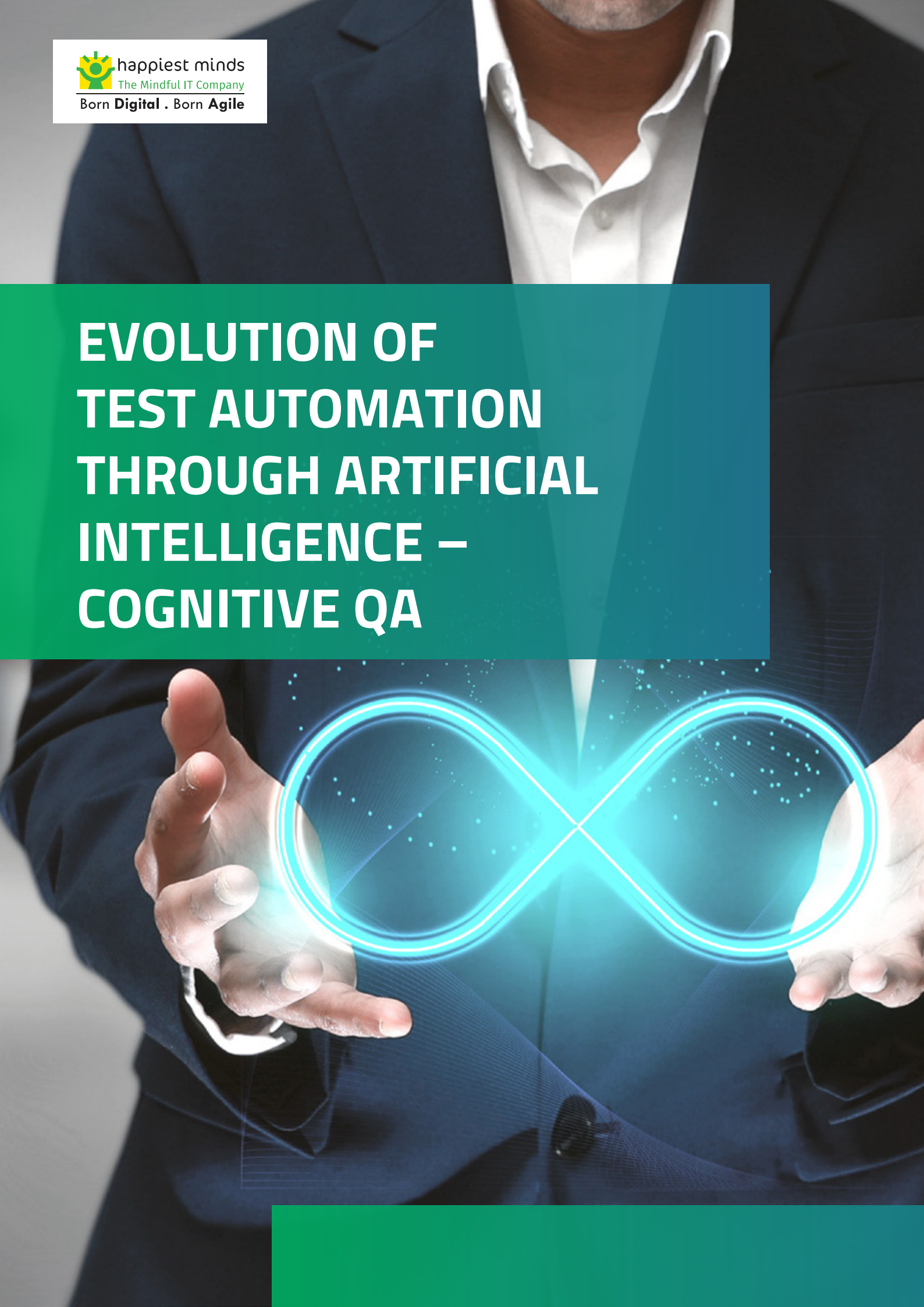
happiest minds
The Mindful IT Company
Born Digital . Born Agile

EVOLUTION OF TEST AUTOMATION THROUGH ARTIFICIAL INTELLIGENCE – COGNITIVE QA



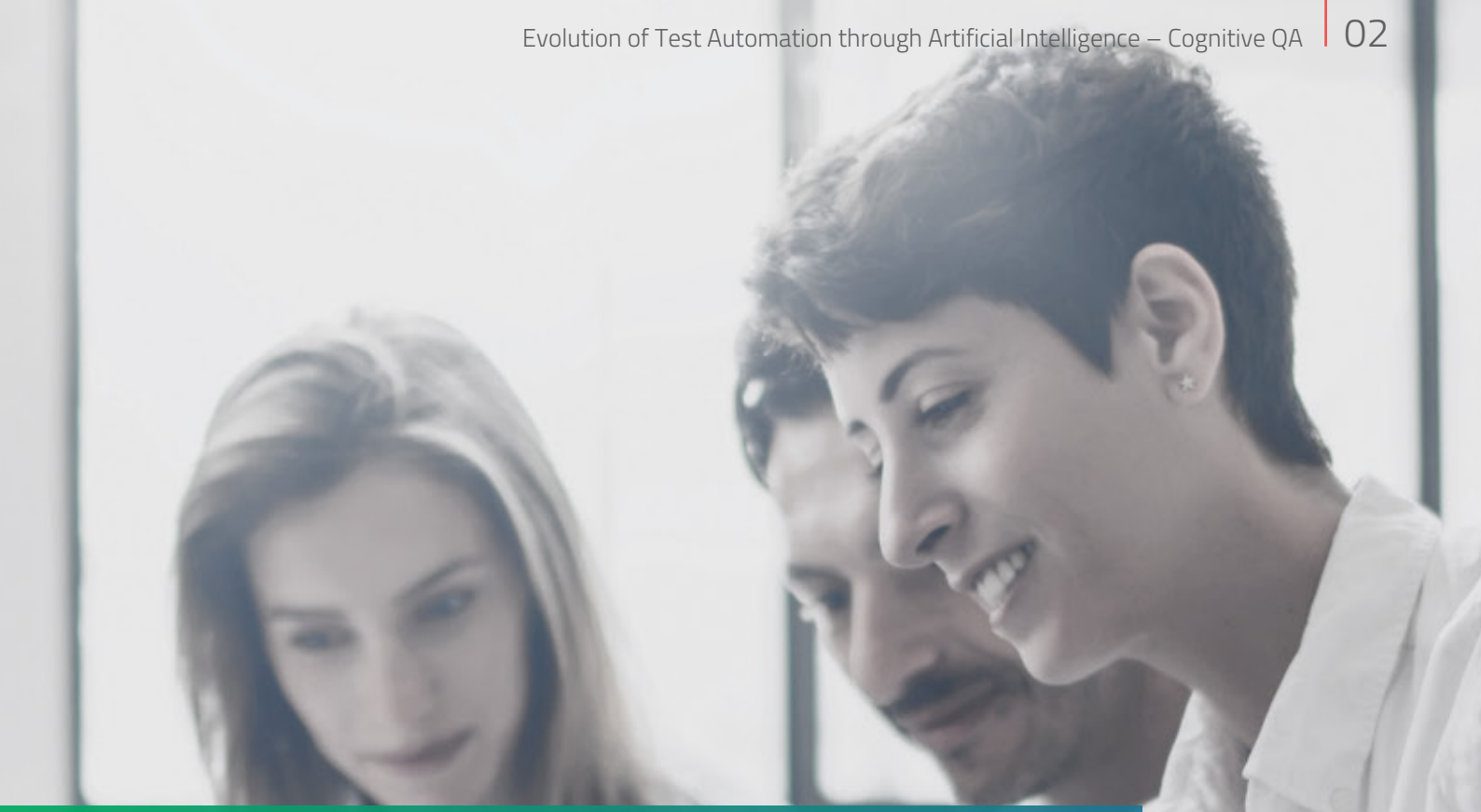


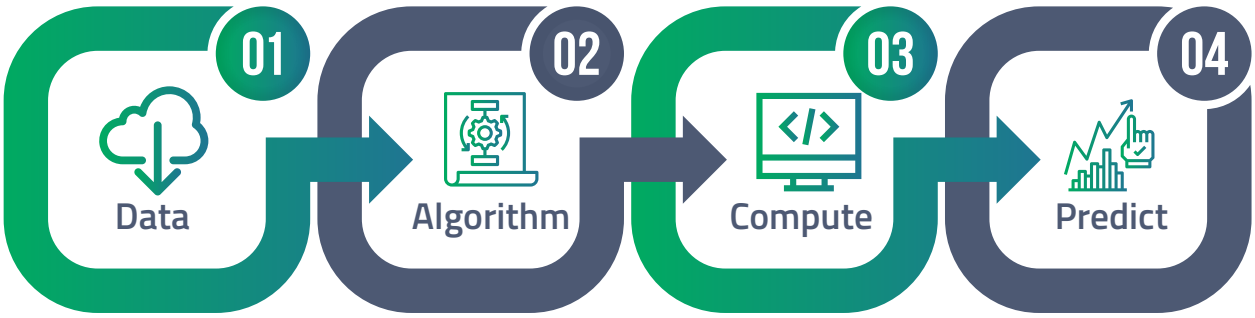
TABLE OF CONTENTS

Introduction

The advent of Industry 4.0 with its niche and sophisticated Deep Learning Algorithms and Technology has enabled a Digital (Automation) revolution across sectors, and Software Testing is not left out as well- Test automation is becoming an integral part of every project replacing Manual methods of Testing. Test Automation was used to ease out the Manual Regression testing effort to a large extent. However, on completion and client delivery of test automation projects, they are sparingly used due to associated high maintenance efforts and Technical knowledge, which resulted in wasted effort and cost. The limited-time span provided for delivering software projects constitutes a plethora of challenges for testing teams, one of which is Project delivery cycles, and it is important for the testing process to evolve and accommodate/overcome these challenges Test Automation has become a pivotal activity to enable teams to achieve it. With the speed of Project delivery cycles, Test Automation has become a key activity to match the pace. In traditional automation, considerable amount of time is required to identify the test scenarios, Develop automation scripts and finally the maintenance of the scripts. Considering all these activities, it is extremely challenging to ensure maximum coverage of the tests and hence to tackle these pain points, the ideal solution would be to bring Artificial Intelligence (AI) into Test Automation.

Test automation using AI has become a buzz of phrase with even large Enterprises such as Google, Amazon talking about Artificial intelligence getting into their mainstream platforms. Despite the existence of AI for over a decade, we are seeing this buzz word used a lot more now, with several platforms or applications leveraging AI and Machine Learning element.
AI-based recommendation engines with the auto-correction features and Google search - suggestions based on recent searches we made, are classic examples about the usage of AI. The AI is already available, but we are trying to make it available even for Software Testing community as AI-powered/ enabled Automations tools & platforms.

This is clearly a big jump forward for software testing. Testing automation revolutionized the process of delivering greater efficiency, scale, and faster time to market with improved coverage. Now, with the insights generated by AI and ML algorithms, it is possible to up the ante by optimizing and continually improving the software development lifecycle, making smarter decisions based on prescriptive analytics.

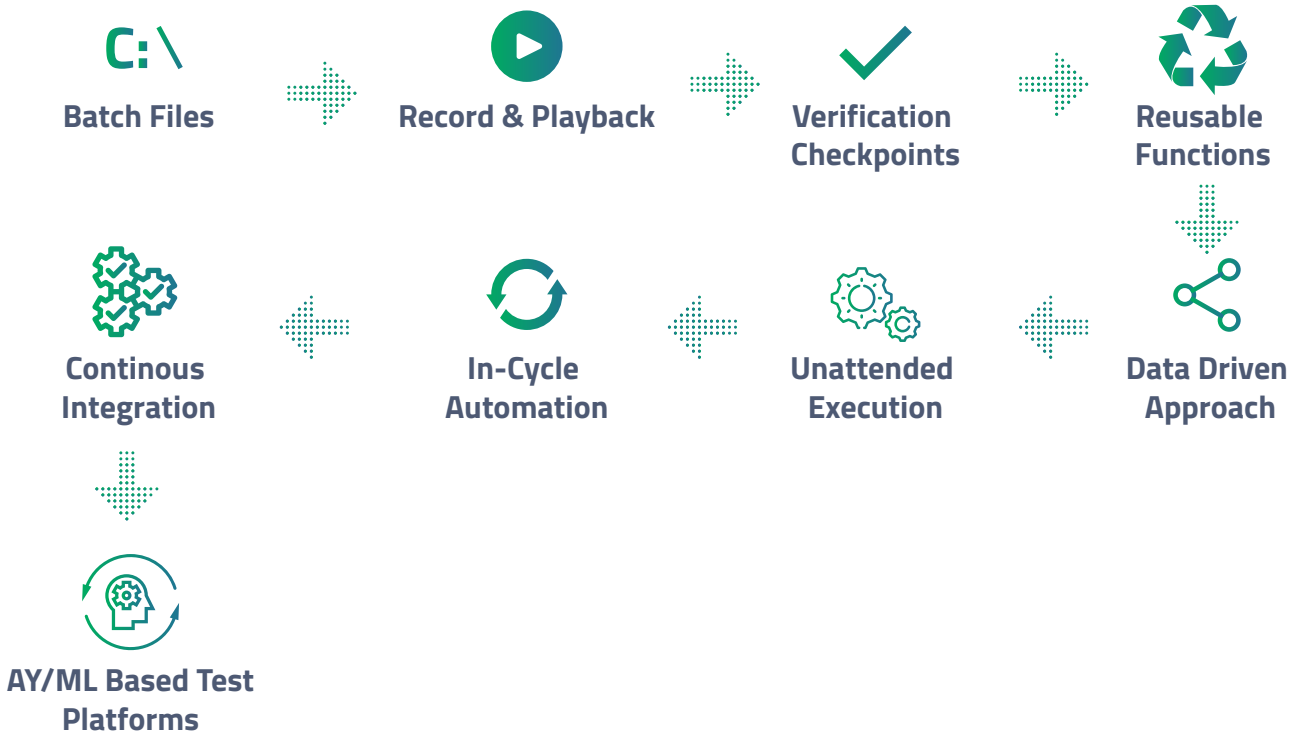


As part of Cognitive QA, we can leverage Artificial Intelligence (AI) and Machine Learning (ML) to achieve higher test automation. These technologies can help in automatic generation of test scripts, covering high-risk test scenarios, achieving higher test coverage, and the most important aspect being Self-healing of the automated scripts which is one of the key factors for a Successful Test Automation Implementation.

The below sections articulate how AI can be used, not only to enhance the Test Automation exercise, but it could even automate the test automation process itself such as auto generation of test scripts, Test data generation, Analyzing test results & Maintenance.

Evolution of Test Automation

The below sections explain the evolution of test automation over the years and how different technologies and frameworks have helped the test automation engineers. It also details out, how the evolution of AI/ML-based testing frameworks and tools has come to the forefront in the world of software testing. It specifies how AI/ML-based testing tools are going to transform and change the life of automation test engineers in the same way how automation had changed the life of manual testers over the last decade.



Evolution of Test Automation over the years

The above diagram represents the evolution of the test automation process and thought process over a period. Test Automation has progressed from writing simple batch scripts to record & playback tools to unattended Software Robots executing a task and finally AI/ML-based smart test automation tools

Batch Files

This was used to trigger a few commands to achieve a specific purpose for performing basic automation using batch scripts and Shell scripts

Record & Playback Tools

The Record & Playback tools were capable of recording & playing back tests. They were mainly used for repeatability, and tools such as WinRunner and Silk Tests were used to achieve this task.

Verification/Checkpoints

Testing tools were used to define checkpoints to validate the expected results vs actual results. Tools such as UFT, Test Complete, and many other testing platforms were extensively used to accomplish this activity.

Reusable Functions

In subsequent years, reusable test frameworks were built wherein test scripts were modularized, and tests were broken down into multiple reusable functions, making it quite easy to apply changes by updating a function. This resulted in significant effort reduction in the maintenance of test scripts.

Data Driven Frameworks

Data-driven automation frameworks allowed the same scripts to be executed with multiple sets of data, and the scripts were made reusable through a concept called Parametrization. E.g., A login test case could be validated by passing multiple sets of data to the "Username" and "Password" fields to make sure that the feature was tested with all combinations of test data which resulted in improving test coverage.

In Cycle Automaton & Continuous Testing

As automation frameworks evolved, unattended execution, in cycle execution & Continuous Testing, became prominent where test scripts were supposed to run unattended (without human intervention). The scripts were designed in the same sprint cycle to leverage the maximum benefit of regression testing. The scripts were integrated with build tools wherein a CI/CD pipeline was created, and once the builds were deployed, automated smoke tests were triggered. The code would get deployed to UAT or the production environment once the automated smoke/regression scripts were passed.

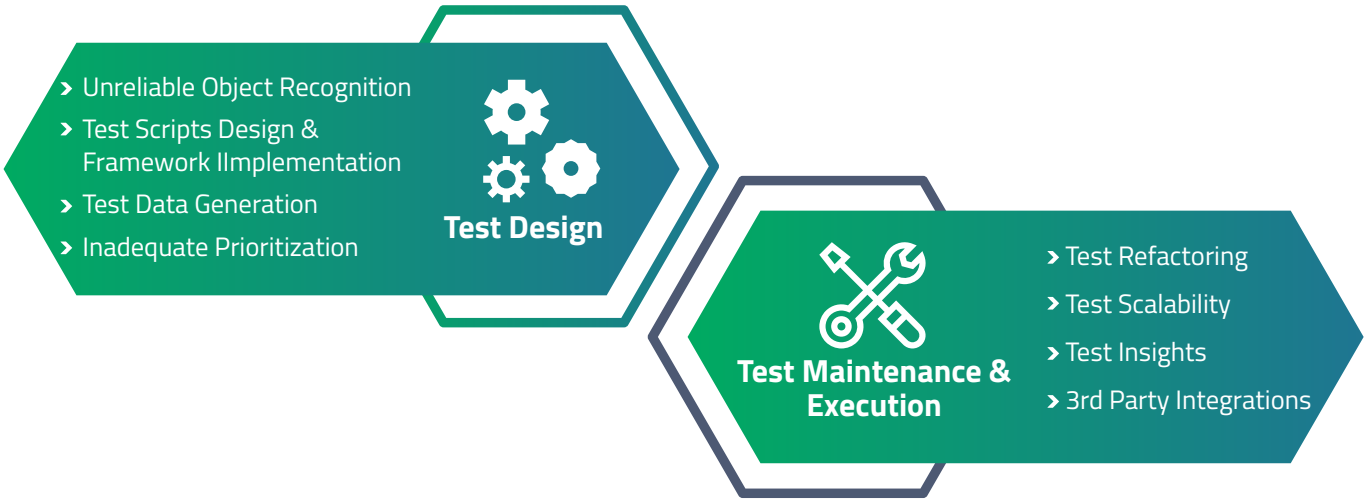
AI/ML based Testing Platforms

Finally, in the era of Cognitive QA, AI/ML algorithms were infused into the test automation tools. Test platforms have become smart and intelligent. These transformations naturally have an impact on the quality assurance operations of a company. Today there is a need for an end to end testing platforms comprising of UI, API's, Performance and other areas to be smarter and more intelligent to address the various nuances in problems faced by traditional test automation.

The below sections cover the various challenges in traditional automation and how these challenges are being addressed through AI-based test tools. It also covers the various techniques and processes required to address the challenges faced by test automation engineers and finally, the salient benefits associated with it.

Challenges in Test Automation

The below snapshot highlights a few challenges in test automation and how AI-based test tools are leveraged to address these challenges.



Unreliable Object Recognition

Object recognition is the foundation of UI automation. Traditional automation tools and frameworks work based on the locators of these objects in a webpage. It can be ID, Name, XPath, or CSS. When these properties of the object are tweaked and changed by the development team, the traditional test automation tools may no longer be able to recognize them, and the scripts tend to fail. Manual identification of these objects takes a long time, and it can be very cumbersome to update the object repository. These interruptions might slow down the entire test automation process, thereby impacting the software delivery life cycle.

Test Scripts Design & Framework Implementation

Designing automation test scripts is a key area in test automation. Designing test scripts requires technical skills and this may not be easily performed by non-technical folks. Also, some of the key factors like re-usability, parameterization, test data management might not be addressed. Artificial Intelligence and Machine Learning eliminate these dependencies and will generate automated scripts by reading English test cases from spreadsheets. The platforms use Natural Language Processing (NLP) to read the manual tests and auto-generate the scripts. In a nutshell, if the intent of the test step is right, the AI platforms generate automated scripts that can be run by any individual in the project teams.

Test Data Generation

Test data being a critical part of testing, there's a need to ensure that enough and valid data is available during testing. At times, due to the unavailability of data sets for testing negative and corner case scenarios, comprehensive testing of applications is not performed, hence reducing the quality of releases. This is usually due to the huge time and efforts required to create such exhaustive data sets. The AI test platform's data generation feature helps users generate such data from defined patterns provided by the users. Users can also choose to provide seed data, and the platform will autonomously identify the relationships between different fields in the provided data and generate additional data appropriately which will improve the overall test coverage.

Inadequate Test Prioritization

Test case prioritization is another interesting challenge faced by project teams wherein QA teams are not quite sure about the minimum number of test scripts to be run for a given code fix. Generally, test engineers pick up smoke/regression tests cases based on their experience, and probably, a few guesses are made. In fact, many companies are using AI tools which does that. Machine Learning can help in this -it can spell out with precision, the smallest number of tests that need to be run to test a piece of changed code. In the future, the tools would be able to offer recommendations based on the information they collect to identify tests that frequently fails, what piece of code is written by a developer, if the developer is more likely to develop buggy code, etc. Overall, AI will help the testing teams to run tests that matter and not the entire Regression test suites. It will help project teams to optimize test execution effort.

Test Refactoring

This is a huge problem in the UI automation space. Tests that are stable for a few days fail due to change in locators of the objects, wait time issues, etc. The automated test scripts must undergo periodic maintenance to overcome these issues, and the activity consumes a lot of time in the overall test automation activity.

Test Scalability

Test scalability is another issue that requires careful thinking. On average, it takes about 30-40 mins to create a robust Automation script. If we multiply that by the number of devices and configurations, the activity requires a considerable amount of time and effort. There are quite a few areas that organizations need to reconsider. How scalable is our automation suite? Can we run more automation scripts? Can we add more functionalities to the automation suite? Can we extend the functionalities? Can we run tests in parallel? - With AI testing tools, we can create automated Web-based tests that can recommend the type of tests that users need to run, provision the environment, run the scripts on relevant browsers and mobile devices on cloud-based test platforms when needed.

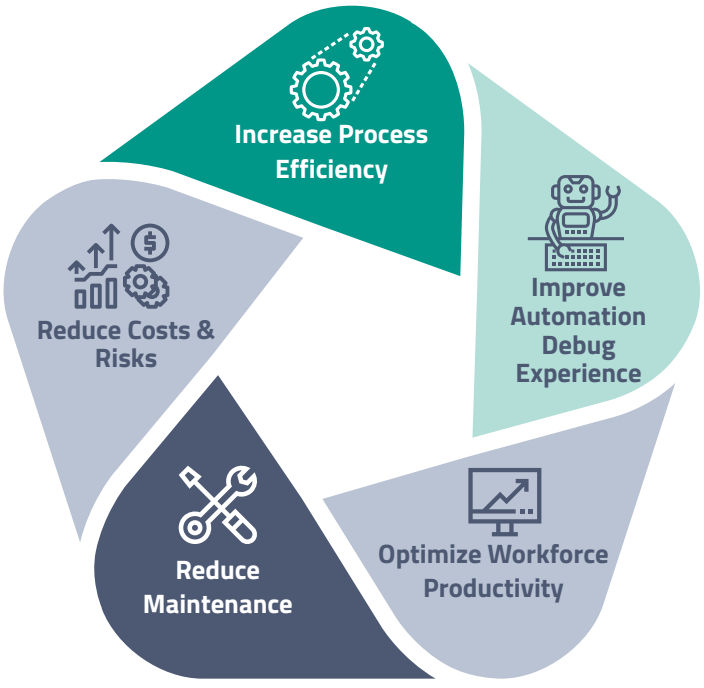
Addressing Challenges in Test Automation through AI/ML Approaches

As articulated in the previous section, The overall testing effort and coverage can be improved by infusing smart and intelligent test automation tools to address pain points in traditional test automation.

The below sections capture and articulate how smart automation techniques help project teams to reduce the testing effort and improve test coverage. In the below section, we have picked up a few pertinent pain points faced by test automation teams and elaborated them in detail, articulating the various key advantages when AI is infused into the test automation solution.

Self-Healing

The self-healing technique in test automation is a solution that solves major pain points such as test maintenance where automation scripts break whenever there is a change in the object property (Name, ID, Xpath, CSS etc.). Manual identification of these object changes and updating them in the object repository can take away a lot of time, adding up to the overall testing time and effort. There is a need for programs to automatically detect these changes and fix them dynamically without human intervention. This concept is called as Dynamic Location Strategy.



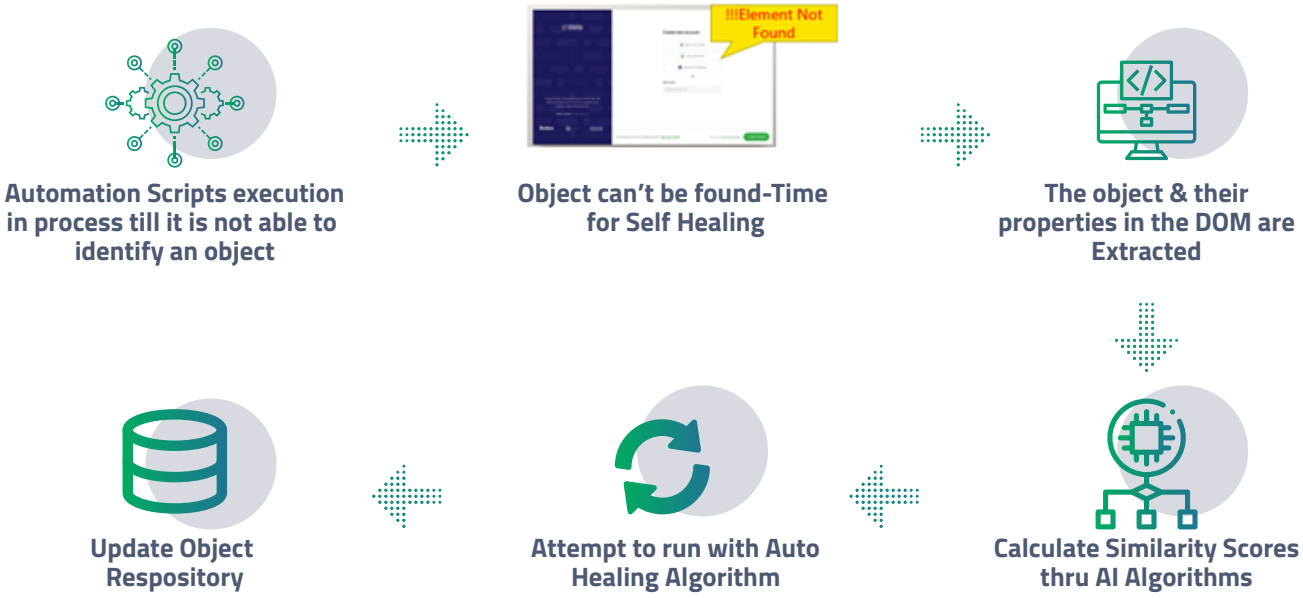
Advantages of Self-Healing in Test Automation

This vast change with respect to self-healing changes the overall approach to test automation. This solution helps the project teams to embrace Shift- Left approach in the Agile Methodology and the overall testing process will be more efficient with increased productivity and faster delivery.

With the self-healing feature, the UI identifier in the test case is automatically corrected whenever the developer changes any of the object identifiers in the HTML page. The AI engine is smart enough to locate the element despite the change in the attribute and then modify it to match the change made in the source code.

Self-healing saves the tester's time and effort by identifying the change and updating the test cases for every small change in the UI.

The process flow below depicts the overall end -to -end workflow of the self-healing technique which is handled by artificial intelligence based test platforms.



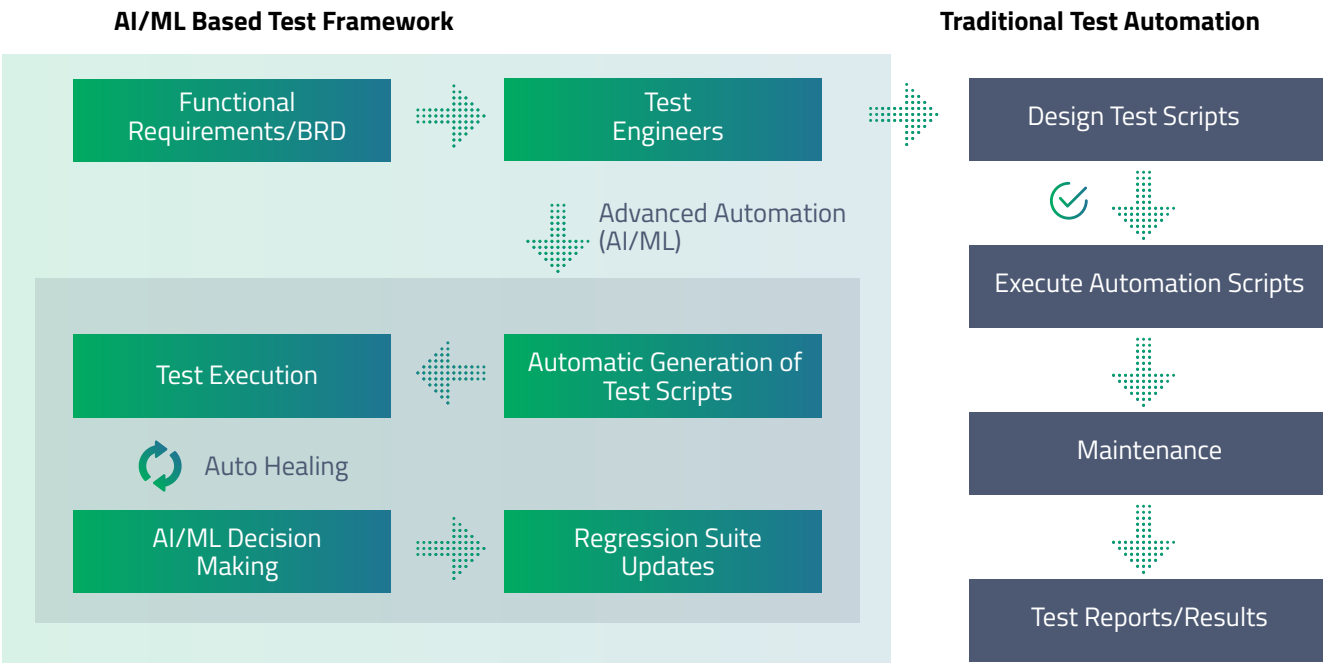
Self- Healing of Automation Test Scripts - Process Flow

In this process flow, the moment the AI engine figures out that the object property has changed, and tests might break, it extracts the complete DOM and studies the object properties. Based on weighted scoring systems, it extracts the updated property using a dynamic location strategy for the object. It runs the test scripts in a seamless manner without even the user getting to know about the changes made to the object using Dynamic Location Strategy.

Auto Generation of Test Scripts

The design of automation test scripts is a cumbersome activity in any automation project. Here, scripts must be developed using programming languages such as Java, Python, and Ruby. The entire test development project requires a lot of initial effort, time, and highly skilled people.

Automation script development takes away at least 50% of the total automation effort. For example, designing a medium complexity test case with 10 -15 steps, takes about two hours, including unit testing and running the scripts as a suite. With this, the call for infusing artificial intelligence and machine learning techniques grew louder to ease out the test script design effort.



Workflow – Auto generation of Test Scripts

There are test tools available in the market, where Selenium Automation test scripts are generated by feeding the manual test cases (English test cases). The platform automatically reads the test steps and generates automation scripts. The AI algorithm uses NLP (Natural Language Processing) which can understand the intent of the user and able to mimic those actions on the Web application. This is achieved without the test engineer having to write any code for automating the workflows. It reduces the test script design time and effort by over 70%., What used to take 1-2 hours for automating a single test case, can be achieved in minutes. The concept is referred to as Touchless Testing

Future Use Cases

This section highlights a few interesting scenarios where NLP and AI/ML engine could be leveraged in AI-based testing tools in the future:

Test Recommendation Engine

AI can help determine which test cases need to be run if there are any code changes to the application. The AI engine should identify tests that are relevant to the code fix and run only those scripts instead of the entire test suite. This will help project teams to avoid running the entire test suite for any small change in the application. For this use case, machine learning algorithms can be used to identify the patterns and make suitable decisions. The solution should provide the minimum tests to run for a given change made in the code. This will significantly help testing teams to reduce their manual activity in identifying the smoke/regression tests to be run for every bug fix or release and take a risk-based approach with true data-driven decisions.

Learning by Observation

AI can be leveraged by looking at common patterns that are performed by users and then trying to author test scripts based on user actions. For example, in a typical login scenario, where a user has logged in 15 times and logged out 15 times during scripts execution, & based on this execution pattern, the AI and ML engine can automatically learn the pattern and creates a re-usable login component automatically to be used across multiple test scripts, which significantly reduces the scripting effort. It will also help generate re-usable components by traversing across multiple user journeys in the application and continuously learning user patterns.

Predictive Analysis

This is another area where AI/ML will be able to predict the probability of a build failing when code changes are made to the application. A self-learning system will have the ability to make predictions based on historical data and will be able to create new test scripts and update the existing automation scripts as well. This will help project teams release builds faster to production.

Continuous Learning from Production Data

AI algorithms can create tests based on production data or real data. This will help project teams observe & figure out various user actions that are commonly performed in the application in a production environment. The AI algorithm can identify these actions that are frequently performed by users and can have them clustered into reusable components. This will help testing teams identify critical workflows, thereby increasing test coverage. These components can be used later within other tests.

Handling Response Time

During UI automation, at times, a page takes 3 seconds to load, and in some cases, it takes 10 seconds. With this insight, the tool keeps learning the behavioral patterns of the application & finally arrives at a conclusion & automatically sets an optimal wait time for the automation scripts. The AI engine will also help the tool record and capture server responses beforehand so that scripts need not always have to hit servers to get the response data. This, in turn, will reduce overall test execution time.

Benefits from AI-based Test Automation Frameworks

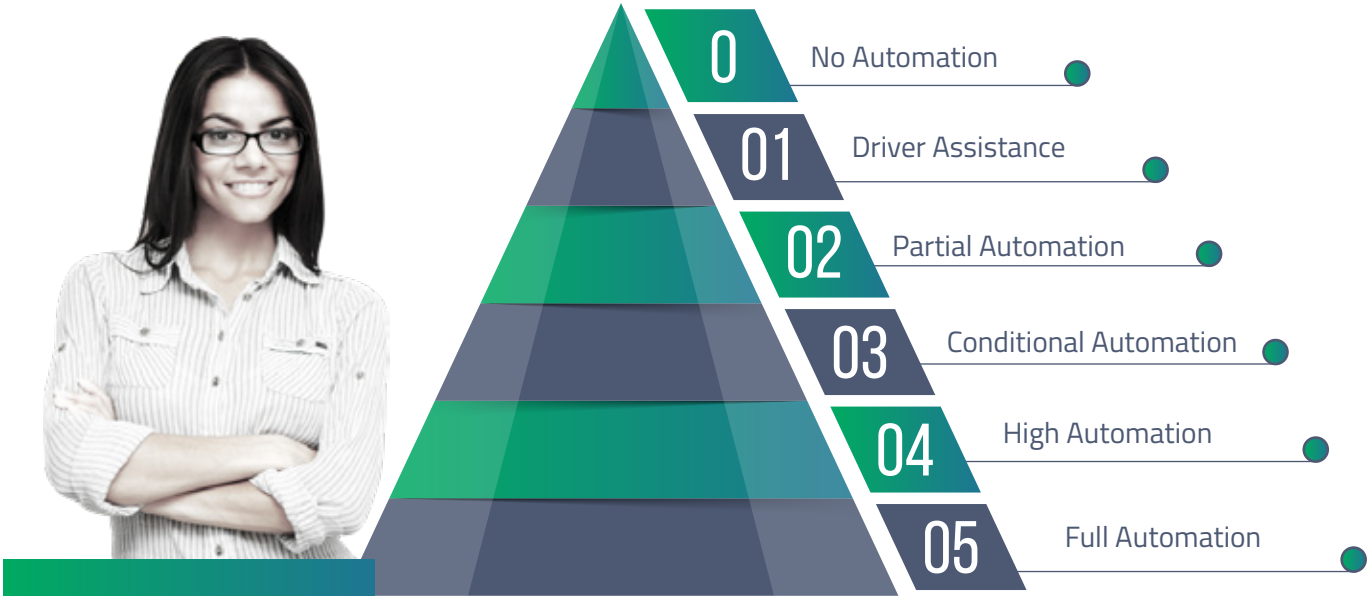
The below points articulate the various benefits that teams can derive by making automated testing tools smart and intelligent:

Automated generation of test cases and test scripts, using machine learning	Codeless automation	Faster and stable UI tests
Reduced script maintenance effort using auto healing	Increase in test coverage	Visual identification of Flaws in the application
Intelligent trace logging to analyze the test flow and failures	Regression/Smoke test case recommendations based on historical data	AI testing tools to reduce testing cycle time to a large extent
Smart QA dashboards	Defect prediction based on historical data and failures	Better execution history maintenance and metrics availability
Test insights and real time diagnostics		

Here are the salient benefits of infusing AI/ML into Test Automation:

Improved accuracy	Going beyond the limitations of Manual testing and traditional automation	Increase in overall test coverage
Ease of use for non-technical folks	Faster time to market	Improved flexibility

The below snapshot gives us an insight into the progress from traditional automation to full-fledged automation, including the testing processes.



Conclusion

To conclude, the efficiency and effectiveness of test automation are extremely crucial for the success of the project. The QA process is increasingly becoming more complex and vast. The urgency for faster delivery is also increasing day by day. Today, there is a need for end-to-end product testing, but the timeframe is very small, which makes it a challenge to create test cases that cover all the crucial & critical test scenarios. AI/ML testing tools that can tackle these issues have become popular in the market, and the management is going the extra mile and allocating the necessary budget to incorporate the tools/platforms as part of their testing process.

About the Author

Prasanth TV *is a Sr Test Architect at Happiest Minds with over 14 years of experience in the area of Software Testing & Quality Assurance and leads the QA practice within DBS unit of Happiest minds. He has extensive experience in areas of Test Automation across multiple tools and technologies. He has worked on multiple domains including Airlines, Retail and Manufacturing. He also possesses wide experience in the consulting and pre-sales areas with a keen interest in emerging Digital Technologies.*