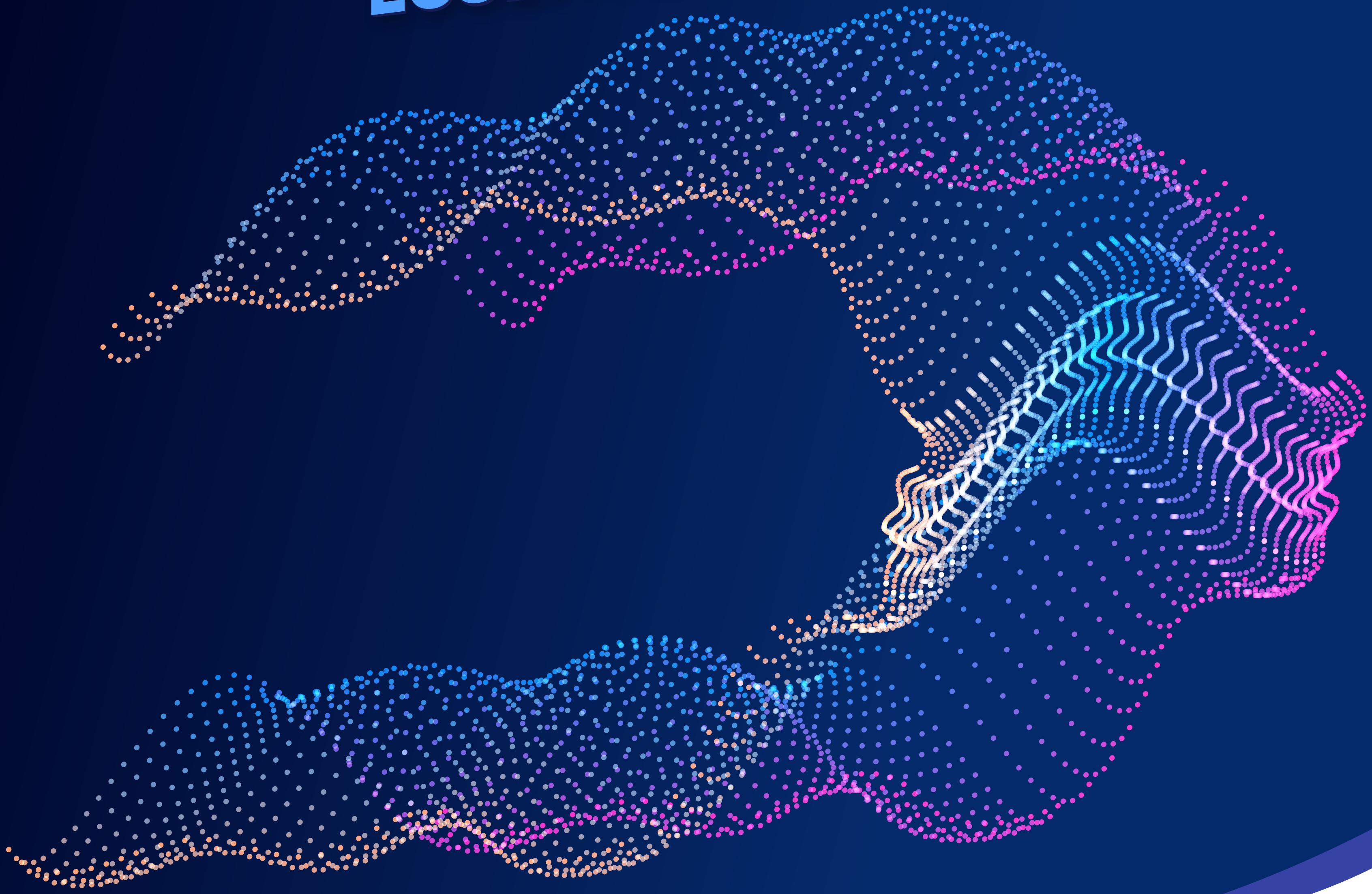
TEST AUTOMATION ESSENTIAL GUIDE



QMETRY

Table of Content

Test Automation Essential Guide

It is no news that organizations across sectors are in a race to release faster, better quality software to meet user expectations and stand the test of a dynamic economy and competitive landscape.

As enterprises embrace digital transformation, DevOps and Agile have challenged all aspects of the SDLC including software testing and quality assurance. The traditional, manual and delayed methods of software testing no longer work in the current ecosystem of connected devices and complex applications.

One critical element of this new normal is Test Automation. Defined as the practice of running tests automatically, managing test data and using test results to speed up the cycle and improve software quality. As a result, Test Automation is indispensable to the CI/CD pipeline and Continuous Testing.

Broadly speaking, the process of automated testing involves running scripts that are executed by using software testing tools. The result? Applications with higher accuracy and stability.

Why Test Automation

Challenges of Manual Testing

Manual testing has its limitations. When organizations scale up and deploy apps continuously with incremental improvements, speed is of essence.

Automated testing can streamline repetitive but important tasks using a formalized testing process or carry out additional testing that is difficult to do manually.

The practice of CI and CD depend heavily on tests that run quickly and reliably. If you tried to achieve this manually, it would be impossible to get the desired velocity.

In the modern context, automation is imperative.

There is a wide consensus that the main benefits of test automation save company time and money. But perhaps, the most important benefit of automation is the ability to give quick feedback to developers.

Many medium to large-scale organizations and software enterprises have depended on Test Automation for some time now.

Some key advantages of test automation include

- Digital Agility
- Regression Testing
- Test Coverage and Scale
- Continuous Testing and Delivery

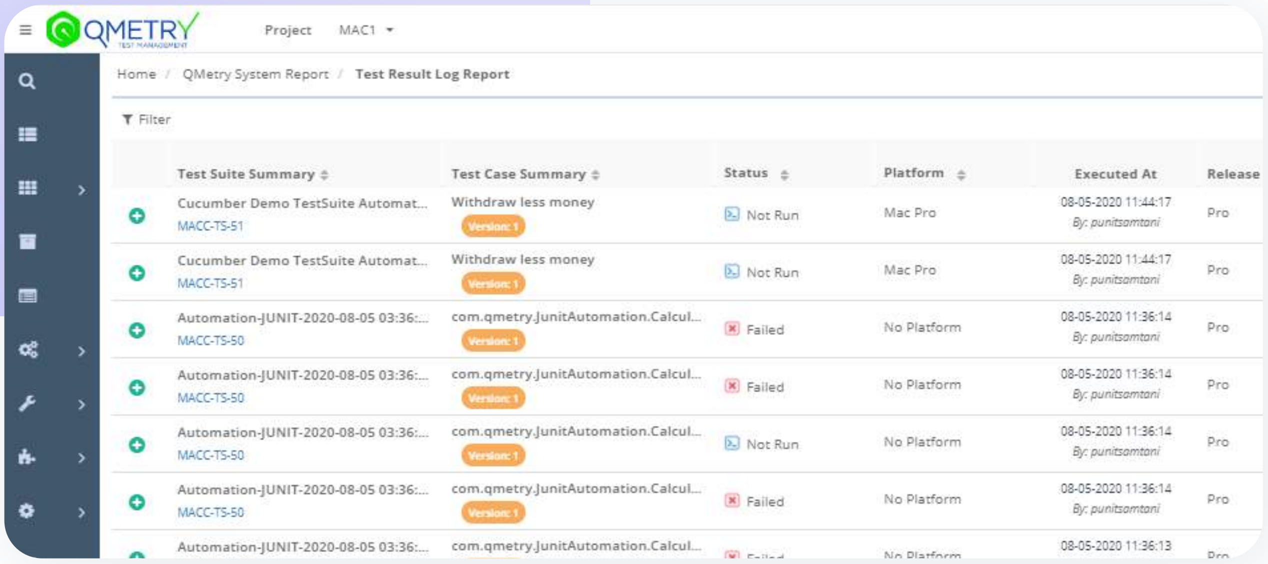
Test Automation for Beginners

If your organization is in the process of transitioning from manual to automated testing, then you will experience many immediate and long-term benefits. However, many enterprises aren't sure how and where to start.

Want to know more about transitioning from manual to automation?

Sign Up for Free Consultation

Create a Test Case Foundation



Test Suite Summary	Test Case Summary	Status	Platform	Executed At	Release
Cucumber Demo TestSuite Automat... MACC-TS-51	Withdraw less money Version: 1	Not Run	Mac Pro	08-05-2020 11:44:17 By: punitsamtani	Pro
Cucumber Demo TestSuite Automat... MACC-TS-51	Withdraw less money Version: 1	Not Run	Mac Pro	08-05-2020 11:44:17 By: punitsamtani	Pro
Automation-JUNIT-2020-08-05 03:36:... MACC-TS-50	com.qmetry.JunitAutomation.Calcul... Version: 1	Failed	No Platform	08-05-2020 11:36:14 By: punitsamtani	Pro
Automation-JUNIT-2020-08-05 03:36:... MACC-TS-50	com.qmetry.JunitAutomation.Calcul... Version: 1	Failed	No Platform	08-05-2020 11:36:14 By: punitsamtani	Pro
Automation-JUNIT-2020-08-05 03:36:... MACC-TS-50	com.qmetry.JunitAutomation.Calcul... Version: 1	Not Run	No Platform	08-05-2020 11:36:14 By: punitsamtani	Pro
Automation-JUNIT-2020-08-05 03:36:... MACC-TS-50	com.qmetry.JunitAutomation.Calcul... Version: 1	Failed	No Platform	08-05-2020 11:36:14 By: punitsamtani	Pro
Automation-JUNIT-2020-08-05 03:36:...	com.qmetry.JunitAutomation.Calcul...	Failed	No Platform	08-05-2020 11:36:13	Pro

Well-designed manual test cases are the blueprints for what you can and cannot automate. By building your test case foundation, your organization will be able to explore automation framework options, aim for the right KPIs, pick a tool that will also enable you to store and organize your manual test cases. When automating, you must not ignore manual test case organization considerations.

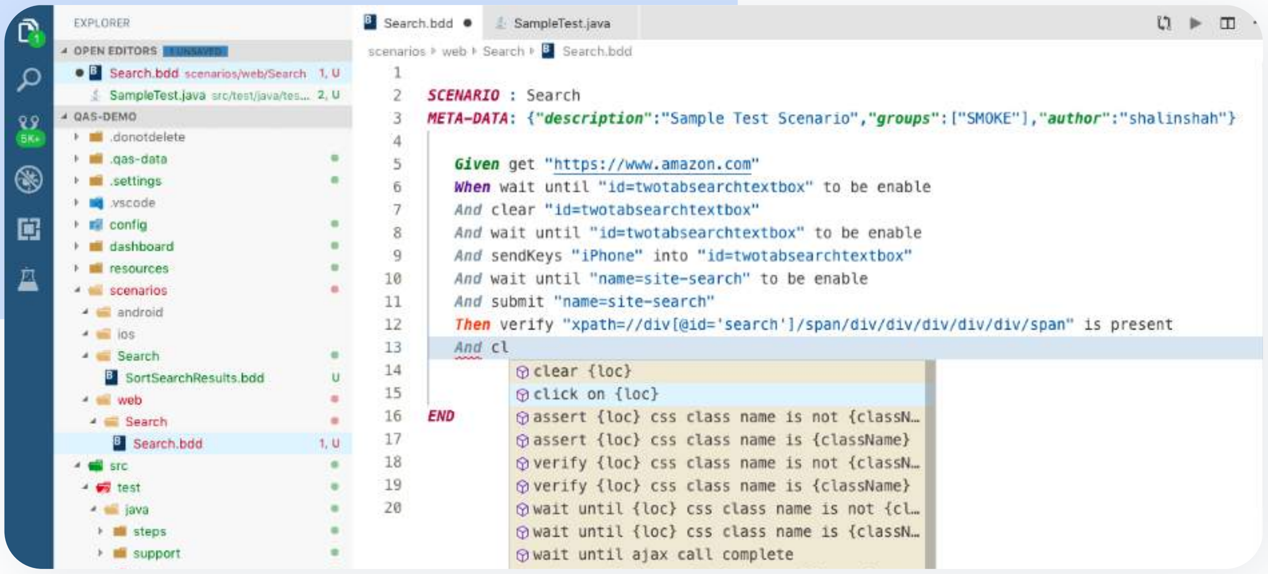
For example, the above image is of QMetry Test Management which helps agile teams to manage testing and enable them to achieve Continuous Testing. As you can see in the image, your test management tool must show tracking of what has been automated, control and manage different versions, allow test plan and cycle management, integrate with defect tracking tools such as Jira, export options and able to execute automated tests from the tools.

It is important to remember that everything is not 'automatable'. Planning which test cases to automate, you must first look for tests that are deterministic, require less or no human intervention, and are generally laborious to test manually, or repetitive in nature.

Typically, tests that are ideally suited for automation include unit tests and tests that run against different data sets, tests that focus on critical paths of application or those that need to run against several builds and browsers. Load/stress testing is also ideally suited for automation.

A good rule of thumb is to choose test automation for any activity that saves team efforts and time.

Choosing the right automation framework options



```

SCENARIO : Search
META-DATA: {"description":"Sample Test Scenario","groups":["SMOKE"],"author":"shalinshah"}

   Given get "https://www.amazon.com"
   When wait until "id=twotabsearchtextbox" to be enable
   And clear "id=twotabsearchtextbox"
   And wait until "id=twotabsearchtextbox" to be enable
   And sendKeys "iPhone" into "id=twotabsearchtextbox"
   And wait until "name=site-search" to be enable
   And submit "name=site-search"
   Then verify "xpath=//div[@id='search']/span/div/div/div/div/span" is present
   And cl
           clear {loc}
           click on {loc}
END        assert {loc} css class name is not {classN…
           assert {loc} css class name is {className}
           verify {loc} css class name is not {classN…
           verify {loc} css class name is {className}
           wait until {loc} css class name is not {cl…
           wait until {loc} css class name is {classN…
           wait until ajax call complete
```

Think of a test automation framework as the scaffolding that provides the execution environment for your automated test scripts. It enables the users to develop, execute and report on the automation test scripts efficiently.

Test Automation Frameworks can be classified on the basis of their design. Generally, they are categorized as the ones that provide module-based testing, library architecture testing, data driven testing, keyword driven testing and behavior driven development.

For example, QMetry Automation Studio supports data driven, code-driven as well as behavior driven development.

To pick the right framework for your company, ask yourself some of these questions:

- The application and technology involved: What is the user experience? How was the app built?
- Testing requirements: What is the application workflow like?
- License cost of the tool: What costs are associated with each framework?
- Skillsets available within your company: Is your team skilled enough to use the framework? Will you need to hire more expertise?

When evaluating different tools, it is a good idea to create a scorecard to review parameters like ease of scripting, UX, reporting, integration options, etc. This makes it easier to select the right tool.

Infrastructure options

Also take into consideration the hardware options. You will need servers either physical, self-managed virtual machines or cloud-hosted servers. Explore how you can install and maintain the hardware and have a backup strategy in place.

What will be the software requirements? Analyze the merits of buy versus build vis a vis hosted software. It is necessary to have a futuristic strategy that includes both testing type - manual and automated.

Proof of Concept

It is always ideal to get a Proof of Concept to get a wider buy in when transitioning from manual to automated testing.

Sign up for a free trial of the testing products and tools under consideration. Create your baseline and get started. Remember to include a real date for your proof of concept. Occasionally. you may need to extend the trial. In this case, just reach out to the company and ask.

Free Trial of QMetry Automation Studio

PoC is your chance to share results, suggestions and create excitement within the larger team. It is helpful to use metrics and other supporting details to explain why the time is right for test automation.

Test Automation ROI

Transition from manual to automated testing has a remarkable benefit on your ROI. While test automation has higher upfront costs, you can invest sensibly to maximize the gains.

A simple formula that helps you analyze the ROI of the transition:

Automation cost = the total cost of tools + total cost of labor to create the automated tests + maintenance costs.

If your automation cost calculation is lower than the manual execution cost of these tests, then you have made the right choice. ROI also tends to add up every time you re-run your automated test suite.

Maximizing Coverage and the ROI of Automation for Established Test Automation Practitioners

Many organizations have already made the leap from manual to automated testing. They have also seen improvements in release velocity and their ROI. Yet, a common goal that many organizations aim for is reaching the full potential of test automation. This means full coverage or maximum coverage.

While the goal of 100% coverage seems far-fetched, we must at least know how to measure it.

Some of the most common metrics that help you analyze the coverage include:

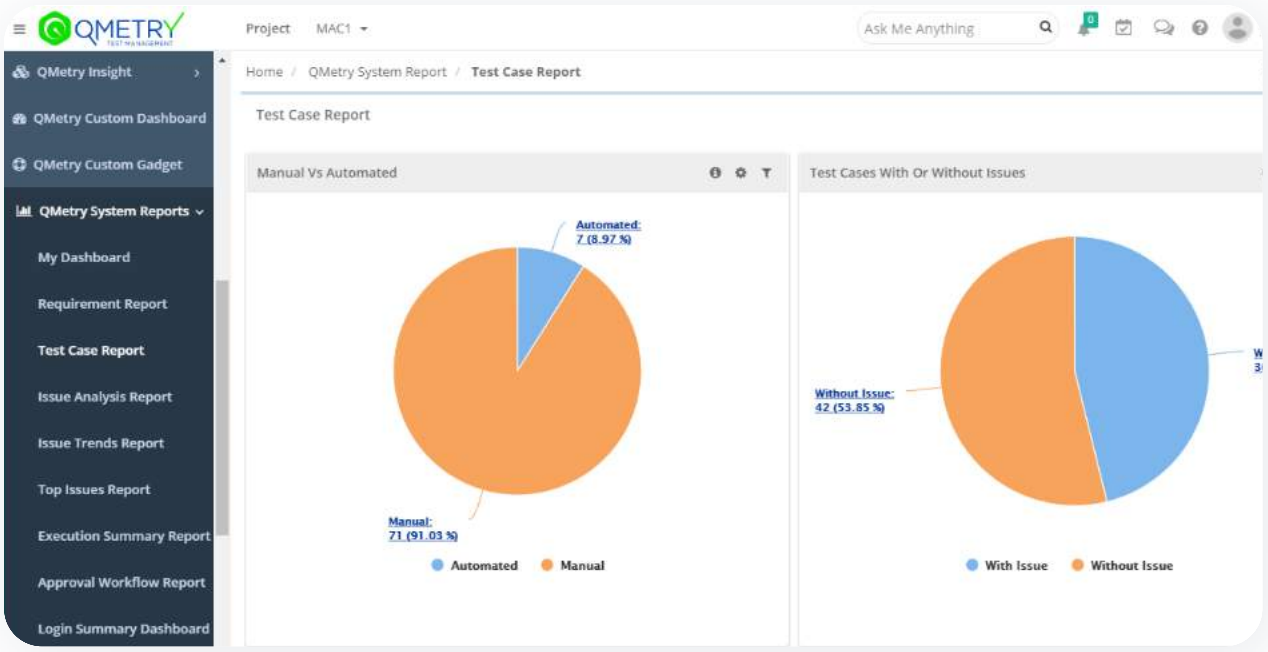
Percentage of Manual Test Cases covered by Automation

When you can convert manual tests to automated tests as they are, then you can confidently say that you have reached 100% when you are done.

Since manual tests never fit automation process as they are, they need to be broken into a few tests or changed or modified to be converted into reliable and useful automated tests.

Yet, there will be test cases that are not ideal for test automation and will remain manual. Another assumption that we make when calculating the coverage is that all our manual scenarios cover 100% of what needs to be covered.

This is seldom true, especially in the case of the legacy applications. Your test management tool should provide reports that should enable your testing teams and eventually your organization to know the manual vs. automation testing coverage.



Percentage of Covered Features

Whether you are talking of manual tests or automated ones, it is helpful to know how much functionality or features these tests cover.

To calculate this effectively, you will need to map all the features to respective test cases. One simple feature may have 10 comprehensive tests and a complex one might have 10 simple tests.

For this metric to reflect the accurate percentages, documentation and test management are equally important, or they will not provide a real picture of the coverage.

Percentage of Code Coverage

Martin Fowler has rightly said about automation test coverage analysis: “It helps you find which bits of your code is not being tested. It is worth running coverage tools every so often and looking at these bits of untested code.

Measuring code coverage as a metric is of course helpful. But what’s more important is the analysis of the uncovered areas of code. Doing this exercise helps you to discover unexplored areas and add new tests to cover it. Or, delete that piece of code if it is redundant or dead. Getting close to maximum code coverage (ideally 100%) gives one the confidence that the tested code is almost perfect.

High code coverage simply offers you the confidence for refactoring and improving the inner code quality.

Here are a few tips with which you can optimize your test coverage and even aspire for the 100% goal.

1. Capturing potential tests early: Include test cases in the story or feature card along with the acceptance criteria. This allows developers to build from the tester’s perspective.

2. Estimate the effort for automation of tests: Include testers in your estimation sessions to account for roadblocks like additional data setup requirements or a change in the testing approach.

3 Write API tests — include acceptance criteria: Functional tests rarely get the importance they deserve at the development stage. If Business Analysts start including 'writing API tests' as an acceptance criterion on a story or feature card, it will improve the outcomes.

4 Use a code coverage tool: It is also important to publish code coverage reports of all the test suites to the entire team to drive the importance of writing automation tests. Ideally, the tool should be a part of the build pipeline and the pipeline can be made to stop when the agreed benchmark is not achieved.

Code coverage metrics also help teams to identify automated areas and limit the scope of manual regression. This helps in avoiding redundant or repetitive testing cycles.

Project level KPIs

Digitally mature organizations that are well ahead in their DevOps journey need both metrics and project level KPIs to measure or validate that the test automation efforts ultimately lead to better software that meets customer expectations.

Some of these include:

Requirements coverage

This metric measures the testing effort and helps answer the question "How much of the application was tested?"

Determining this is quite simple. Divide the number of requirements covered by the total number of scoped requirements for each sprint.

$$\text{Requirements coverage} = \frac{\{\text{Number of requirements covered}\} \times 100}{\{\text{Total number of requirements}\}}$$

This metric is quite important for key stakeholders because it demonstrates the progress of the app/product.

Defect Distribution

Defect Distribution metric provides visibility into those areas where defects are being found. The number of defects found should gradually lessen as the project progresses. Above all, defect distribution metrics help identify hotspots like problematic requirements that create bottlenecks for development.

Defect Density

Similarly, defect density is measured by dividing the total number of known defects with the size of the software entity.

This metric is useful because it helps identify those areas that require automation. If defect density is higher for a specific function or feature, then it requires retesting. However, instead of retesting, test cases for known bugs can be automated.

Defect Open and Close Rate

This metric is a ratio of the defects found after delivery divided by the number of defects found before delivery. This ensures the project runs rapidly and mitigates major defects in.

Execution Trends

This KPI tells you exactly which tests have been executed by the QA team and other trends related to the defect status. This is a useful metric for QA managers because it allows them to quantify the ability and effectiveness of individual team members. Additionally, there are a large number of metrics and KPIs that help you measure the effectiveness and ROI of your test automation better.

Download this guide from QMetry to find out more about metrics through your test automation journey.

Download Automation Metrics

Test Automation and Continuous Testing for Enterprise Teams & Seasoned DevOps Practitioners

Many enterprises that were early adopters of DevOps are now experienced in their test automation journey and experiencing great ROI benefits of successful automation adoption.

The foundation of a good DevOps practice is a continuous integration/continuous delivery pipeline that automates the entire process of software development, packaging apps and deploying them to the desired environments along with service calls that enable the application. This is only possible through effective use of Continuous Testing and Test Automation.

Why Continuous Testing?

The key benefit of Continuous Testing or CT is that it provides quick insights into the risks associated with new release candidates before they hit production.

CT comprises many types of testing across different layers in the application. So, right from unit, integration, API, performance, systems, security and functional UI/UX testing, CT is present across the lifecycle of software development to:

- Identify critical business risks
- Deliver feedback at every stage of the delivery pipeline (Continuous Feedback)
- Shorten the feedback loop between development and deployment of software

The magic words here are 'Shorter time' in finding risky and detrimental defects within the software to ensure a quality product.

A good DevOps practice is therefore – DevTestOps. Where continuous testing goes hand in hand with CI/CD steps to enhance the time between product enhancements and feature updates.

Continuous Testing reveals development defects much earlier in SDLC leading to faster time to patch issues and eliminating the risk of production defects.

This efficiency is what leads to more frequent and better managed deployments. As a result companies release new build deployments on a shorter cadence — often once every two weeks to multiple times in a day.

Amazon was reportedly deploying code at 11.7 seconds on average, while Netflix deploys code thousands times per day. Facebook claims 50,000 and 60,000 builds for Android alone. However, the benefits are not restricted to faster deployment alone.

Nordstrom's customer app team did a DevOps makeover resulting in higher throughput, lower production defects and monthly releases instead of bi-yearly. Fidelity Worldwide Investment implemented a DevOps approach and automated software releases framework to meet a critical trading application with a strict launch deadline. For Fidelity, this solution led to a saving of $2.3 million per year in cost avoidance for the single app. Using this precedent, Fidelity has automated the release of dozens of applications to increase release velocity and decrease test-team downtime.
(Source: https://techbeacon.com/devops/10-companies-killing-it-devops)

How Continuous Testing Makes DevOps happen with Test Automation?

The core idea of Continuous Testing relies on automated testing of software as it passes through various procedures in the pipeline.

Obviously, automation at this level requires scripting individual processes and orchestrating the steps from checking in code to finally running the application. Matured DevOps organizations use automation effectively to drive this process change and ideally do smaller, more frequent deployments to deploy new releases and functionality to users and improve quality.

Yet automation coupled with frequent deployments is no guarantee for quality.

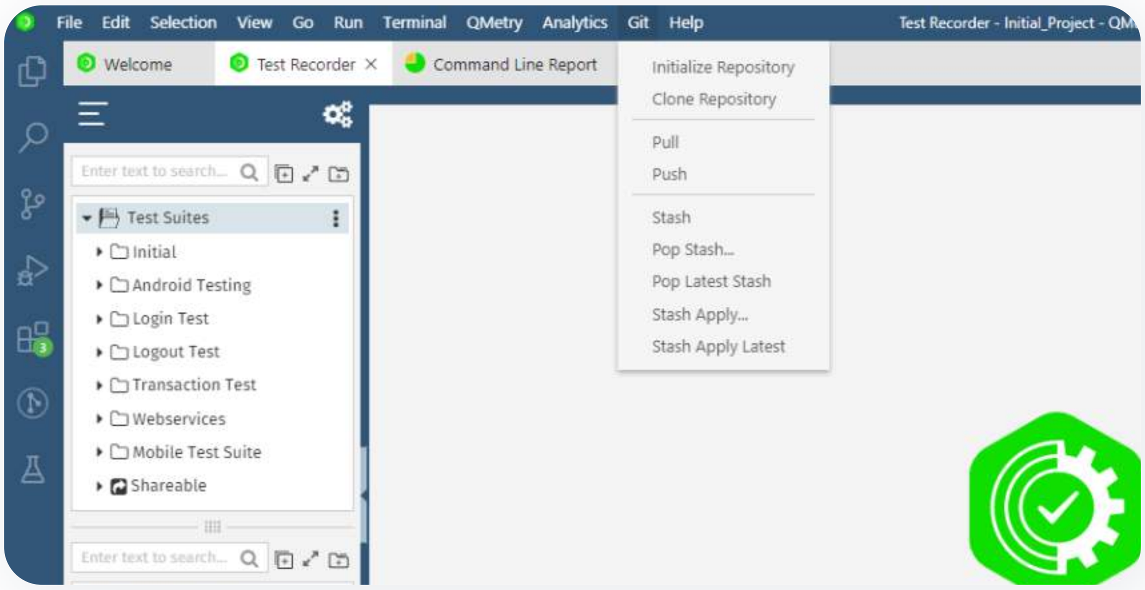
This is what test automation needs to test for:

- Whether the application and code changes have been tested thoroughly
- If the release meets the minimal acceptance benchmark for deployment
- Will the new release introduce production bugs that impact users, or are difficult to resolve? (These might be disruptive for the teams to resolve)
- Is the application tested for known security issues?
- Has the application been evaluated sufficiently for performance?

The solution? A near-perfect CI/CD practice testing should not only be automated but be integral to the CI/CD pipeline. To put it simply, ensuring that dev teams that aim for CI/CD orchestration must implement continuous testing aligned with test automation.

For this, teams need to have a set of ready automated tests that can be plugged into the pipeline. A technical best practice to evaluate risks, prioritize quality assurance executions and automation of the most critical app tests needs a combination of people, processes and technology to achieve the “pipe dream”. For example, your automation tool or framework that you are using should be able to integrate with CI/CD tools such as Jenkins, Bamboo; Source code repository tools such as Git, Bitbucket, GitLab; and other test management tools such as QMetry.

Below image is an example of QMetry Automation Studio which integrates with Git and allows developers to sync with Git.



Why Test Automation is crucial for Digitally Matured Enterprises?

Businesses invest a great deal in customer experiences as part of digital transformation programs. They are constantly fighting for high-quality and speed. Mobile applications need to be tested across a range of platforms and operating systems.

In addition to productivity and speed gains, automation provides better test coverage. For complex systems such serverless computing, debugging issues is harder and more labor intensive. With the help of test automation, teams can develop tests against individual or groups of application components and services. This makes it easier to isolate issues that may become production errors.

There are several other benefits of a sound CT practice aligned with test automation. Evaluating the end-user experience, reducing the number of false positives, better code quality, lower risks for mission-critical applications etc.

However, Continuous Testing is a relatively new concept and one that requires a cultural shift for all stakeholders involved in the development process.

Continuous Testing can prove challenging to execute without the right tools, sufficient know-how and a sound test automation strategy.

Summary

To conclude, test automation has a significant advantage for organizations at various stages in their adoption journey. For beginners, the transition from manual testing to automated testing shows immediate gains in efficiency and the ROI, for more evolved organizations — achieving maximum code coverage is possible using the right metrics and automation strategy, whereas for those who are higher on the DevOps maturity scale, continuous testing positively impacts continuous delivery with the aid of test automation.

QMetry Automation Studio can meet you wherever you are in this journey. omni-channel, multi-language scripting, reusability, easy transition from manual to automated, multi-platform testing, exploratory testing recording, integration with other defect management and CI/CD tools, QMetry Automation Studio is consistently ranked as one of the top test automation tools in the market.

Hands-on Experience of QMetry Automation Studio and see the benefits for yourself

Sign up for your free Trial

Help your team on-board quickly and jump start the test Automation process with QMetry's Automation Jumpstart

Sign Up for Free Consultation

QMETRY

(408) 727-1101 info@qmetry.com www.qmetry.com