White Paper

Application Delivery Management

The Buyer's Guide for Selecting Software Test Automation Tools

While many companies are racing to embrace agile and DevOps practices that increase the speed of development, surprisingly few are using test automation to keep up with the increased pressure on testing. Explore the current state of software test automation tools, see why testing is so late to the game, and learn how to create a testing practice that can deliver on the dictum: Quality at Speed.

MICRO FOCUS

Table of Contents

page

The Practical Guide for Selecting Software Test Automation Tools 1
Why Test Automation Matters 1
Why Teams Are Slow to Embrace Test Automation 3
The State of Software Test Automation Tools 4
Choose a Tool the Tester Can Use 4
How to Choose between Open Source and Commercial Tools 6
The Types of Test Automation Tools: What to Look for in Each 7
Emerging Test Automation Trends 10
Issues, Gaps, and Areas that Need Improvement 12
What's Coming Next? 13
Next Steps 14
How to Create an RFP for Test Automation Tools 15
Introduce Yourself: Provide a Statement of Your Current Status 15
Vendor Information Section 16

Test automation is a challenge for many organizations, but offers extraordinary benefits. Here's how you can embrace best practices, understand the types of test automation tools available today and how they support quality assurance objectives, and learn what to look for when evaluating your options.

The Buyer's Guide for Selecting Software Test Automation Tools

Ready or not, it is time to get serious about software test automation. In organizations of every size, developers are working faster and smarter as they adopt agile methodologies. Unfortunately, in most places, testing is still predominately manual, and testers simply can't keep up. The result is "a mountain of technical debt and increased complexity," according to the survey report Accelerate: State of DevOps 2018: Strategies for a New Economy, which leads to more manual controls and layers of process that slow the work even more.

"We're right in the middle of this transition where people are starting to understand that to do continuous delivery they need more automation in testing to keep up," said Diego Lo Giudice, Vice President and Principal Analyst at Forrester Research. The quest for "quality at speed," which emphasizes reducing application cycle times, increasing quality, and getting to market more quickly, is a major driver of this IT trend, according to the World Quality Report 2018-19.

Both technology and practice issues are slowing the adoption of test automation. "It's going to take some time before organizations transform themselves with the skills to go beyond the UI level of testing," cautions Lo Giudice. Test automation must shift left and become part of everyone's job. But the right automation tools, applied to the right tasks, in the hands of the right people, is a potent combination that is well worth the investment.

Test automation is a challenge for many organizations, but it offers extraordinary benefits. Here's how you can embrace best practices, understand the types of test automation tools available today and how they support quality assurance objectives, and learn what to look for when evaluating your options.

The guide concludes with a step-by-step guide to creating an effective request for proposal (RFP) when acquiring test automation tools.

Why Test Automation Matters

Although test automation entered the market over 15 years ago, only 14–18% of organizations are using even the most basic test automation tools. If you're not there yet, you're not alone. According to the World Quality Report, this lack of automation has become the "number one bottleneck for maturing testing in enterprises."

Figure 1. Why Test Automation Matters

The ones who have embraced test automation enjoy significant competitive advantages, including lower costs, faster time to market, and better detection of defects.

The ones who have embraced test automation enjoy significant competitive advantages, including lower costs, faster time to market, and better detection of defects. The Accelerate study analyzed the practices of different development teams that lead to higher software delivery performance. Automation was a key differentiator.

It found that elite DevOps teams do only about 10% of their testing manually. These teams:

- Can produce multiple deploys per day.
- Need less than one hour of lead time for changes (between code commit and code deploy).
- Have a change failure rate of 0–15%.
- Can restore an application or service in less than an hour after an unplanned outage or service incident occurs.

They may not always choose to deploy multiple times a day, but the code of a high-performing team is actually deployable at any time. This is due in part to the fact that testing happens all the time, so quality is built in.

Figure 2. Automated vs. Manual

Automation tools can help provision test data securely and quickly, as well as perform fast infrastructure provisioning.

At the other end of the spectrum, low-performing teams did 30–50% of their testing manually. These teams:

- Deploy between once a week and once a month.
- Require one to six months of lead time for changes, on average.
- Have a change failure rate of 46–60%.
- Need between one week and one month to restore an application or service after an unplanned outage or service incident.

Elite and high-performing organizations did significantly less manual work than their lower-performing peers in all dimensions. This enabled them to spend more time doing new work and less time remediating security issues or defects.

Why Teams Are Slow to Embrace Test Automation

With results like that, why isn't everybody rushing to automate?

Technical Challenges

More than half the respondents in the World Quality Report cited automation challenges arising from things like "applications that change too much with every release," and a primary challenge was the lack of predictable and reusable test data and test environments. In fact, 66% of respondents still use spreadsheets to manually generate test data for multiple iterations of testing. And 62% use copies of their production data for testing.

Fortunately, automation tools can help provision test data securely and quickly, as well as perform fast infrastructure provisioning.

Bad Experiences with Previous Automation Projects

It's not just about choosing the right tool. It's about choosing the right tool for the right team for the right job. Teams that get any of those elements wrong may have a bad experience that can sour the company on test automation for a long time. See Top 8 Reasons Software Test Automation Initiatives Fail.

It all starts with the wrong expectations. The managers "may think they are replacing things that testers do with automation, instead of seeing automation as a support for their testers," said Bas Dijkstra, test automation trainer and consultant. If they bought an automation product to reduce headcount, they will have chased a misguided expectation. "Generally, test automation ends up taking more people, not less," said Paul Merrill, Automated Testing Consultant at Beaufort Fairmont. But it produces many multiples of the work that they would do without it.

Teams may have tried to automate the wrong things, with poor results. Joe Colantonio, Founder of TestTalks.com, advises testers to be smart about what they choose to automate. Identify tests that are repeatable and deterministic to get some quick wins, he advises. If the code is changing, or there is any randomness involved, it's probably not the best place to start with automation.

And don't invest months "creating a framework and scripts to test things you only test once a year, or you won't realize any savings in the long run," he said. Target something that will give you real value right away, and one success will lead to another.

Joe Colantonio, founder of TestTalks.com, advises testers to be smart about what they choose to automate. Identify tests that are repeatable and deterministic to get some quick wins, he advises. If the code is changing, or there is any randomness involved, it's probably not the best place to start with automation.

Developer Tester Skills Are Hard to Find

Many teams can't move forward with test automation because they can't find enough skilled developer-testers. According to The Forrester Wave: Omnichannel Functional Test Automation Tools, Q3 2018 report, "Developers are getting more involved in API-functional, UI-functional and non-functional test automation. But companies are having a hard time finding enough full-stack testers or DevTesters, even from testing service providers."

Fortunately, the remedy for this does not depend on producing more developer-testers. Instead, new tools are emerging that enable non-developers to automate testing. You can choose tools designed for people who aren't coders—technical testers and business testers—and greatly expand your pool of testers.

The State of Software Test Automation Tools

The software test automation market is crowded with choices, and vendors compete vigorously for your business with buzzwords and benefits statements. But you must understand your team's capabilities before you buy, or you could end up choosing tools that can theoretically do everything you want but won't work for your team in practice.

"You might find yourself buying a tool that will hurt you more than help you," warned Angie Jones, Senior Developer Advocate at AppliTools. "In the real world, problems arise from the lack of technical skills in the people who will work with the tools." You'll get something that sounded great until you realize you need a subject matter expert (SME) in a certain area to be able to do what you need, she said, "And you don't have one."

Choose a Tool the Tester Can Use

Think carefully about who will be using an automation tool before you buy it. Most vendors design their tools for the role, or persona, that is most likely to use them. But when you assign a persona to use the tool that is different than the one the vendor envisioned, they may not be able to make it work.

Ignoring tester capabilities can set your team up for failure. For example, technical testers who are not coders are now handling many testing activities that were once the domain of developers.

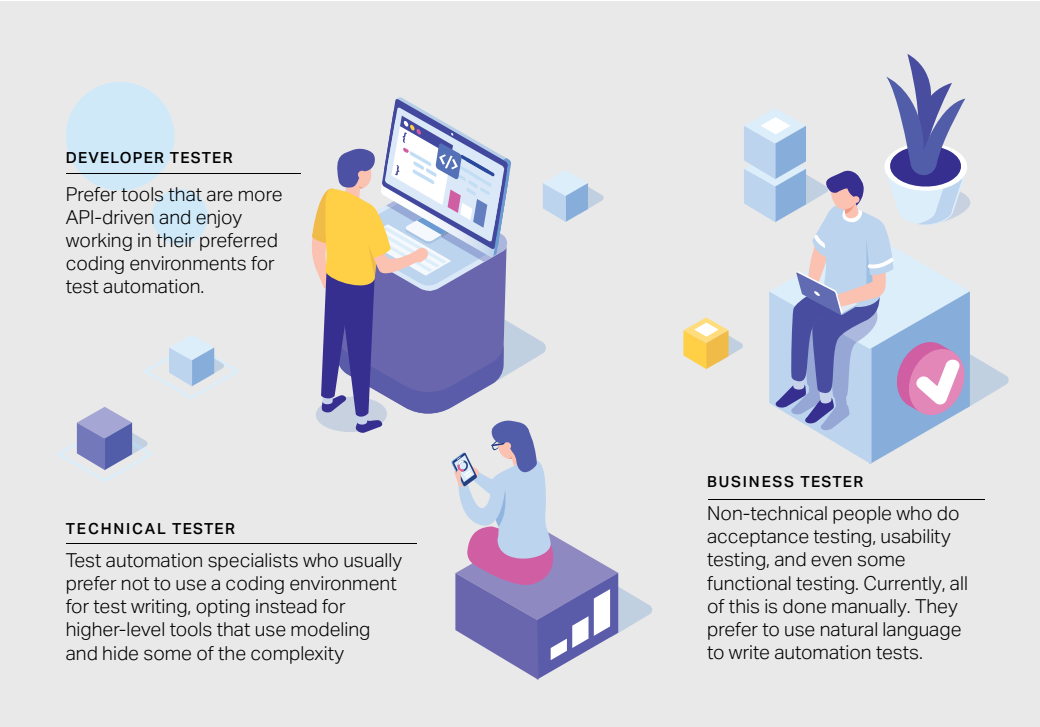

Figure 3. Three Categories of Testers

Testers generally fall into three categories, with distinct preferences and capabilities:

- **Developer testers** prefer tools that are more API-driven and enjoy working in their preferred coding environments for test automation.
- **Technical testers** are test automation specialists who usually prefer not to use a coding environment for test writing, opting instead for higher-level tools that use modeling and hide some of the complexity.
- **Business testers** are non-technical people who do acceptance testing, usability testing, and even some functional testing. Currently, all of this is done manually. They prefer to use natural language to write automation tests.

"It's important that we enable business testers to test in the future because we don't have enough developer and technical testers to do the job," said Lo Giudice. But business testers are not well supported by the tools available today.

Ignoring tester capabilities can set your team up for failure. For example, technical testers who are not coders are now handling many testing activities that were once the domain of developers. "People choose a tool based on what their developers like to use," said Colantonio, then technical testers end up doing automation without any help from the developers, using a tool they don't like.

To avoid surprises, Colantonio recommends doing a two-week proof of concept before buying a tool. "Run it through the full development lifecycle to see how it works in your environment" with the people who will have to use it. This is far more effective than a product demo at revealing any skill gaps that you will need to resolve with training, or even hiring, if you really want to be able to use that tool, he said.

Most test automation functionality is available from both open source projects and commercial tool vendors, but the open source versions require more technical skill to use.

How to Choose between Open Source and Commercial Tools

Most test automation functionality is available from both open source projects and commercial tool vendors, but the open source versions require more technical skill to use.

User preferences in the open source or commercial tools decision have shifted of late. "We're seeing more managers encouraging their people to justify why not an open source tool," said Lo Giudice. "It used to be the other way around—justify why an open source tool instead of a commercial tool."

Jones takes a pragmatic approach to this decision. There are some great open source tools out there that can get the job done. But commercial tools offer more functionality and support than do most open source tools. "I like to start with the open source ones, and figure out if any of them will meet my needs," said Jones, then she looks at the commercial offerings to see if what they offer is worth the cost.

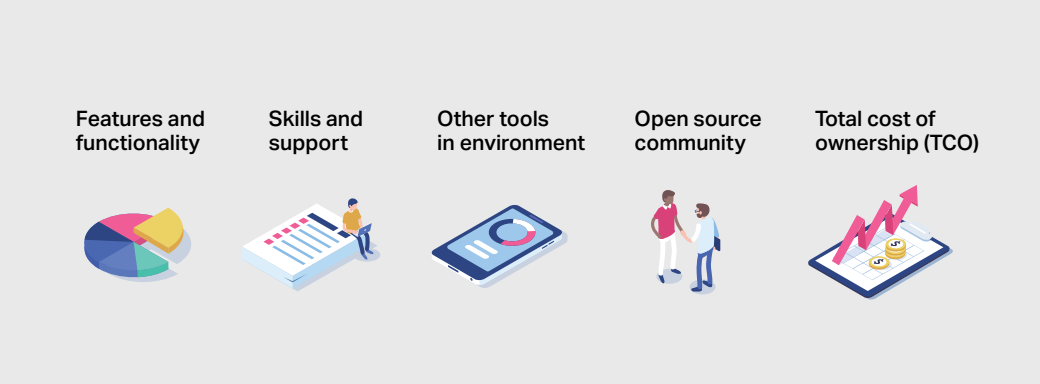

Figure 4. Important Considerations

Important considerations when deciding between an open source and a commercial solution include:

- Required features or functionality
- Skillset of your team and their need for commercial support or training
- Compatibility with other tools in your environment

In general, testers tend to go with more open source tools at the start, said Lo Giudice, then "at some point they find they are better off with a solid commercial tool that can scale and doesn't need all the support they have to put into it."

- For open source tools: the activity, viability, and longevity of the open source community
- Your own total cost of ownership (TCO) analysis

A TCO analysis is key when considering a migration from a commercial to an open source tool. Teams usually undertake them to save on licensing fees. But such migrations have hidden costs, according to Merrill, such as the need to train the team to handle the open source tool and to rewrite existing scripts. And with today's tight labor market for test engineers, you should also consider how the tool you choose might affect who might want to work for you—or stay working for you—said Merrill.

In general, testers tend to go with more open source tools at the start, said Lo Giudice, then "at some point they find they are better off with a solid commercial tool that can scale and doesn't need all the support they have to put into it." When companies are using third-party testing services, they tend to favor open source. If they are doing their own testing, they tend to use more commercial tools that don't require as many technical skills, he added.

The Types of Test Automation Tools: What to Look for in Each

When companies start automating parts of their testing, they typically begin with unit and functional testing, then expand to non-functional tools.

Unit and Functional Test Automation

Automated unit testing works with the smallest pieces of code (aka units) you can isolate, to validate that they perform as intended. This type of testing is typically part of every build and is the job of the developer. This is not testing from a functional perspective but from a programming perspective.

How it helps: Runs through the algorithms that the developer has written to expose any technical errors.

Look for:

- Unit testing frameworks that have been developed for the programming languages you need to support.
- Unit testing frameworks that can produce results and reports that are compatible with other tools that you're using.

Functional test automation tools check to see if the application is doing what the business wants it to do, not just what the developer thinks is right. Some automation tools exercise the application's user interface (UI), others can test the API, and some do both.

How it helps: Looks at the implementation of the app to ensure it is delivering what the business expects.

Look for:

- A tool that can handle the 10–30% of the UI you need to test, as well as API testing.
- Test script creation capabilities that offer a broad set of capture approaches, such as record and replay, keyword- and page-based, custom and common gestures, API recording, etc.
- Flexibility in editing test scripts. A tool that provides both scriptless and script editing won't require advanced dev skills. This will make it friendly for non-dev testers to use.
- Design automation and optimization capabilities. If the tool supports both model-based and data-driven testing, it will provide more flexibility. Also, a tool that supports optimization from artificial intelligence (AI) and machine learning (ML) to improve the automation of tests will reduce test execution times and costs.
- Support for testing processes like test-driven development (TDD) and behavior-driven development (BDD).
- Support for real-time test monitoring. Real-time monitoring will allow you to shift left with monitoring, and integrate it with testing. It will also enable you to shift right with testing so it can be leveraged by monitoring.
- Reusable test scripts that you can use across devices, browsers, and OSes. This will support omni-channel functional testing initiatives.
- Tools with built-in advanced analytics based on machine learning that can offer prescriptive insights.
- Reporting capabilities that let you drill down on data and correlations, and that can pass data into application lifecycle management (ALM), or business intelligence (BI) tools, and save and share reports.

Non-Functional Test Automation

When you're ready to move beyond unit and functional test automation, look at tools that automate non-functional testing in strategic areas that support quality at speed.

Performance testing measures application performance and load stress capacity by simulating user traffic, and analyzing the app's responsiveness, stability, scalability, reliability, and speed. This reveals how the app will do under pressure from high traffic and other stressors that can lead to failure.

How it helps: Tests how well an app will handle projected levels of user traffic to ensure end-user satisfaction.

Look for:

- Lightweight tools that will complement your traditional end-to-end performance testing tools. These tools support the need to shift left, enabling performance testing earlier in the cycle.
- Open source and cloud-based tools, which many developers prefer when apps have to be used on the web or around the world and want to test loads geographically.

Security test automation is an area of growing importance in light of increased data breaches and cyber-security threats, and it may be a requirement of your organization's AppSec program. Some elements of

Performance testing measures application performance and load stress capacity by simulating user traffic, and analyzing the app's responsiveness, stability, scalability, reliability, and speed. This reveals how the app will do under pressure from high traffic and other stressors that can lead to failure.

Many agile teams take an incremental approach to integration testing by using stubs (code that simulates the response of a lower-level module) and drivers (code that simulates the response of a higher-level or parent module) to test integration on a module before all of the other modules with which it will interact have been completed.

security testing don't require additional tools—most functional test tools can test basic security features such as authentication and logout. But security is too critical to rely on functional testing tools alone. Tools are available that can automate static code analysis (looking at vulnerabilities in source code) and perform dynamic analysis (looking at how well an app can withstand attacks). That's invaluable when you need enhanced security testing.

How it helps: Increases the security of applications and reduces the risk of data breaches.

Look for:

- Tools for static code analysis that will identify security vulnerabilities in the application source code, pinpoint the root cause for each of them, and prioritize each by severity.
- Tools for automated dynamic testing that will mimic real-world hacking techniques and attacks and provide a comprehensive dynamic analysis of web applications and services.

Integration testing evaluates whether a component or module integrates properly with the other components and modules on which it will depend in production. This is a problem area for many teams, according to Lo Giudice. They either do it manually or wait until the end of the development process when all modules are available, which is too slow for continuous delivery. But if integration testing is skipped altogether, the result is poor quality and critical production issues, said Lo Giudice.

Many agile teams take an incremental approach to integration testing by using stubs (code that simulates the response of a lower-level module) and drivers (code that simulates the response of a higher-level or parent module) to test integration on a module before all of the other modules with which it will interact have been completed.

How it helps: Reveals potential conflicts between components and modules so you can address them early.

Look for:

- Tools that use stubs and drivers to simulate dependencies not yet available for testing.
- Ability to save stubs and drivers in a library for each application module for future reuse.

Infrastructure automation and system provisioning let you quickly stand up temporary test environments on demand and deploy them in an automated way without heavy configuration overhead. This addresses the bottlenecks caused by reliance on permanent test environments.

How it helps: Improves consistency between tests, supports agile development, and increases overall efficiency.

Look for:

- Tools that will enable the creation of containers that have no dependency on any other platform or services and that can hold standard test configurations.
- Tools that can create and manage clusters of those test containers.

Test Management: There are two different kinds of test management software, including test case management and test data management.

TEST CASE MANAGEMENT

Helps teams manage and organize all testing efforts, including managing all test cases, test plans and runs, tracking results, and producing reports and metrics.

How it helps: Enables QA managers to easily and accurately assess quality and know when a product is ready to deploy.

Look for:

- Tools that support continuous delivery methodologies.
- Integration with the tools, frameworks, and systems your team is using.
- Tools that will manage both manual and automated test cases and defects.

TEST DATA MANAGEMENT

Lets teams run rigorous system tests against masked or reliably mimicked production data, without exposing sensitive data in a test system. These tools address the problems caused by manual practices, such as creating spreadsheets and copies of production data for testing.

How it helps: Helps teams build more reliable software by using high-quality test data that is similar to the actual data that the application will be using, without security and compliance risks.

Look for:

- Tools that offer robust data masking to increase privacy and security. If the data holds personally identifiable information (PII), these mechanisms protect it from being seen by the dev and test teams.
- Tools that provide synthetic test data generation that mimics the production data, for data sets that are too sensitive for masking.

Emerging Test Automation Trends

In high-performance teams, test automation practices are always evolving. They are quick to embrace emerging technologies and find new and better ways to ensure a first-rate experience for their end users without delaying the speed of development. Here are some important trends in the testing practices of mature teams.

In high-performance teams, test automation practices are always evolving. They are quick to embrace emerging technologies and find new and better ways to ensure a first-rate experience for their end users without delaying the speed of development.

Performance testing is becoming more important to organizations of every kind. It has never been something that every organization felt it needed to do, but with the growth of mobile apps and the need to support mobile users, performance testing has become a hot button for everyone.

Testing across All Endpoints

App users expect a good experience no matter what device they are using. That's why more teams are investing in omni-channel functional test tools, which reveal how an app looks and feels on different endpoints, mobile devices, mainframe screens, and desktops, said Lo Giudice. You can find cross-browser testing tools, as well as services that provide a farm of devices on which to test your mobile apps.

Testing the API

Since the complexity of the architecture is growing, "we're seeing more organizations going beyond the interface and automating at the API level—testing the process that is behind the UI," said Lo Giudice. Since the API tends to change less often than the interface, this makes automation less fragile. It also tests the design of the API and can influence API changes.

Performance Testing

Performance testing is becoming more important to organizations of every kind. It has never been something that every organization felt it needed to do, but with the growth of mobile apps and the need to support mobile users, performance testing has become a hot button for everyone.

Functional test and performance test are converging, according to Lo Giudice. Large functional-test suites are adding some performance testing capabilities, so as you test a feature to see that the function is correct, you can also look at performance.

Peak-performance testing is another area that has become business-critical for many companies. If the app can't deal with extraordinary surges in usage—such as online shopping on Black Friday—it is at risk of failing at the most critical time, leading to loss of customers, revenue, and brand loyalty.

Service Virtualization

Service virtualization tools help teams perform integration testing in an automated way. Testing with service virtualization is "early and iterative—ideal for agile and DevOps," said Lo Giudice. These tools create a simulation environment that allows teams to test their app against another app or service without needing to have the other component available for testing. They create a "virtual asset" that behaves like the real component, so testers can exercise it in their tests.

This allows the team to test apps that need to access services that aren't developed yet, have defects, or charge a per-access fee. You are isolated from all these dependencies and can quickly locate the problem when things go wrong in a complex system.

Some of these tools even let you play with the dependent services, letting you see how your app would behave if a service has an error, so you can ensure that users have a good experience even when things outside of your control go awry.

These tools don't have the level of adoption they should, said Lo Giudice, "but for large organizations with acquisitions, or when multiple teams are doing testing, they are very strategic." Only a few of the largest enterprises use it currently, but any organization with more than 1,000 employees would benefit, he said.

Issues, Gaps, and Areas that Need Improvement

Test automation tools still need improvement in several areas to enhance their usability and functionality in the typical practice.

More Affordable, Feature-Rich Service Virtualization Tools

Service virtualization has not been adopted as quickly as it should have, in large part because commercial tool choices are limited, the tools that are available are very expensive, and open source options have limited functionality. Initial adoption was hampered by a lack of understanding of the business case by companies who could have benefited from it, and pricing models forced customers to buy enterprise licenses when they wanted to start small. But the situation has been improving, according to Lo Giudice.

Help Deciding What to Automate

Testers have a fundamental need to know if they are testing the right things. "Are you really covering the risks of your application, or just pumping up the number of automated tests you are doing?" Colantonio asked. Without knowing that, you can get a false sense of security about quality without actually lowering the risk.

Testers need tools equipped with AI that "help you make better decisions about what to automate and what not to automate," since not all tests are good candidates for automation, said Dijkstra. And it would be useful to know which tests have the highest value and ROI, said Jones.

Better Codeless Test Automation Tools

More technical testers need to be engaged in creating automated tests, and this persona is much happier using a codeless tool. However, many codeless tools have significant gaps, which vendors should address to better serve this market. See 10 features every codeless test automation tool should offer.

A Better Way to Triage Failed Tests

With automation tools, testers spend a lot of time with maintenance and debugging why tests failed. Testers need a tool that will triage tests when they fail and then report findings to the team so they don't spend so much time debugging them, said Colantonio.

Testers need tools equipped with AI that "help you make better decisions about what to automate and what not to automate," since not all tests are good candidates for automation, said Dijkstra. And it would be useful to know which tests have the highest value and ROI, said Jones.

Over the next few years, test automation tools will leverage advances in machine learning and artificial intelligence, and data and test automation will "become more predictive when it comes to QA and testing," according to the World Quality Report.

What's Coming Next?

Figure 5. What's Coming Next

Over the next few years, test automation tools will leverage advances in machine learning and artificial intelligence, and data and test automation will "become more predictive when it comes to QA and testing," according to the World Quality Report. You can expect adoption of AI across the whole lifecycle, "from the requirements side, through operations, to customer experience."

Here are three examples of how AI will simplify and augment test automation in the future.

Rise of the Business Tester

The next generation of tools will use AI to enable business testers to help with test automation, said Lo Giudice. "That's important because we don't have enough developers and technical testers to do the job." He expects that the number of business testers will increase, the tools will advance more than for any other category, and more scriptless test writing options will become available.

Helping Testers Work Smarter

Tools will augment testers' intelligence with AI and ML. For example, AI should be able to "automate the exploration of an app and the customer's use of it, to see which path should be added," said Jones. And when there are common elements that appear in things like every shopping cart, "AI should be able to automatically determine how to test them," she said.

Autonomous Testing

A host of new tools from startups with "disruptive AI and ML approaches will augment and automate more, setting a path for autonomous testing," said Lo Giudice. This is still far from fully available, but it is on the horizon, he said.

Next Steps

With so many tools on the market and new technologies on the horizon, there has never been a better time to embrace test automation. If you are among the 80% of companies not yet using it, consider starting with a pilot program to introduce it to your organization.

Identify your objectives, work with your team to choose a problem that can be solved with automation with measurable ROI, and use this guide to find an appropriate tool that is a good fit for the skills of your testers. Starting with one success at a time, you will build a testing practice that can keep up with the pace of agile development without sacrificing quality.

No matter where you are in the adoption of test automation, the hunt for tools never ends. Use this guide on an ongoing basis to fill gaps in your toolchain and expand your automation capabilities.

With so many tools on the market and new technologies on the horizon, there has never been a better time to embrace test automation. If you are among the 80% of companies not yet using it, consider starting with a pilot program to introduce it to your organization.

If you are having problems with an existing tool, describe it and what you expect a new tool to provide. Vendors and service providers can become important partners, so be frank about the issues you may face with your current tool, as they may be able to suggest remedies you are not aware of.

How to Create an RFP for Test Automation Tools

After identifying problems that automation can solve, considering your staff skills, and finding a few tools that seem to match your high-level requirements, use this guide to create a request for proposal (RFP).

You'll find more questions below than you will probably need for your RFP. Use them as a guide and trim away the pieces you don't need. When you send long lists of product requirements to vendors, you get back huge responses that are virtually impossible to compare, so try to limit your product-related questions to the ten that are most critical to your organization. By focusing on your top ten requirements, you will find it much easier to compare vendors and make an informed decision.

When evaluating the vendor responses, consider assigning weightings to each question so that your most important decision criteria rank highest as you score the candidates.

How to Handle Open Source Tools

You can also use the RFP approach to evaluate open source tools, but you'll have to fill in the answers yourself by inquiring with the users in the project communities. By following the same criteria to look at both open source and commercial tools, you will end up with an apples-to-apples comparison to guide your selection.

Introduce Yourself: Provide a Statement of Your Current Status

Statement of Goals, Concerns, and Objectives

Describe your automation goals, the problems you are trying to solve, and any relevant metrics you are hoping to improve via automation with this tool.

State the greatest concerns for company management relative to DevOps, and how you expect this tool to play a role.

If you are having problems with an existing tool, describe the problems and what you expect a new tool to provide. Vendors and service providers can become important partners, so be frank about the issues you may face with your current tool, as they may be able to suggest remedies you are not aware of.

Statement of Resources

Explain the number and type of testers who will use the tool.

- How many developer-testers will use the tool?
- How many non-coding technical testers (including automation engineers)?
- How many business testers?

Infrastructure Requirements

List the main elements of your environment that are relevant to the tool usage.

- Operating Systems
- Hardware
- Databases

Vendor Information Section

- Describe your company, history, and test automation focus.
- Who is the primary contact for sales?
- Who is the primary contact for technical questions?
- Do you have reference customers that can be contacted? If so, please list them with appropriate contact information.

Vendor Products and Capabilities

This section helps you understand the product, the vendor's approach to test automation and DevOps, and any innovations the vendor offers.

- List your product or products, and the primary category each falls into:
  - Unit Testing
  - Functional Testing
  - Test Management
  - Performance Testing
  - Security Testing
  - Integration Testing
  - Service Virtualization
  - Other (please describe)
- What UI technologies and API protocols do you support?
- What skills are required to use this tool?

- What browsers do you support?
- What devices do you support?
- What platforms do you support?
- Do you support test automation on the mainframe?
- In what ways do you support omni-channel functional testing?
- Do you provide packaged application testing for commercial ERP, CRM, and other tools (e.g., Oracle, SalesForce, and SAP) that have been customized for an organization?
- Which Continuous Integration or Continuous Delivery tools do you support?
- How easy is it to integrate your solution with open source solutions?
- Describe how extensible your solution is.
- What types of reporting and analytics capabilities does the product provide?
- Is a third-party reporting tool integration required for reporting?
- For security testing, does your product support static code analysis? Dynamic analysis?
- What are the unique features or innovations that differentiate your product from the competition?

Deployment, Updates, and Support

- How is your product or service offered?
  - On-premises
  - As a cloud service
  - As a hybrid solution
- Does deployment require a consulting engagement?
- How often do you release new versions or update the software?
- What is your pricing model? Is there a distinction between users who define tests and users who execute them? How flexible is your licensing?
- Do you offer support? What are the terms of your support contracts?
- What training options do you offer?
- How many hours of training do you recommend to attain proficiency in using the product?

Contact us at:
www.microfocus.com

Like what you read? Share it.