AI BASED TESTING

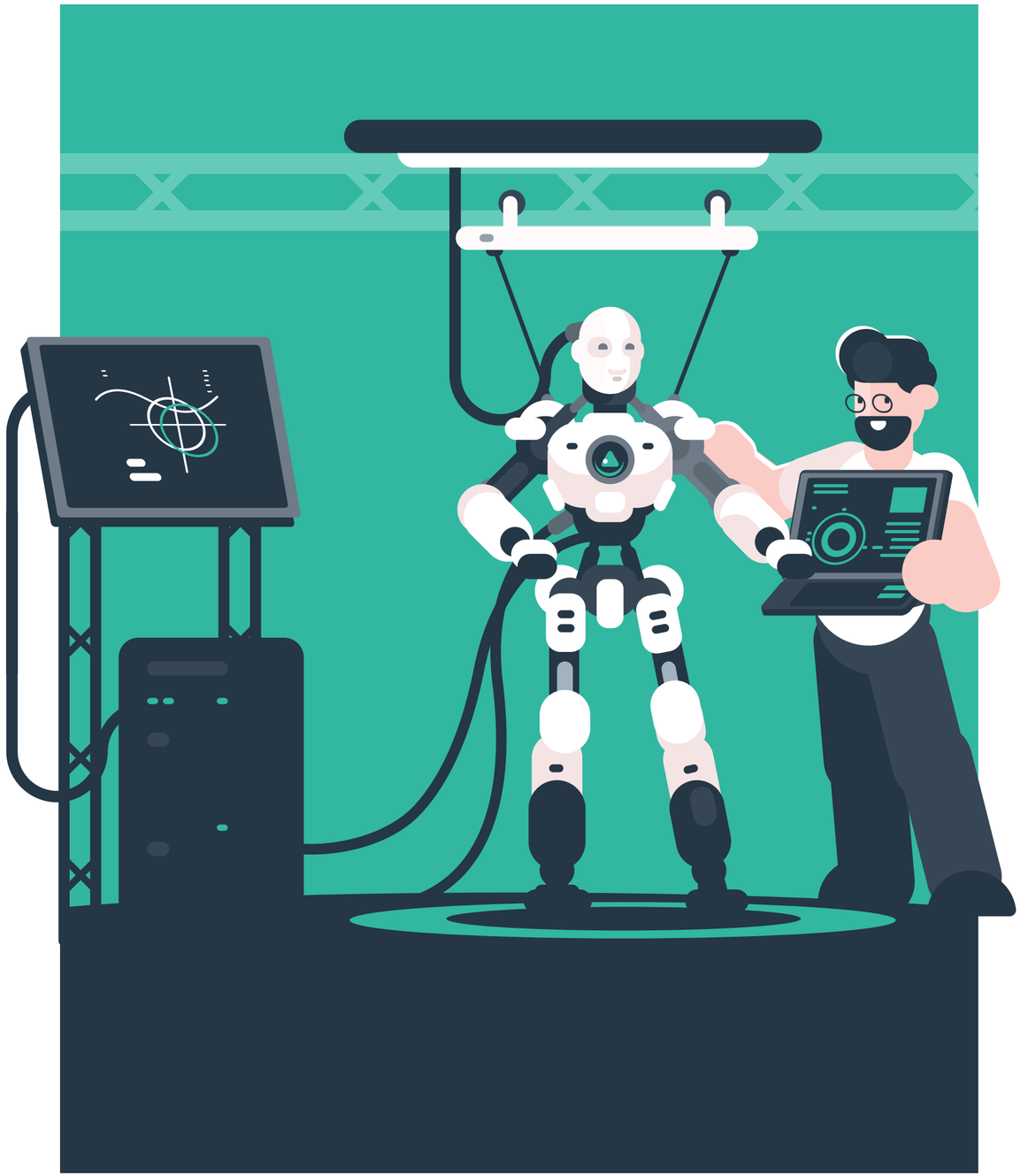
The Future of Test Automation

www.testim.io

About 5 years ago everyone was talking about “Mobile First” and giving the user a mobile experience using mobile web, native and hybrid applications. Now, the new buzzword is Artificial Intelligence (AI). Not a day goes by without people or articles mentioning some sort of AI development happening in self-driving cars, speech recognition, computer vision, healthcare, fintech and now in software testing. AI and machine learning are advancing at a rapid pace. Companies like Apple, Tesla, Google, Amazon, Facebook and others have started investing more into AI to solve different technological problems in the areas of healthcare, autonomous cars, search engines, predictive modeling and much more testing It’s affecting every business, big or small. This being the case how are we as Testers going to adapt to this change and embrace AI?

In this digital era, organizations are forced to trade off between faster time to market and flawless user experience. The current goal for organizations is to run more tests, find bugs fast and release faster. In this white paper, we will discuss how AI is impacting software testing, discuss current challenges with test automation, how AI can help to solve these challenges, how testers can embrace AI and what the future of automation looks like from the lens of AI.

Table of Contents



What is Artificial Intelligence?

Artificial Intelligence (AI), Machine Learning (ML) and Deep Learning (DL) are used synonymously but there are some differences.

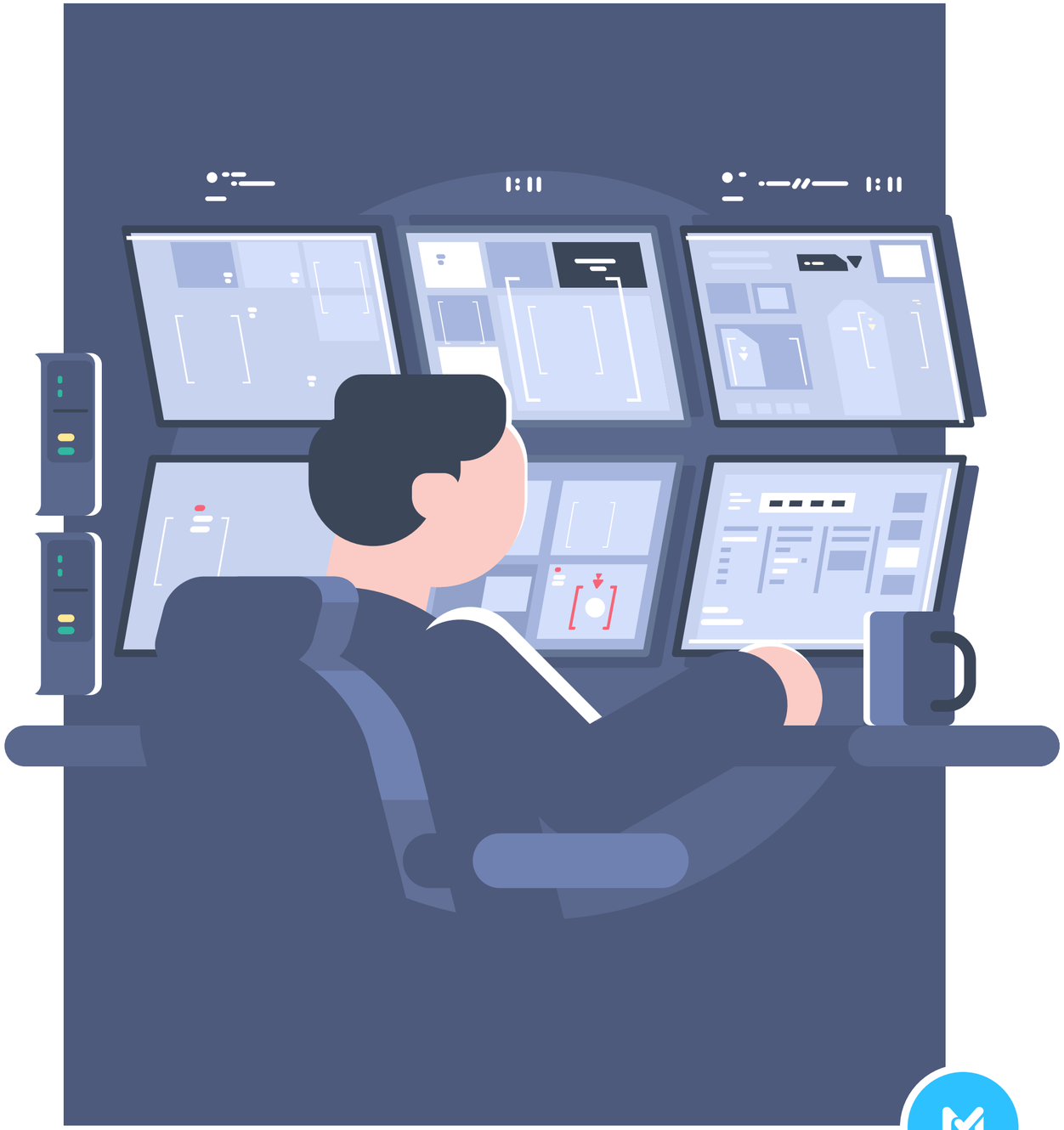
- AI is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans.
- ML is a subset of AI and evolved from the study of pattern recognition and computational learning theory (studying design and analysis of ML algorithms) in AI. It is a field of study that gives computers ability to learn without being explicitly programmed.
- DL is one of the many approaches to ML. Other approaches include decision tree learning, inductive logic programming, clustering and Bayesian networks. It is based on neural networks of the human body. Each neuron keeps learning and interconnects with other neurons to perform different actions based on different responses

AI is not a new concept. The first research was done in 1989 and there was even a video published on how Stanford did an experiment with using AI for cars (Autonomous Cars). The term AI has again become prominent in the past few years.

Impact to Software Testing

The role of technology within our personal and professional lives continues to evolve at an exceptionally fast pace. From mobile apps that command home appliances to virtual reality, the digital revolution is expanding to cover every aspect of the human experience. On a global scale, businesses are launching apps that are used by thousands, if not millions, of people. The majority of those businesses are developing apps using the Agile rapid delivery framework, which equates to roughly a new launch every 2 weeks. Before each launch, those apps must be tested to ensure an optimal experience for the end user. At that pace, manual testing is simply inadequate.

The time needed to complete the slew of required test cases directly conflicts with the fast pace driven by Agile-like frameworks and continuous development. The exploration of alternative and superior testing methods, such as automation and AI, is now a necessity in order to keep pace and equip QA and test teams with augmented efficiencies. AI is showing great potential in identifying testing defects quickly and eliminating human intervention. Such advances have provided the capability to determine how a product will perform, both at the machine-level and data-server level. AI just like automation tools is going to aid in the overall testing effort. In the current era where emphasis is on DevOps, CI/CD Integration and Continuous Testing, AI can help to fasten this process and make them more efficient.

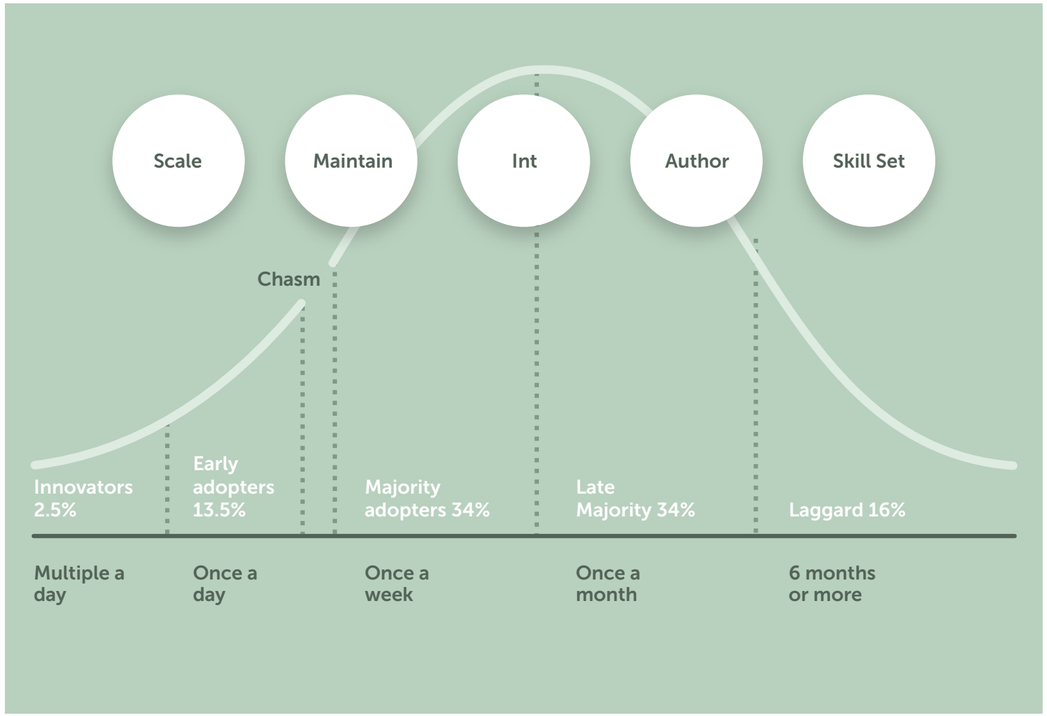


Current Challenges transitioning into Agile

Let’s discuss the current challenges industry face while transitioning into agile and how it affects release cycles. This will help to understand how AI can help to solve some of these challenges.

Impact to Software Testing

CATEGORY	CHALLENGES
Skill Set	Testers are doing only manual testing Skilled Testers with a developer mindset and experience are hard to find and expensive
Authoring	Now you have skilled. testers but they just started authoring a number of tests without following best practices Need to think about how to handle elements waits, making reusable components, parameterize the tests Authoring tests consumes a lot of time (On an average 6-8 hours)
Initialization state	Imagine a tester that wrote one test, and was passing successfully; they run it again, only to find that state of the app is different, and needs to be reset Common scenarios where testers need to take care of initial states are: Login scenario - Need to check whether a user is already logged in or not Adding items to a cart - Need to check for already added items in the cart
Maintenance	We gave 10 passing tests! We're super happy! The next day, we come to work, and all tests fail because of a change in the app. A recent survey states, this can be up to 20% of a tester work, and it's a big obstacle in adding more tests Now we start thinking about how to maintain all these tests because there are some flaky tests and there is some refactoring which needs to be done to increase stability
Scalability	Next, we have 500 tests, which take 5 hours to run. Especially if they fail and find bugs, it's really hard to release more than once a day Tests need to be written faster and the failures has been found fast if we want to release faster There is a need to start thinking about scale Whether the automated test/framework is extensive Need more server space for running Need to run tests in parallel Need to run the tests more often



CURRENT CHALLENGES IN RAPID RELEASE CYCLES

How AI can solve some of these challenges

First of all, organizations should take testing seriously and have a dedicated QA team with set process and standards. The most common mistake teams do is concentrate on developing the feature and when it comes to testing they get whomever is free during that time to test the software. Software testing is an art and a dedicated craft, we need curious testers with a great attitude and skeptical mindset, to question the product and evaluate it without any biases.

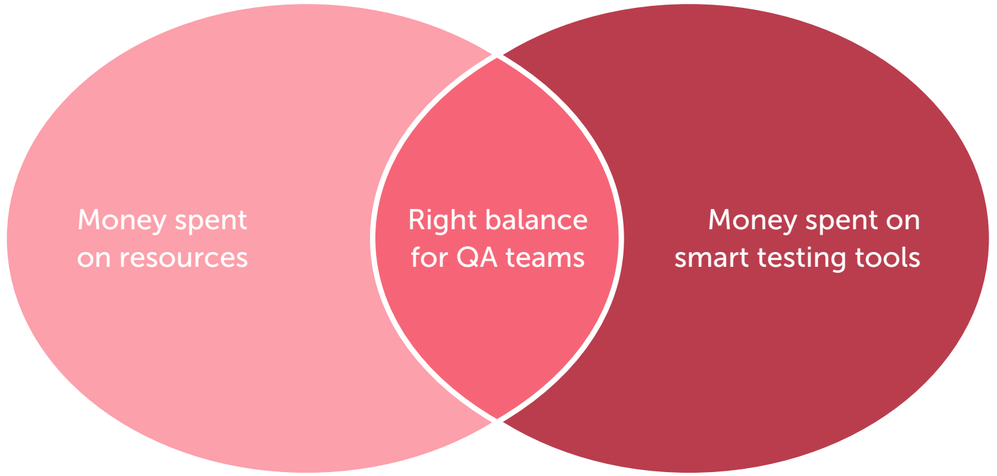
Secondly, forming a skilled dedicated QA team takes a lot of time, effort and cost. There is a fine balance we need to maintain in terms of how much money we are are going to invest in bringing in resources vs investing in a smart tool that can aid in testing. A good start would be to have 2-3 people dedicated to testing. They start simple by using the app running through high level scenarios using an AI based tool. The more tests we run the more data the AI gets to learn and start building relationships related to the application under test (AUT).

Now we have some tests authored the flows, the next challenge is applying best practices. Quite often, teams do not pay attention to best practices in authoring a test such as giving adequate waits, parametrizing your tests, creating reusable components and having single responsibility tests with low coupling. Also, testing starts late in the software development lifecycle (SDLC) making cost of finding bugs even higher as a lot of re-work needs to be done when bugs are found late in the SDLC and it affects timeline and release cycles. This is the reason we need to align with the "Shift Left Paradigm" where testing should start in parallel with development as early as possible, right from the requirements phase and continues all the way till release. This way, bugs can be found faster and cost of fixing them is way lesser than finding them in the later stages of the release cycle. While authoring these tests, we also need to pay attention to the initial states of the tests. For example - Say we are writing a script to add items to a shopping cart. Everytime we run the test, we need to ensure the test is set to its initial state of having an empty cart before adding items. This prevents skewed results and is in line with good practices of automation.

The next biggest problem with test automation is "Maintenance". As the complexity of software increases, we end up adding more tests. As a result, we have plethora of tests to run and maintain. When tests fail, we need to spend a lot of time troubleshooting the failure and fixing it. A recent study conducted found that, testers spend about 40% of time in just maintaining the tests. Imagine what can happen if we reduce this to say 1-5%. Won't that be amazing? Testers can now spend their valuable time actually testing the software instead of troubleshooting tests. This can be made a reality by using **Dynamic Locators**.

This is a strategy where the AI in real time parses multiple attributes of each and every element the user interacts with in the application and creates a list of location strategies. So, even if an attribute of an element changes, the tests do not fail; instead the AI detects this problem and goes to the next best location strategy to successfully identify the element in the page. In this way, the tests are more stable and as a result, the authoring and execution of tests are really fast as well.

Finally, AI can help in overcoming the challenge of scale by finding bugs fast and releasing faster and release faster through the self-healing mechanism; where the AI proactively identifies and solves problems before it even occurs, this way our approach to test automation will be more proactive than reactive.



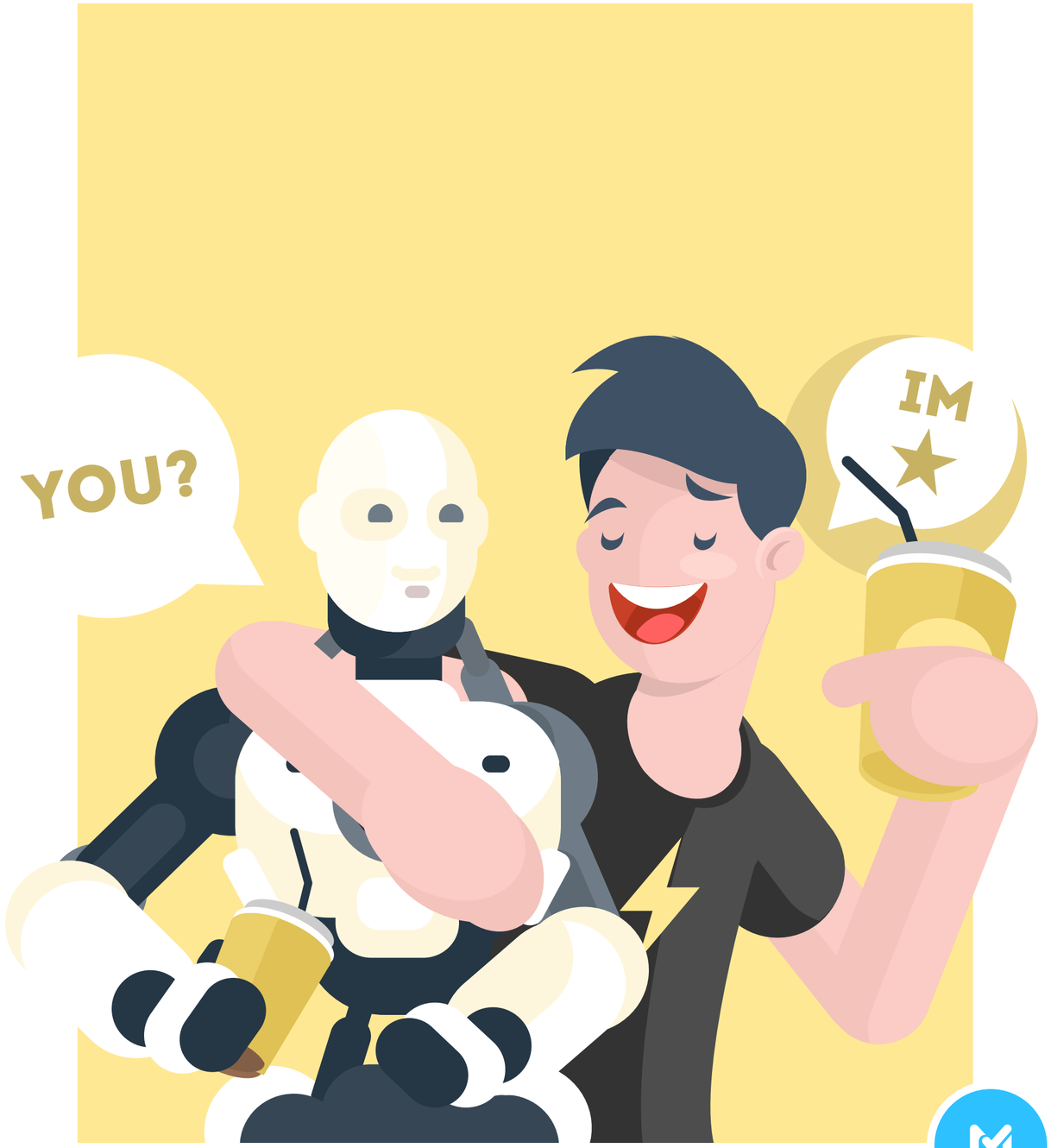
RIGHT BALANCE FOR QA TEAMS

How Testers can embrace AI?

AI has proven ability to function with more collective intelligence, speed and scale than even the best funded app teams of today. With Continuous Development setting an ever aggressive pace, and the combined pressure from AI inspired automation, robots, and chatbots, it begs the question that looms in the mind of most likely all software testers: are testing and QA teams under siege? Are QA roles in jeopardy of being phased out or replaced similar to the manufacturing industry?

Over the past decade technologies have evolved drastically, there have been so many changes happening in the technology space but one thing constant is human testers interaction with them and how we use them for our needs. The same holds true for AI as well. Secondly, to train the AI, we need good input/output combinations (which we call as training dataset). So to work with modern software we need to choose this training dataset carefully as the AI starts learning from this and starts creating relationships based on what we give to it. Also, it is important to monitor how the AI is learning as we give different training datasets. This is going to be vital to how the software is going to be tested as well. We would still need human involvement in training the AI.

Finally, it is important to ensure while working with AI the security, privacy and ethical aspects of the software are not compromised. All these factors contribute to better testability of the software. We need humans for this too.



In summary, we will continue to do exploratory testing manually but will use AI to automate processes while we do this exploration. It is just like automation tools which do not replace manual testing but complements it.

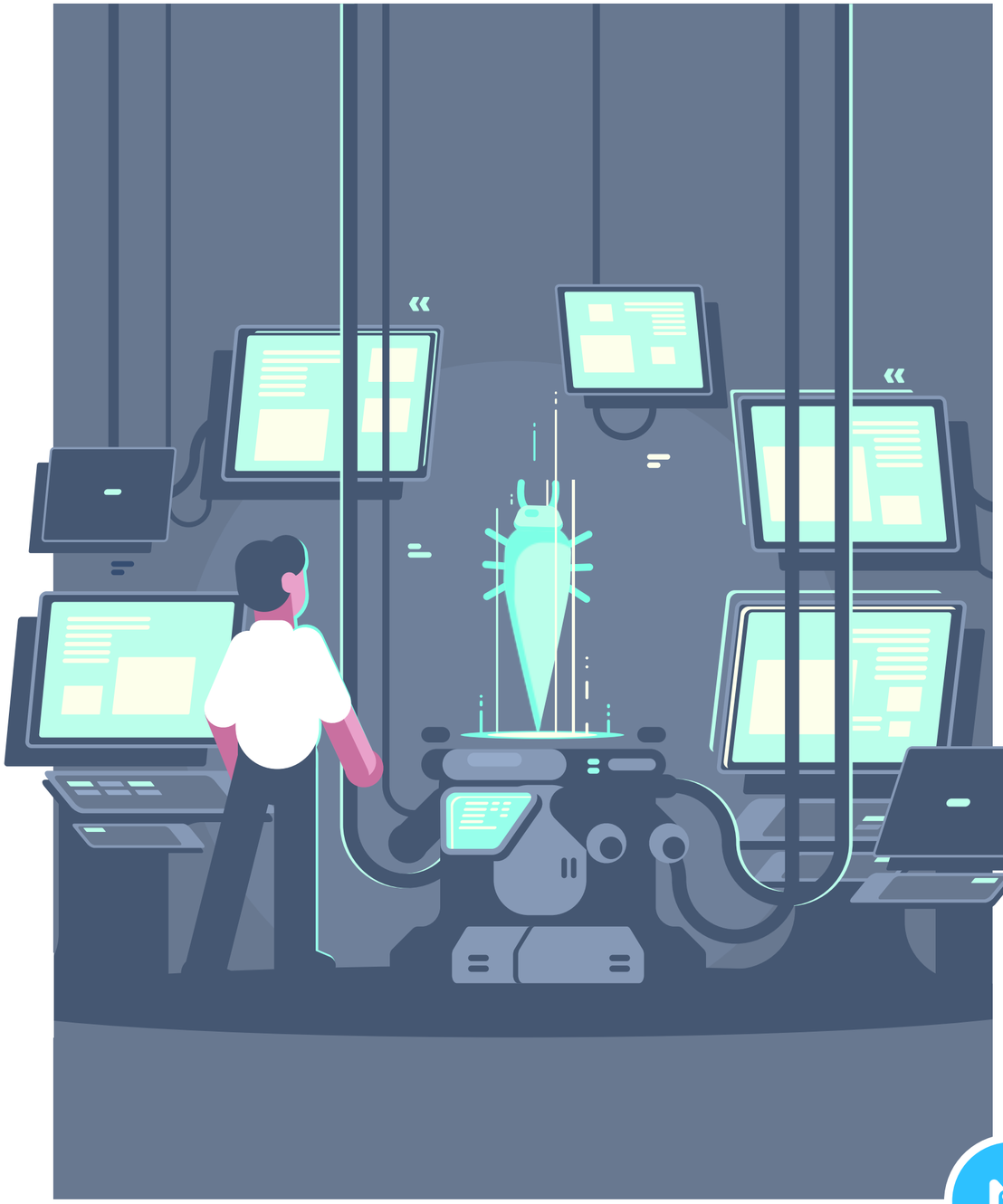
So, contrary to popular beliefs, the outlook is not all 'doom-and-gloom;' being a real, live human does have its advantages. For instance, human testers can improvise and test without written specifications, differentiate clarity from confusion, and sense when the 'look and feel' of an on-screen component is 'off' or wrong. Complete replacement of manual testers will only happen when AI exceeds those unique qualities of human intellect. There are a myriad of areas that will require in-depth testing to ensure safety, security, and accuracy of all the data-driven technology and apps being created on a daily basis. In this regard, utilizing AI for software testing is still in its infancy with the potential for monumental impact.

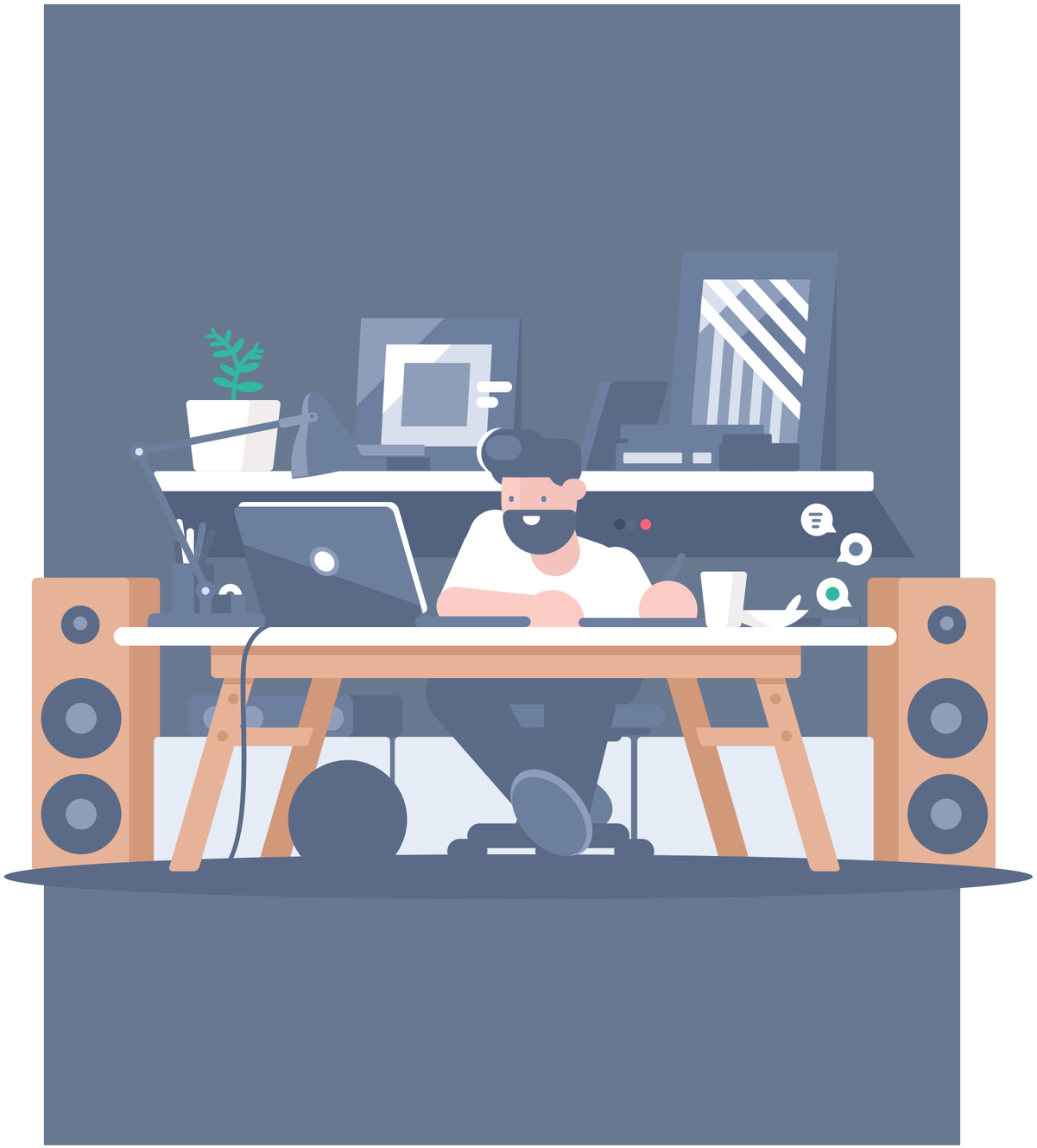
Future of Test Automation

We have discussed various ways, AI will influence the field of software testing and help to solve some of the biggest challenges with test automation. In the future, the way we do test automation is going to significantly change in terms of taking a more risk based approach to software testing.

AI has the ability to learn from different user flows and create test cases based on actual user data. We no longer have to spend a lot of time to create test data based on production users since the AI is doing it automatically for us. This helps to increase test coverage and makes the automated tests much more effective as it has been created based on real user flows.

Replacing static locators with dynamic locators is going to make the tests more stable and as a result of this the authoring and execution of tests is going to be much faster. For example - Say we have a “Checkout” button and change its name to “Buy”, the tests that would have failed earlier due to the use of a single attribute to locate the element will no longer fail as we will start using dynamic locators and make use of multiple attributes for the same element. Where in, even if the name of the button changed, the AI will go to the next best attribute to locate the element on the page instead of failing the tests. This saves considerable amount of time maintaining the automated tests.





Also, the more tests the user runs, the more data the AI collects about the test and the more stable the tests become over a period of time. For example - based on the number of test runs the AI can start optimizing the wait times in tests to accommodate different page load times in the application.

Finally, test automation is no more a developer focused task rather everyone in the team can participate in writing automated tests as the authoring and execution of test become really simple with the use of AI. Users can record tests on their own and also use the tests that are automatically created by the AI to create effective automated test suites. In this way, even non-technical people can author and write effective tests.

Conclusion

With the steady progress being made with AI, the truth remains that mimicking the human brain is no easy task. Humans are the users of apps and the technological innovations that are being created takes into account that, the human understanding, creativity, and contextualization are traits necessary to ensure a quality product. That said, manual testing remains essential and should compliment automation and AI. They are distinct and different functions that, instead of being compared, should be leveraged according to their respective strengths. Rather than AI solutions replacing QA teams, AI can augment software testing and infuse testers with super human-like efficiency.

What's clear is that leaders in the technology industry will continue to dissolve boundaries, discover and innovate with machine learning and AI. As QA teams continue to embrace automation and welcome AI into their software testing practices, the outcomes will contribute to new solutions and ways of working, reinventing what’s possible.

About the author

Raj Subramanian is a former developer who moved to testing to focus on his passion. Raj currently works as a Developer Evangelist for Testim.io, that provides stable self-healing AI based test automation to enterprises such as Netapp, Swisscom, Wix and Autodesk. He also provides mobile training and consulting for different clients. He actively contributes to the testing community by speaking at conferences, writing articles, blogging, making videos on his youtube channel and being directly involved in various testing-related activities.

He currently resides in Chicago and can be reached at raj@testim.io and on twitter at @epsilon11.
He actively blogs on www.testim.io and his website www.rajsubra.com. His videos on testing, leadership and productivity can be found here

Thank You!

www.testim.io