:

- Determine if your automation strategy can leverage the benefits of AI tools
- Discuss success case study of using AI tools as part of automation testing
- Leverage AI tools in your CI/CD environment
- Discuss the ROI for your project and your team happiness factor.

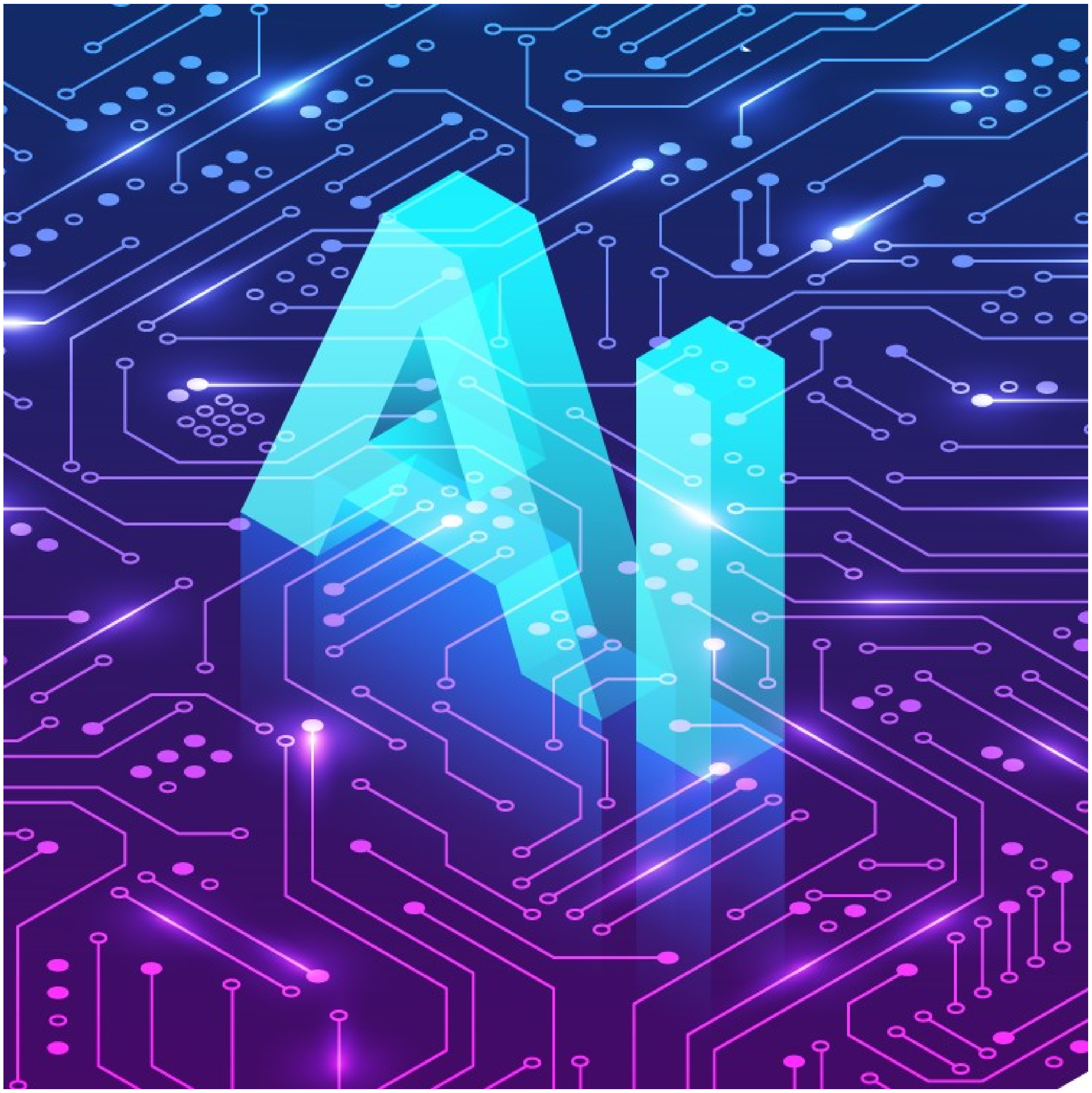
Biography

Lisette Zounon is a passionate quality engineering leader with over 16 years of experiences helping people and companies improve the quality of their applications, with solid tools, a simple process, and a smart team.

Lisette was responsible for leading and managing high-performing quality-testing teams throughout all phases of the software development testing cycle. This includes establishing and maintaining the QA strategy, processes, platforms, and resources needed to deliver 24x7 operationally critical solutions for many of the world's largest companies.

Lisette is a proven leader who thrives in a highly technical software development environment. She is a constant champion of employing best practices for QA, agile methodologies, and Scrum implementation to achieve high quality delivery and increase your team's and customers' happiness.

1. Introduction



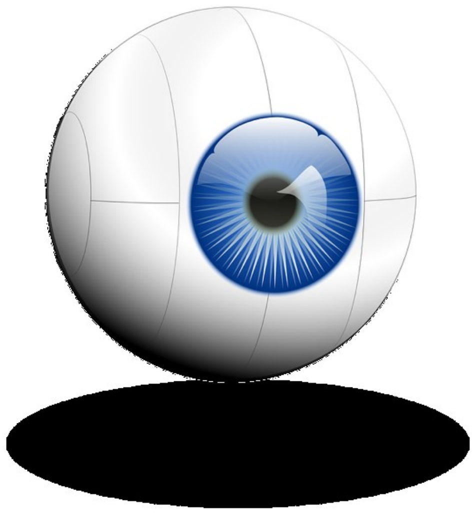
In the past ten years, most QA automation tools provide QA engineers opportunities to create test automation, but we still lack in some areas such as efficiency in execution, reusability of the test cases, and test maintenance update. Manual testing may take a long time whenever a break in the script occurs. QA automation has room for improvement to enable fast error detection. At its core, Machine Learning is a pattern-recognition technology—it uses patterns identified by your machine learning algorithms to predict future trends.

Artificial intelligence is becoming very crucial in information technology. In the last few years, software automation has been experimenting with AI and machine learning. Below I am going to detail some of the benefits of AI-based tools. My personal approach to this paper is really to be tool agnostic in terms of considering what benefits you can drive from Artificial intelligence. I am not going to provide you with a specific tool but focus on the functionality that AI-based tools can provide software testers and automation engineers.

2- Determine the benefits of using an AI-based tool

2.1 Visual validation testing

Because of the accuracy of AI-powered automated visual testing, they can be deployed in more than just functional and visual testing. Visual testing is about verifying if the visual aspects of an application seem appropriate to the end-users. Visual and functional testing complement each other to help you take a holistic approach to test the user interface (UI) of your application. Visual validation automation testing also refers to automated validation UI testing, uses Machine Learning to make sure that UI appears correctly to the users. From our naked eye, it is very difficult to spot very simple errors on the front UI of an application. Machine Learning tools can find differences that human testers miss. Visual validation testing comes as a technique to help supplement testers' role in finding annoying visual errors. Most visual bugs are usually rendering issues.



The software engineer creates a script that drives your app, your app creates web pages with static visual elements. Functional testing changes visual elements, so each step of a functional test creates a new UI state you can visually test. Automated visual testing evolved from functional testing. Instead of the unnecessary need to write multiple assertions to check the properties of each visual element, automated visual testing tools visually check the visual appearance of an entire screen with just one assertion statement. This leads to test scripts that are simpler and easier to maintain. Visual smoke tests can be created to automatically check important web apps and site pages for correct loading, visual changes, broken links, JavaScript errors, and network activity issues across browsers. AI-powered visual testing tools are crucial to validating any application that requires a constant change in content and format. Visual testing and monitoring capabilities include:

- Review captured screenshots of the app state during test runs
- Get notified about insights for detected visual changes
- See a side-by-side comparison with the visual diffs highlighted
- Update the visual baseline to adjust for intended changes
- Create a fixed visual baseline to identify all visual diffs, including dynamic content

2.2 Test APIs

Artificial intelligence-enabled API testing provides a leap forward in productivity and efficiency. You can capture all the steps in one set. In addition to functional testing, you can work with web requests and automate testing of your application programming interfaces (APIs). Unlike visual tests that need to interact with a browser, API tests are performed at the message layer (e.g. http protocol) and therefore, run much faster than browser tests, completing in seconds, not minutes. With an AI-enabled API, you can define something once and lock it in in a smart test template and share that template across a large body of testers. It helps to manage and understand the full API inventory. This lowers the threshold of the skills needed for API testing; therefore, businesses can build a sustainable API testing strategy.

2.3 Self-healing of automated tests

You are probably testing software that changes on a regular basis. Some changes are major and require old tests to be rewritten. But many changes, such as relabeling a button or layout adjustments, are minor. A person might not even notice these changes, but they can easily break automated tests. Automated tests require a lot of maintenance when you have an application that changes frequently. Self-healing or auto-healing is used to help adapt to any minor changes, so the tests keep working. Every time tests are run; it interacts with an element; it collects element attributes that it uses to find the element next time and track change overtime.

The test could track many attributes such as text, CSS Classes, data, and other information like location and size on the page. Any test step that interacts with an element on the page can be auto healed. Sometimes the best way to find an element is to look for related elements that have particular text or other attributes. In these cases, a collection of information about the element's ancestor elements is used to help find the specific element. The general process of identifying and potentially healing an element starts with looking for element candidates by matches to the tracked attributes and then choosing the best match based on which combination of attributes matches for each candidate and the history of finds for that element and similar elements. If an ancestor is recorded for the element, this process happens first for the ancestor, then it identifies candidates for the target element within the ancestor and again chooses the best match.

At the beginning, it is usually a challenge for testers to trust and rely on auto healing. For most testers, auto-healing has been a time saver and successfully helps in test cases maintenance. After some time and with some tangible results of auto healing, QA engineers easily embrace the self-healing of automated tests.

AI-enabled automation can provide some more intangible value to QA testers. AI allows your tests to be reliable because your tests can now adapt to developer changes. AI is not here to take over quality assurance, but it is more to allow the quality engineer to become more productive and focus on what QA does best. Below are some of the advantages, as an unintentional consequence of AI has proven.

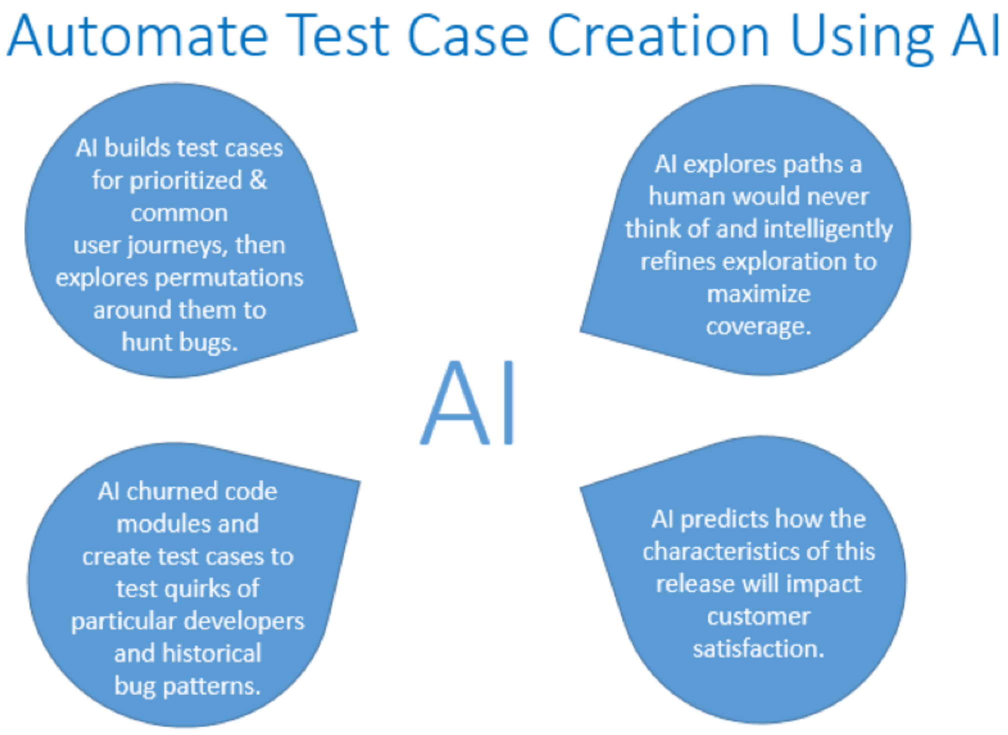
2.4 Improve test cases creation

One of the unintentional consequences of using AI-based tools for automation is really improving test creation. AI/ML algorithms can “read” your application and learn about it. These algorithms build a data set that contains observations about your application, including how its various features should behave given certain conditions. This helps them to automatically create test cases where they record the expected results. Given that these tools “learn” using their algorithms, their ability to create test cases is far superior to what rule-based automation can achieve. AI/ML helps learn application, hence creating test cases where they show expected results. This provides the ability to create superior test cases.

2.5 Spot duplicate errors and new errors in code changes

Another advantage of AI-based tools is the ability to track patterns, spot derivations, and flag it as a potential issue. As a testing professional, it is a daunting task to analyze failures and test flow. This means that dedicated professionals need to continuously monitor the process and the order in which things need to be executed.
With an advanced AI-based test procedure, one can expect the system to analyze failures and decide how to fix the errors based on its previous actions.

2.6 Create more reliable automated tests


Automate Test Case Creation Using AI
AI builds test cases for prioritized & common user journeys, then explores permutations around them to hunt bugs.
AI explores paths a human would never think of and intelligently refines exploration to maximize coverage.
AI
AI churned code modules and create test cases to test quirks of particular developers and historical bug patterns.
AI predicts how the characteristics of this release will impact customer satisfaction.

AI can open new possibilities in streamlining automation testing procedures for enhanced results. With AI, one can expect automation testing without coding. There are a lot of ways automation testing can be useful in lowering the efforts and reducing the time required to execute scripts multiple times. Reoccurring tests can be monitored without a test engineer with a machine learning approach. By enhancing the conventional testing procedure with machine learning, you can achieve results that surpass expectations.

3. Summary of successful case study of using AI-based tool

I have the opportunity to experience and use testing tools with functional testing and load testing, both leveraging artificial intelligence. In this paper, I would like to focus on one successful case study that leverages an AI-based tool for a QA automation transformation. Below is a summary of one case study among many using AI based automation tools.

3.1 Problem

For the last ten months, this high performing agile team has been working together on building a new site with various services for the customers. They had met the initial deadline after all the hard work to make the customers happy. But now they have a huge backlog of features that the customer expects them to deliver in the next six months. The team's concern is mostly around ensuring the new features; new code release does not break existing functionalities that the customers have come accustomed to enjoying. The only QA engineer in this agile team has created over 1100 test cases to successfully test all the features of this application. These test cases consisted of smoke tests, sanity tests, and functional test cases that are now part of regression test cases suites. QA engineer firmly believes that these test cases are a necessary part of testing for any new release. This QA member usually takes 2-4 days, three days on average to manually execute all the regression test cases after each code release in one environment. The goal is to ensure fast delivery, fast error detection from one environment to another, and finally fast feedback from the happy customers.

3.2 Solution

QA Leadership took charge of this problem and went out to look for a tool that can help solve this challenge and ensure fast delivery, fast error detection, and fast feedback. It was not an easy challenge to look for a new toolset. I personally believe in solid tools, simple process that empower smart team. I also like to ensure that you understand as a team the features and benefits that each testing tool provides. The features of the testing tools should support your QA automation strategies. We began by defining our QA automation strategies to select the appropriate testing automation tool. We reviewed dozens of QA automation tools from open source tools to paid testing systems. In terms of QA automation tools, most of the tools fall into two categories either a code scripting tool or a codeless tool. But there are few tools that leverage AI for visual validation, auto-healing, and API testing. We are in the area of intelligent testing so we set out to look for a tool that can:

- Offer intuitive interface to quickly create and manage test cases
- Not require specialized scripting expertise and no overhead

- Expand test coverage and evolve test quickly as application evolved

Here are our tool requirements and options for selection:

- Automation testing
- DevOps
- CI/CD, CT, CM (integration, delivery, testing, and monitoring)
- AI and bot for testing

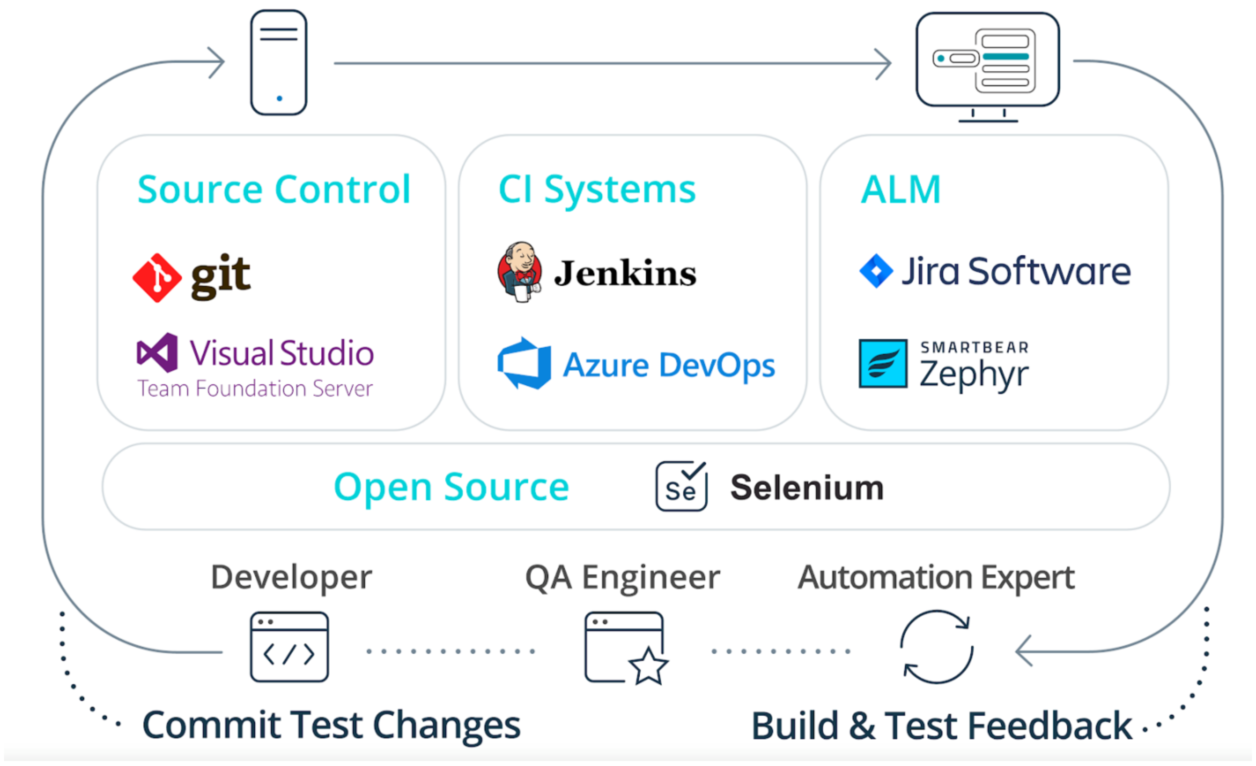
We review the following tools available on the market at that time:

- Selenium
- Test complete
- UFT/QTP
- Katalon Studio

- Mabl

The learning curve was also a crucial part in choosing a testing system. With a small team with no coding experience and prior QA automation experience, it was important to ensure training with a new tool was not time-consuming for the new QA engineer.

Interested Ecosystem



We choose an AI-based tool that the QA team member can leverage to solve this challenge. After sharing the tool with the agile team and demoing this tool for them, they all got excited and planned time in their sprint cycle for QA to automate the regression suite. The QA analyst with no prior coding experience starts creating test cases from smoke test, sanity test then regression test cases in the AI based tool in addition to training in parallel. We learn that the team can learn a new tool especially if they are focused with adequate documentation and support provided by a user community and support team. It took about 3-4 weeks for all these efforts in automating all the 1100 manual test cases. Every engineer's level of understanding and embracing a new tool is different. But in this case study, the total effort with training took about four weeks for this one QA engineer.

3.3 Result

With the implementation of test cases automated in the AI-based tool, the execution of the 1100 automated test cases take about 30 min to run successfully in one environment. Developers are able to leverage the smoke tests in the DEV environment. The sanity test cases were executed in the QA environment before the regression test cases were executed in the staging environment. These test cases are easily executed in another environment from QA to staging to production environments. The QA team is now focused on sprint work items testing and can leverage the regression suite testing to quickly execute test cases. QA team is able to leverage the test case automation in the CI/CD environment. This enables fast delivery of new features, fast error detection, and fast feedback to customers. The team now has a consistent execution of the same test cases in all environments. But the major ROI is the gain in time of execution of the test cases via the automated tool. Some of the advantages for the team is the quick error detection and the quick response from developers when a bug is found. QA is able to execute the regression suite for the entire application to provide a level of confidence to the stakeholders. We collect several quality KPI metrics for each test execution run and we realize that most of our success rate has increased. We also observed as a result that we have less than 5% of escape bugs found in production. This provides a high level of confidence to our stakeholders and customers were greatly satisfied with our collective efforts in investing in AI-based tools.

4. Return on Investment on the project and team happiness

The return on investment ROI on using AI-based tools in your QA automation is huge for your project and your team. Here are few that we know and have experienced:

1. Embrace lifelong learning mindset
2. Fast time to market
3. High-quality software
4. Fast feedback loop to customer
5. Fast response to the issue
6. Cross-functional team involvement
7. Increase team confidence
8. QA becomes a fun activity and no longer a bottleneck
9. Team happiness increase and better team engagement
10. Great teamwork and collaboration

5. The overall impact of AI on future of SDLC

Software development life cycle is definitely being impacted by artificial intelligence. With AI, the new approach will not only automate and facilitate the process but also result in models that are constantly trained and improved. Although agile development significantly accelerated the traditional software development life cycle, all components, including features, functionalities and integrations have to be manually managed and updated. Therefore, it leads to numerous bugs and inconsistencies due to the complexity of the system. With machine learning models, most of the features will be automated which means that human error will be eliminated. That is where the shift from agile to AI development is much more dramatic than that from waterfall to agile. One of the most important factors when it comes to software development is planning a budget and deadlines.

Unfortunately, artificial intelligence can be used to offer more precise estimates and predictions. It takes a lot of expertise and understanding of the nitty-gritty of every individual project, as well as being familiar with the implementation team for these estimates to be reliable. Machine learning can use the data from previous projects, including customer feedback, feature definitions, estimates, and final results to calculate how long it will take to build a new product and what its cost will be.

Although it has been quite a revolutionary change in software testing and quality automation, Artificial intelligence is still promising. Human interaction is still needed to create reliable, reusable test cases. And we know that there is still some bias embedded in AI-based tools.

6. Limitations of Artificial Intelligence

While AI presents an incredible amount of promise in software testing, there are some limitations that we must face, regarding current software testing efforts that involve AI. One of the first limitations is around the ability to create test cases. We don’t currently have the ability to have test cases create automatically. We are still relying on engineer with subject matter knowledge to create test cases. Another limitation is around AI-related bias. Bias can creep in at many stages of the deep learning process, and the standard practices in computer science aren't designed to detect it.

7. Conclusion

Artificial intelligence has a profound impact on software development. No matter whether you opt for an approach entirely based on machine-learning models or stick to the traditional SDLC agile approach with a machine-learning facelift, you can expect to boost your productivity, cut costs, speed up the entire development process, and create a more successful, easily-scalable product. Over the next three years, executives expect automation to increase their workforce capacity by 27 percent: equivalent to 2.4 million extras full-time employees. In their embrace of more digitized ways of working, many have adopted robotics to automate repetitive rules-based processes. And the software testing industry has successfully embraced QA systems and QA tools are now seeking to scale these solutions and make them smarter by integrating AI capabilities.

Test automation is a key capability in the age of agile since it facilitates faster product iterations. Given their ability to “learn”, AI-powered test automation tools bring a level of automation on the table that simple rule-based automation can’t achieve. Whether you are an IT leader in an enterprise or a tester, you should be opened to embrace the innovation that AI has unleashed in the world of test automation.

References

https://support.smartbear.com/testcomplete/docs/testing-with/running/self-healing-tests.html

https://techwireasia.com/2019/07/what-is-self-healing-automation-and-why-is-it-important-to-devops/

https://applitools.com/blog/visual-testing/

https://www.parasoft.com/to-make-api-testing-easier-add-machine-learning-to-your-ai/

https://techbeacon.com/app-dev-testing/how-ai-changing-test-automation-5-examples

https://testsigma.com/blog/can-ai-driven-test-automation-enhance-test-automation/

https://www.technology.org/2019/12/15/how-artificial-intelligence-affects-software-development/

https://www.techfriend.in/pros-and-cons-of-ai-based-software-testing.html