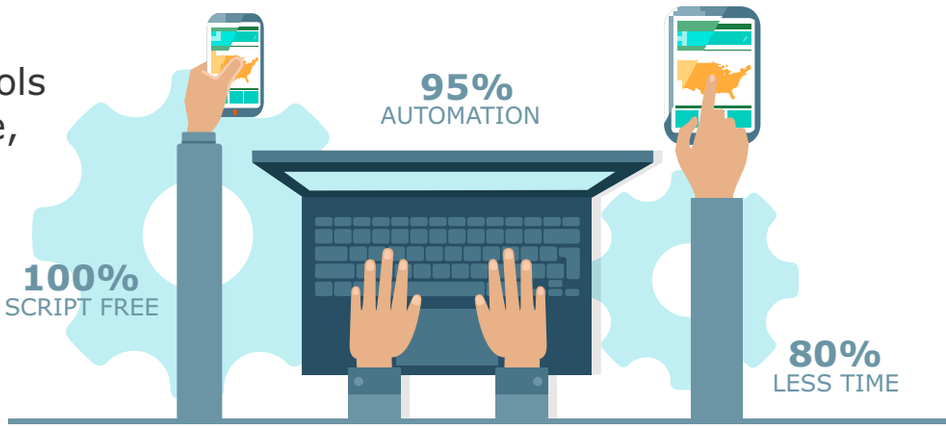
QUALITIA®

FUNCTIONAL TEST AUTOMATION TOOL: EVALUATION CRITERIA

+91 20 30157482

There are several types of test automation tools – unit & integration testing, performance, security, functional testing etc. This document focuses on helping you determine right evaluation criteria for Functional Test Automation tools.



SKILLS AVAILABILITY

You need to consider if you have test automation engineers or mostly manual testers/ SMEs. In case test automation engineers are available with intermediate to expert level programming knowledge, design skills and motivation to create and maintain test automation framework and test suite, then it is perhaps appropriate to consider conventional automation tools (scripting). In case you have more of manual testers or SMEs or testers without adequate programming and design knowledge, then you are better off selecting scriptless automation platforms.

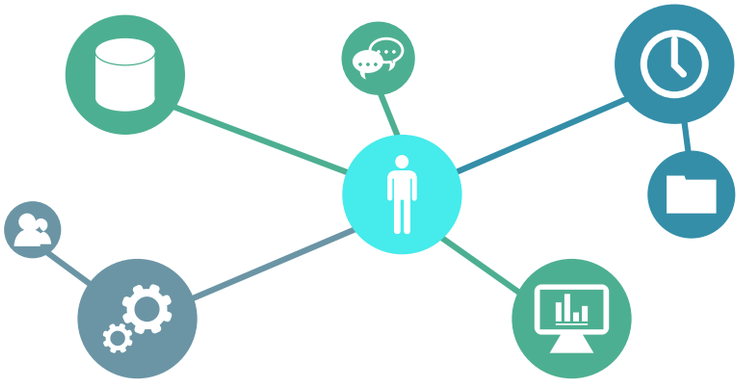
EASE OF USE

Test automation tool should be easy to use. Ease of use should be evaluated from the context of your user base & their skills. Tool should make it easy to create test scripts, test data and re-usable components. The tool should also make it easy to maintain your automation suite and adapt to any changes in requirements. Generally many tools talk about "record and playback" as part of their solution for ease of use. However just record and playback cannot be a scalable option. The tool has to provide ease of parameterization, handling of dynamic objects/ workflows/ pages/ data as well as ease during maintenance.

SPEED OF AUTOMATION

Rapid test script development is very important. This is facilitated by simplicity, usability and multiple productivity features like debugging, comments and modularity. Simple, cohesive & comprehensive user interface with powerful features should exist in order to accomplish complex tasks – test cases involving multiple data conditions, conditional execution (IF/ ELSE, conditional operators, loops), simple data

management, ability to automate test cases with complex workflows like DB, web services, file handling, complex UI objects like - trees, tables, dynamic objects, nested objects in UI automation etc.
Speed of test suite execution is equally important. Truly unattended mode of execution helps in saving lot of time otherwise spent in monitoring the execution. Here reliability & accurate reporting/ logging of the execution also plays an important role.



SUPPORT FOR MULTIPLE TECHNOLOGIES

The tool should support all the technologies that are used in your softwares under test. Technologies used for creating thick client applications, web based user interface, web services (RESTful or SOAP), Mobile Applications (Android/ iOS, native/ hybrid/ mobile web), databases, file handling etc. need to be considered. Tool should provide out-of-the box support for these technologies, while providing unified user experience.

SUITABILITY TO DEVELOPMENT METHODOLOGY

Conventional automated testing tools are usually targeted for use at the end of usually long programming phase as part of waterfall development methodologies. So you have time to develop automated test scripts before they are used. However, in Agile methodologies, the time between iterations is short and time available for developing automated test scripts is short too. So you need support for rapid development of test suites as part of automated test tools. Agile also demands support for evolving requirements. Agile often may not put too much emphasis on documentation as compared to waterfall/ SDLC. It is important for the automation tool to enable in-sprint automation. Automation should begin as soon as wireframes (and without documentation) are ready, with much less modifications when the software is ready for testing.

DEVOPS

In case you are moving towards Continuous Delivery, then Continuous Testing is also your need. In this scenario, the automation tool should be able to integrate seamlessly with your Continuous Integration tools (e.g. Jenkins, Bamboo, TeamCity, Hudson etc.) and infrastructure. In addition, the tool should support automation suite movement from testing to staging to production environment, as the test cases continue to evolve and as software under test is promoted from testing to staging to production.

REUSABILITY

Modularity in your test suites is extremely important in order to ensure high maintainability and traceability. Automation tool should provide out-of-the-box capabilities to drive modularity. Your team will be much more productive if this is available. This is very useful especially during the maintenance phases of your automation suites. Usually creation of version 1 of automation suite takes fraction of time as compared to the time and effort spent in maintenance during it's entire life span. Hence you ought to ensure that you get the modularity right in v1 itself. You should choose the tool that encourages this modular thinking within your testing team.

REAL-TIME MULTI-USER COLLABORATION

In case you are going to create automation framework and test suites in a project having multiple team members, it is important that the tool has capability to ensure real-time collaboration. In case you have distributed teams, the tool architecture should enable teamwork in order to boost productivity.

CROSS PLATFORM SUPPORT (INCLUDING CLOUD)

You need to assess the platforms where your test suites should be executed – Operating systems, browsers and multiple

combinations thereof. It is also important to assess whether testing needs to be carried out in Internationalization and Localization environments. Automation tool should execute the test suite across any of these environments, with minimal or zero modifications. Tool should also support execution on-demand on the cloud platforms of your choice. “Write once and run anywhere” should be your goal.

INTEGRATION WITH YOUR DEVELOPMENT ECOSYSTEM

Test automation is never an isolated piece, rather it should be an integral part of your development and testing ecosystem. Hence it is important that the automation tool has integration with project management, test management, defect management, continuous integration etc.

ERROR HANDLING AND VALIDATIONS

During testing, it is seldom possible to encounter no errors/ exceptions with software under test. These errors can be native OS dialogs or unexpected objects from software under test. The test automation tool should provide out-of-the-box capability of run-time error detection, take appropriate action once error is detected, recover from error state etc.

Automation tool should also have in-built library of functions for a number of validations – UI, data, objects, DB, files (word, PDF, text, xml), web services etc. These are commonly known as assertions. This is an integral part of any test automation. Assertions help testers to check actual results against expected ones.

fraction of time as compared to the time and effort spent in maintenance during it’s entire life span. Hence you ought to ensure that you get the modularity right in v1 itself. You should choose the tool that encourages this modular thinking within your testing team.

TEST DATA MANAGEMENT & PARAMETERIZATION

Test data and test flow should be loosely coupled. Any tight coupling between flow and data will result into heavy costs in test creation and maintenance. The automation tool should follow test scenarios/ use case approach rather than tying the user to specific inputs per test case.

Test data should be extracted from multiple input types – DB, flat files, XLS, web services etc. In addition, it should be simple to centrally maintain this test data per test case.

MAINTAINABILITY

Many organizations select the automation tool considering scenarios at the time of running initial proof of concept. However software maintenance phase is the biggest phase in application life cycle. Hence any automation tool should be easy, quick and adaptable to constant changes to software under test.

You should be able to quickly adapt automated test suites from sprint to sprint. This becomes critical since Agile methodologies offer less time between sprints for adapting/ modifying automated test suites. If this is not properly supported, then slowly automation suite maintenance starts to develop a lag and cannot keep up with software under test any longer. Automated tests may lag a few sprints, but that may defeat the very reason to go Agile. Slowly this lag moves from a few sprints to a few releases, and one is compelled to fall back to manual testing.

EASE OF TRANSITION

Either you have existing automation tool that you want to replace or you don't. In either case, the new tool should make your team's transition smoother. In case you already have investments in existing suites (created with existing tool), the new tool should co-exist with test suites from past. Phased transition should be possible.

In case you are transitioning from manual testing, then learning curve for the new tool needs to be pretty fast. There should be ample training and support provided during the tool adoption phase. Again phased transition is important here too.

EXTENSIBILITY

Most of the times a tool may not support 100% of your needs out-of-the-box. In case there are a few such requirements that are unfulfilled, then your team should be able to easily extend the tool to ensure support for all your needs. Customization and adaptability will be important here.

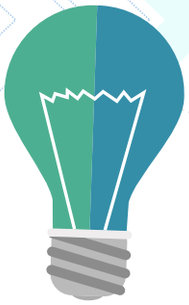
REPORTING

Reporting is extremely important at every level – per execution cycle for every suite and every test case, across multiple execution cycles and across multiple builds (of software under test). Every failure or defect should be clearly pinpointed in the reports. Reports should capture exact state of the software under test, during every stage of the execution. The execution reports should help quickly analyse results and troubleshoot any issues. Every granular information should be available, in order for the tester to understand and report defects accurately and efficiently.

Management reporting is also important part of automation reports. This should help managers/ leaders make right decisions.

TOTAL COST OF OWNERSHIP

It is not just about price of the tool, but also installation time, infrastructure needed, time for learning curve, market price of people with required skills, amount of resources required, maintenance costs, skills availability, time to value and many other indirect costs. You should absolutely be careful to determine all these costs - especially the indirect ones, since those are not so obvious. Sometimes the total price of the commercial tool may be less expensive than the open source software and vice versa. We don’t realize this, but indirect costs can cause a major issue and often show up when you least expect it.



PRODUCT SUPPORT & INNOVATION

You should gather information about innovation happening on the automation tool you want to choose. Any tool with significant R&D investments is more likely to innovate faster.
Product support should be available when you need it. Highly responsive technical support will save your time and resources. You should look for support SLAs, support plans etc.

INDUSTRY AWARDS AND RECOGNITION

You should look for industry awards as well as recognition by leading global technology analysts. Many of the well-known awards normally have very rigorous nomination and selection process. Being even nominated for such awards is a significant testimonial for several technology companies.

EXISTING CUSTOMERS

Existing customer list of the tool is a direct testimonial of the organizations that trusted the tool. Hence it is often a good idea to compare the client lists, talk to a few references and read up client testimonials. Look for clients that could potentially have similar business needs as yours e.g. same industry, geography and size etc.