happiest minds
The Mindful IT Company
Born Digital . Born Agile

A GUIDE FOR DEVELOPING EFFECTIVE TEST AUTOMATION STRATEGIES & HOW TO AVOID PITFALLS

-A POINT OF VIEW

Introduction

Test automation is a "hot topic" for many project teams across software development projects. Test automation has great potential in achieving the required objectives of easing out the regression testing effort. More often than not, test automation projects, once delivered to the customer, get into cold storage and will be hardly used, which results in wasted efforts and cost.

According to the World Quality Report, only 14~18% of organizations have basic test automation tools in place. Choosing the right test automation tool is of utmost importance to reap long-term benefits from automation testing. The test automation strategy should be built in a scalable manner, and the ROI should not be judged on short-term success.
The sections below provide us insights into why test automation projects fail, mitigation plans, best Practices & the relevant success factors for test automation.

Business Benefits of Test Automation:

Test automation has great potential in achieving the required objectives of easing out the regression or repeated testing effort.

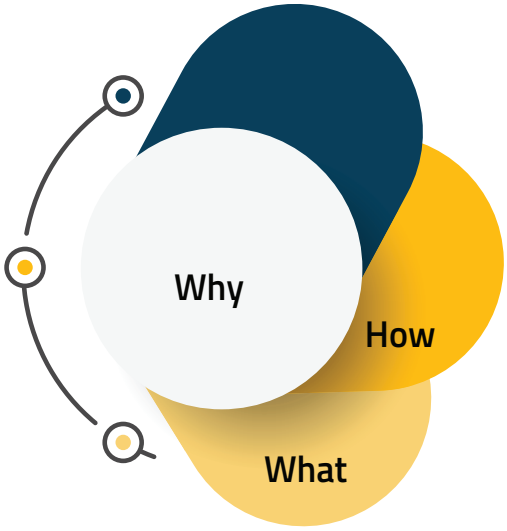
A few of the important benefits of performing test automation are listed below:

Automate the repetitive and execute mundane tasks and workflows	Higher test coverage	Faster feedback about the software results in early detection of defects
Reduce costs & time savings	Reduced regression testing time	Improved accuracy
Integration with CI/CD pipelines & DevOps implementation for continuous testing and faster feedback	Reusability of test functions which improves the overall maintainability of the tests	Fewer human resources

Test automation, after being projected as a savior to reduce testing effort & cost, thereby increasing the ROI, often fails to live up to the expectations.

Why-How-What of Test Automation

One of the main reasons for failure can be attributed to a greater focus on "how to do automation" rather than thoroughly addressing "why automation is needed."



In real terms, test automation is performed to facilitate a business need. But firstly, it is important to understand "Why automation" and then move on to the "How" and "What" aspects of the test automation exercise. The need for the entire automation might vary from increasing testing efficiency, increasing test coverage, discovering defects, and various other aspects.

Once the "Why" is clear and measurable, the "How" constitutes the planning and resources to achieve the "Why" and "What is the overall approach, tools & technologies, strategy, skill sets required that complies with the "How" This is an inside-outside approach of planning a test Automation strategy.

The promises of automation are many; however, many test automation initiatives fail to achieve them.

Some wrong notions that people have about test automation include:

We have an industry-leading tool that does everything at the click of a button	Automaton is easy – just record and playback	Any individual who is trained on a tool can automate
Test automation is a part-time job	Automate 100%, and it will provide us the ROI at the earliest	Automation is a replacement for manual testing
Time crunch, resource Crunch – Go for automation	Too many defects – Go for automation	Need to cut down on testing costs — Try automation
Everything has to be driven from the GUI of the application under test	A tester with tool knowledge can be an automation engineer.	

To make your test automation strategy successful, you need to avoid some common pitfalls & failures. With that in mind, here are some of the common problems with automation testing that you need to avoid for building a long-term successful test automation strategy.

Business Challenges During Test Automation

Automation testing can be a highly effective productivity booster and quality enhancer for your product pipeline or system development projects. Test automation can do wonders for businesses by reducing time, cost, and effort. But to get effective results, it is important to avoid the below challenges as they can lead to wastage of automation efforts. Following are some of the key slip-ups or judgmental errors which are typically made by project teams when kick-starting their automation journey:

01 Unrealistic Goals for Automation

02 Selecting a Wrong Automation Framework/Tool

03 Lack of Proper Planning

04 Ignoring False Positives & Negatives

05 Contaminated Test Environment

06 Underestimating Time, Effort & Cost

07 Test Script Maintenance

08 Bringing in the Wrong People for the Job

Unrealistic Goals for Automation:

There are unrealistic expectations regarding what test automation can and cannot deliver. The management or the project teams who have unrealistic expectations for automation are sometimes hoping for a "silver bullet" that will solve all of their problems. Unrealistic goals and expectations for test automation could create chaos in the entire automation process.

A clear-cut automation strategy is essential before we kick start test automation. We need to ask ourselves two basic questions-

1) When do projects require test automation
2) What are the goals of the automation activity?

For example, testers may believe that automated tests will "notice" things that a human being would notice, or developers may think that tests will "automate themselves." Testers are not needed to identify which tests are best to automate.

A few examples of Unrealistic Expectations are as follows:

- Every single (manual) test case can and should be automated.
- There is no need to spend time developing a framework - the project team has purchased a tool, now just get on with the automation activities.
- When automated tests pass, it means there are no defects in the software.
- Running an automated test for the first time can automatically give Return on Investment (even if it takes 10 times longer to automate the test than it would to run it manually).
- Any manual tester can write automated test cases directly in the tool scripting language.
- Maintenance of automated tests will not be needed or has a very small cost.

Selecting A Wrong Automation Framework/Tool:

Many project teams want to choose one tool for their entire enterprise. That's a blunder because there are many diverse problems you want to solve with automation, and one tool can't solve all of them. "The key is finding out what problem you want to solve with a tool, rather than buying a tool and then finding a problem for it."
Define specific problems you need to solve before searching for a tool and try before you buy. Before we finalize a tool, a two-week proof-of-concept trial with each tool runs it through the full development lifecycle to see how it works in your environment and to make sure it addresses the problem you are trying to solve.

There are several aspects we need to look for when choosing the test automation framework for their project.

Adaptability – This is one of the top criteria when we select a framework. It should be based on the complexity of the framework, programming language & learning curve involved.

Reusability – A key aspect of selecting a framework is the ability to re-use the same code when required. The framework should be modular in nature, and it should have components or functions which can be re-used for handling repetitive functionalities.

Reporting – Ability of the test automation tool/framework to generate detailed reports. The success of the designed test automation framework and its survival also depends on how effectively reporting mechanism will be implemented.

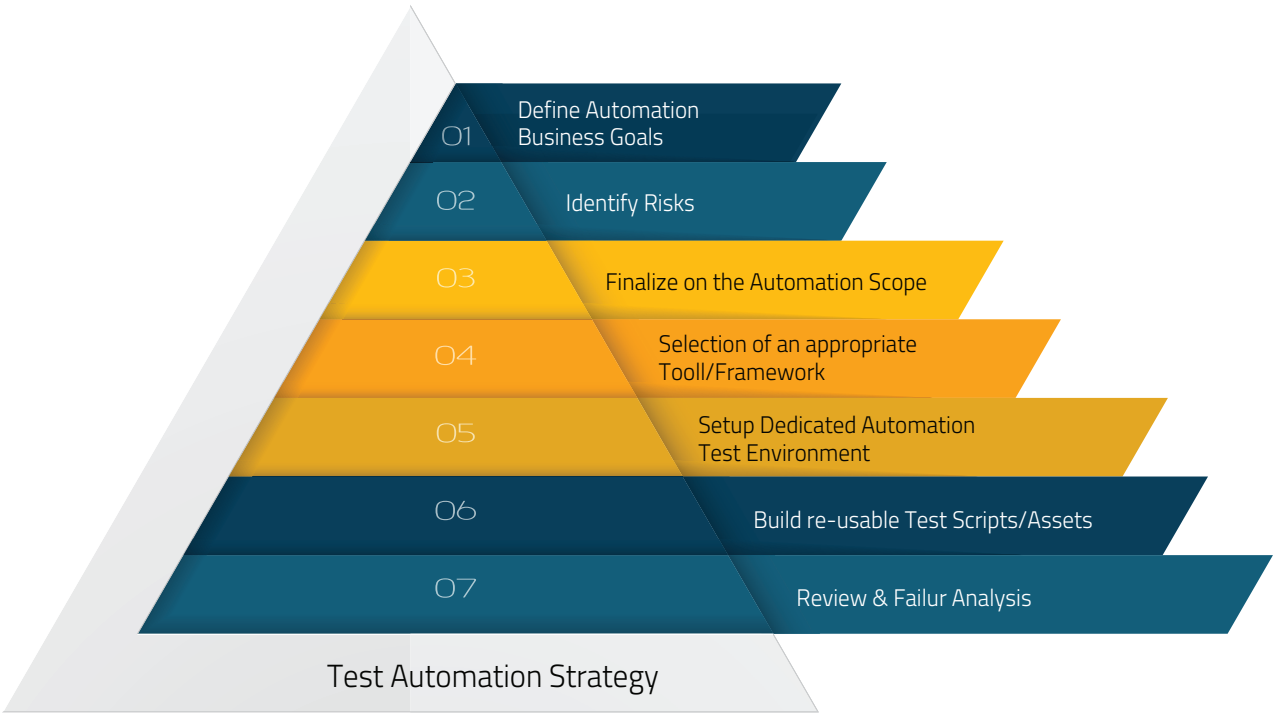
Integration – One of the key factors of a successful Automation implementation is the ability for seamless integration with CI/CD tools, Source code repository, Database Engines, etc.

Integrating automated test scripts into continuous delivery pipelines covers a crucial component, and being without that means things will fall short, or an organization may not be able to reap the full benefits of DevOps.

Lack of Proper Planning

One of the major pitfalls in test automation activity is the lack of planning and having a long-term vision of the expectations of the overall test automation. One of the major reasons for test automation not being very effective is the absence of test strategy and planning. One of the main reasons for failure can be attributed to more focus on "how to do automation" rather than thoroughly addressing "why automation is needed".

Below is a list of building blocks for a good Automation Strategy:



The next point of discussion is about the implementation of test automation. Few pertinent queries to be addressed: Who will be automating the tests? Do you have specialists on your team, or are your testers trained? Or do you want your developers to pitch in to write the automation test scripts? Is there a plan in place to train your testers to take up automation of scripts in the long run? Can you schedule training or onboarding for your testers to bring them up to pace with the new tool?

Depending on who is going to be allocated the test automation scripting task, you will need to select the tools and technology to be followed based on their expertise as well as on the needs of your system.

Whoever you pick to do the automation also needs to understand the business flows and scenarios that will need automation. There needs to be a process to hand over the tests to be automated to the right people with the relevant information and complete test data.

If testers themselves are doing the automation scripting, they must be given guidelines on which tests to pick up for automation, priorities, and sequencing, and a robust framework to which they can add to their test scripts and rerun them easily.

Ignoring False Positives & Negatives

When dealing with automation, one of its most delicate subjects is the results that lie, otherwise known as false positives or false negatives. Those who have already automated know this to be an issue, and for those who are about to begin, let us give you a fair warning that you will encounter this problem.

False Negative: When a test is executed, and despite it running correctly, the test tells us there is an error that is not really true. This adds a lot of costs, as the tester puts in a lot of effort in searching for the non-existent bug.

False Positive: When the execution of a test shows no faults even though there is a bug in the application. A false positive can lead to a bug leak in the market as it was pursued to be working well, and this can greatly impact the reputation and business loss of the company.

Some of the possible reasons for the occurrence of a False Negative/Positive are:

Incorrect Test Data	Duplicate Test Data	Improper Coding Standards
Missing Assertions/Checkpoints at every important stage of the Test case	Incorrect handling of Wait Statements	Initiating and Closing Browser Connections not managed correctly
Not handling Exceptions		

Contaminated Test Environment:

A well-planned and designed test environment provide us with the necessary compact platform to run our test cases. It is very important to have a dedicated test environment or testbed to run automated test scripts. The test environment includes more than just setting up a server to run tests on. It also involves hardware and network configurations. For example, let's say we want to test whether a specific function creates invoices for sales data that is present in a specific database. Since we need to have a database created to validate a certain workflow, The entire testbed and environment setup becomes very important. If the test framework has integrations with CI/CD tools or version controls systems like Jenkins/Maven, GitHub, etc. In that case, the entire testbed needs to be configured appropriately so that tests can be run overnight without any human intervention or dependencies.

Below is a list of requirements that are typically created as part of setting up test environments:

Configure test environment based on the tool/framework	Set up a database for test data management	Configure the network (e.g., cross-origin resource sharing
Select the right hardware and operating system	Integrations with other 3rd Party tools like Jenkins, Docker instances, GitHub, etc.	

Underestimating Time, Effort & Cost:

Management buy-in is absolutely necessary for a successful test automation project. The management team needs to understand the big picture, goals, and the ROI of the automation exercise before approving the effort and budget for the test automation project. The most common scenario is that the management team will not be aware of the advantages and nuances of the test automation process and approach. They would treat the automation work as another testing process and would not really understand that automation scripting requires specialized skills and experience based on the tool for the framework chosen to build them. Also, the team should be convinced that the entire automation effort would indeed bring down the regression testing effort with increased test coverage.

Development of the tests should begin as early as possible and in parallel to the feature and product development. As described earlier, a test automation project should not be treated like another regular testing exercise and instead treat it as an ongoing development activity. This will help the project teams to have the right skilled people deployed to the project.

Test Script Maintenance:

Automation is complex, and the maintenance of automated tests is more expensive than manual tests. People often do not realize the cost of maintenance of the automated test suite. As automation means code, the tests must be modified or recreated continually. Test automation maintenance is time spent rewriting or updating the test scripts. Test maintenance is required when the application undergoes a change that results in breaking the existing tests. For example, a UI design update moves the button that a test clicks on, and therefore the test fails, even if the functionality still works. Testers can write scripts that accommodate some degree of change. Design-agnostic locators, like IDs, solve the UI revamp problem. A good, automated test design minimizes the time spent on maintenance.

One of the pain points as part of the test automation exercise is the maintenance of the scripts because of the frequent change in locators in the application under test. Hard coding the locators in the script each time results in code duplication. Also, if the value of any single locator changes, it needs to be updated throughout the project to ensure consistency. This is a time-consuming task. Maintaining the object repository in a centralized location will solve the problem to a large extent. Incorporating industry standards frameworks like Page Object Model (POM), which is a design pattern in Selenium that creates an object repository for storing all web elements. It is useful in reducing code duplication and improves test case maintenance effort.

Bringing In the Wrong People For The Job:

Automated tests are programs. Automation of tests involves writing software, and this needs people with technical skills and good knowledge of programming. There are occasions when the people brought into developing automated tests do not come with any programming knowledge; perhaps we have worked with managers who suggested that a manual tester with limited programming knowledge could automate the tests. This will not work more often unless the manual tester also happens to be a great programmer.

Another reason this happens is that there may be a belief that automated tests can be generated, either procedurally or using machine learning. The effectiveness of this is so far limited to simple test cases. To have a comprehensive test suite in place, we will require handwritten code, and this requires people with specialized technical skillsets, and we need to have people with the right skillsets to do this activity. Force-fitting people for performing a certain activity might not work.

Best Practices & Success Pre-requisites for Test Automation:

The below pointers are a few of the best practices of test automation to ensure that the automation exercise is a successful one.

- Clear definition of the sole objectives of test automation
- Select the right test automation tool and frameworks
- Invest time in building a robust test automation framework that supports automation scripting effort and makes sure to use industry popular tools and technologies
- Automate only what is required and try to resist the temptation to automate 100%
- Have the right skilled people to work on the automation tool and invest in training them
- Stable & predictable environments are pre-requisite for successful test automation
- Constant maintenance of automation scripts for changes against the application under test
- Having a plan for how to analyze failing test cases and the corrective actions taken is critical

Build re-usable libraries and make sure that the entire automation scripts are modular in nature

Create good quality test data and make sure that none of the test data is hardcoded inside the scripts

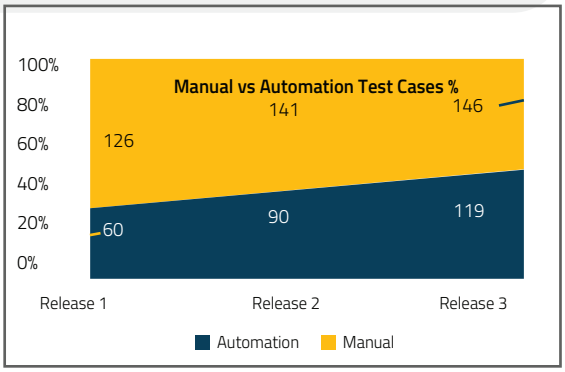
There should be a clear-cut measurement for the success of the test automation project (reduce regression time, increase test coverage for critical flows, improve ROI, etc.)

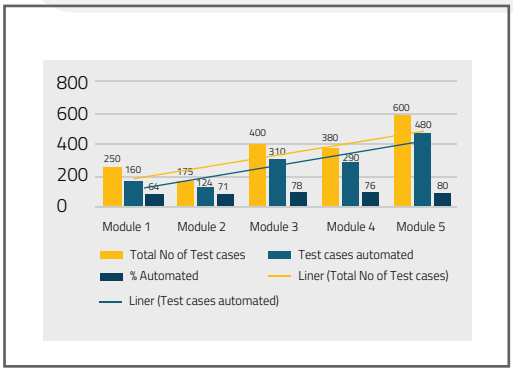
KPIs for Successful Test Automation

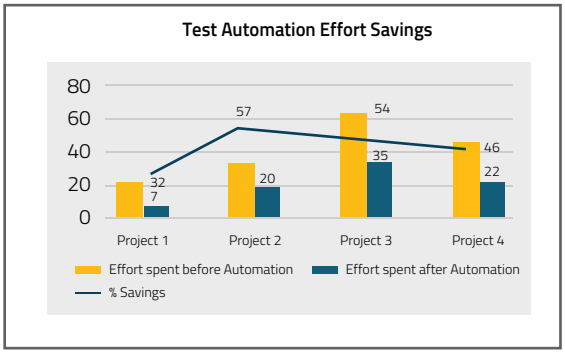
Typically, automated testing metrics quantify the performance of the implemented automated test process. These include past, present, and future efforts. When we talk of KPIs and metrics, we need clearly defined goals. Once you establish the goals clearly, metrics should be able to evaluate some of the following things. A few of the important metrics and KPIs which are mostly captured in projects are:

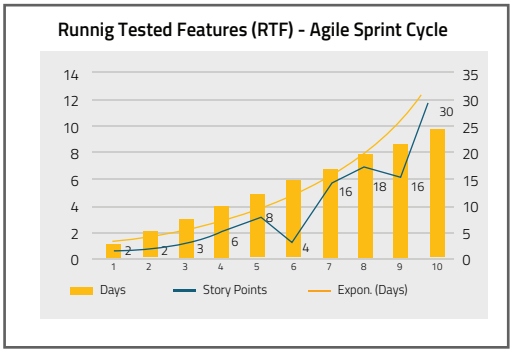
Automation Index- Percentage of Automatable Test cases: This metric will assist teams on the tests picked for automation, critical workflows which are being automated, and what areas will still require manual and exploratory testing.

Automation Progress: This KPI will help project teams to track the progress towards their automation goals against the defined timelines. This helps in establishing cadence for performing in-sprint automation.









Indicative Metrics (for representational purpose only)

Test Automation Coverage (%): This metric will help quantify the Test Automation Efforts and will provide the % of the total test cases automated against the Identified Test cases Test Automation Coverage (%) = (Total No. of tests automated/Total tests identified for automation) * 100

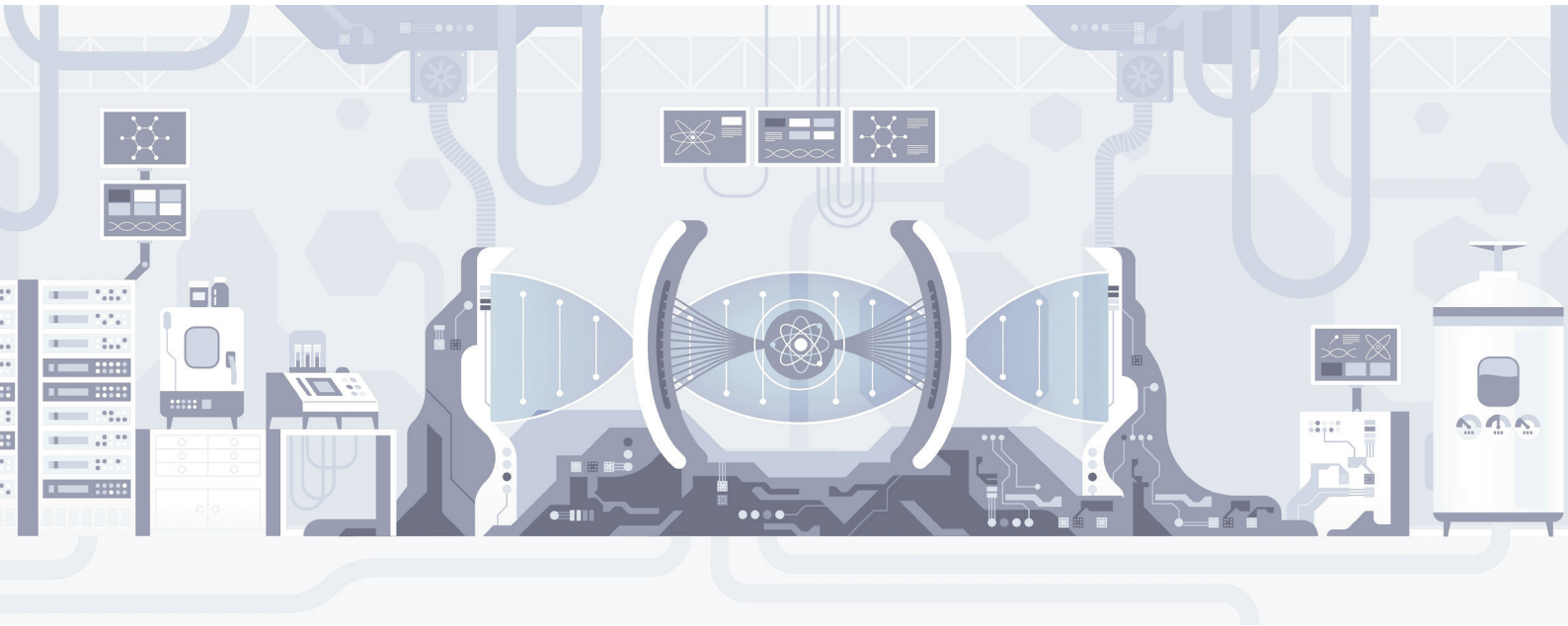
Equivalent Manual Test Effort (EMTE): This metric provides insights into the efforts it would have taken to execute the same automated tests manually. EMTE largely serves to answer such questions as, "Are we better off after this automation project than before we started?"
EMTE - Length of Time taken for Manual Testing - Length of Time taken for Automated Testing

Test Script Maintenance Effort: Automation Test script maintenance is one of the very critical and time-consuming activities which is performed on a regular basis. This effort is captured before the close of every Sprint or Release cycle to factor in the overall time required for Test Automation Maintenance.

Automated Smoke/Regression Tests Run-Time: This is one of the KPIs which is captured to measure the time taken to run the smoke/regression tests for better planning & analysis. This metric is provided by any CI/CD tools.

Industry Trends for Test Automation

The Covid-19 pandemic has severely impacted QA and IT strategy, forcing people to be more agile and achieve more with less. Quality of the software delivered remains a critical success factor globally, and with the rapid advancements in technology, the future of automation testing is also evolving. The emergence of new test automation techniques helps enhance the quality of software testing and ensures the delivery of flawless applications for an enhanced user experience. To meet these increasing demands, enterprises are under immense pressure to continuously adapt, improve, and deliver competitive customer-centric solutions.



Below are some of the Top Trends in Test Automation for the year 2022:

The rise of Artificial Intelligence and Machine Learning in Testing: By leveraging AI and Machine Learning, Project teams can speed up their release cycles. AI algorithms can be used across multiple testing areas such as:

- Utilizing algorithms to identify test cases that require automation
- Optimizing test suites by eradicating irrelevant test cases
- Automatically generate test scripts based on the requirements
- Smart Test Execution which automatically tests the tests that need to be run or a particular code check-in

Codeless Test Automation: As we all know about the booming IT industry, companies now want to get the maximum output done in no time. This enables the doors for codeless automation testing for software testers. These tools have been built with AI technology and allow test scripts to be developed and designed at a rapid pace. This means without any human intervention, the test cases for automation testing can be created as well as executed. The most significant advantage that AI-powered tools provide is – ease of maintenance of the test scripts. These have the ability to heal the test scripts because of fragile UI elements in the application. AI-powered tools can easily detect a change in the application and modify the script. With these tools, testers will no longer need to worry about test case maintenance.

QA-OPS: The digital world now requires a faster release of applications without any compromise in quality. QAOps is the combination of QA & DevOps. In the past, the role of QA has been restricted to application software testing, but now it is playing a crucial role in all phases of software development. It allows the software to be tested using the continuous integration (CI)/continuous delivery (CD) pipeline, rather than having the QA team operate separately.
Automated tests will be considered one of the most intrinsic parts of every application build and part of the CI/CD pipelines. Testers will not be involved just for QA, but rather more towards understanding the application architecture and involved in deployment activities as well.

Hyper-Automation Testing: Manually writing test scripts is gradually giving way to robotic process automation (RPA). With time being at a premium and the complexity of software testing increasing, business enterprises are looking to embrace hyper-automation. In other words, they are looking to implement test automation using RPA, AI, ML, and Natural Language Processing (NLP). This way, enterprises can reduce manual efforts, get more work done faster with fewer resources, save more in terms of cost and time, automate many processes, achieve better integration and ROI, and deliver impactful solutions.

Conclusion: Measuring the Success of Test Automation – Key Drivers

To avoid the failure of test automation, automation should be a transformative approach that changes not only the testing itself but the entire culture of an organization. Also, a clear strategy and a shared vision of the test automation goals, tasks, and outcomes should be supported by teams across the company. Only then should the teams decide on the cases that need automation and the tools.

Test automation is a type of software development, and just like any other code, it requires time, effort, and resources as well as continuous maintenance and management. As soon as Project teams accept that, they will be able to make the most out of automation and improve the frequency and quality of releases significantly.

Below are a few pointers that can be used to measure the success of test automation:

Is automation your default practice?	Do your automated tests execute anywhere, anytime?	Do tests execute without human intervention?
Do your tests run frequently?	Are all executing tests trusted?	Does maintenance take less than 15% of engineers' time?
Is your reporting understandable at a glance?	Does your automation team use software development best practices?	

Author Bio

Prashanth TV is a Practice Director at Happiest Minds with over 16 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.

Business Contact **business@happiestminds.com**

happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

www.happiestminds.com

About Happiest Minds Technologies

Happiest Minds Technologies Limited (NSE: HAPPSTMNDS), a Mindful IT Company, enables digital transformation for enterprises and technology providers by delivering seamless customer experiences, business efficiency and actionable insights. We do this by leveraging a spectrum of disruptive technologies such as: artificial intelligence, blockchain, cloud, digital process automation, internet of things, robotics/drones, security, virtual/augmented reality, etc. Positioned as 'Born Digital . Born Agile', our capabilities span digital solutions, infrastructure, product engineering and security. We deliver these services across industry sectors such as automotive, BFSI, consumer packaged goods, e-commerce, edutech, engineering R&D, hi-tech, manufacturing, retail and travel/transportation/hospitality.

A Great Place to Work-Certified™ company, Happiest Minds is headquartered in Bangalore, India with operations in the U.S., UK, Canada, Australia and Middle East.