Automated Testing: Not Just a Quick Fix

INFUSE

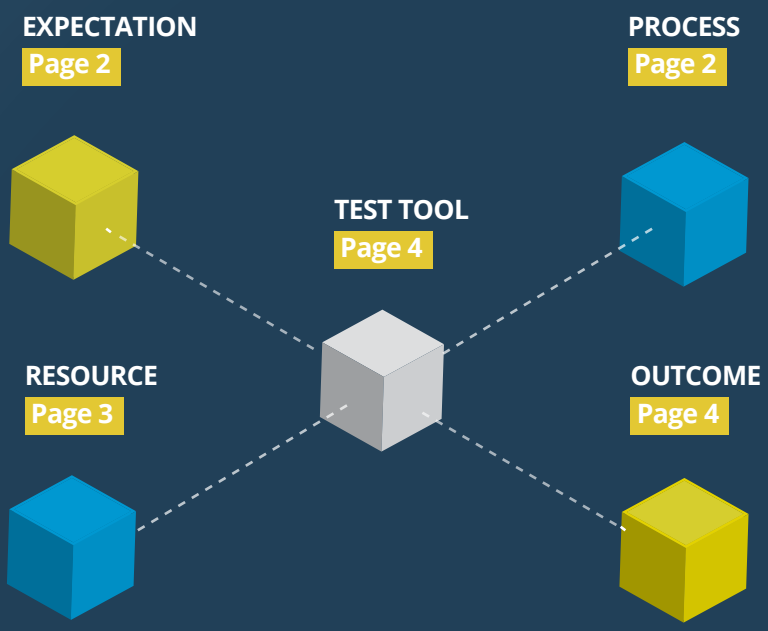
INTRODUCTION

It is amazing how often you hear that a management team is disappointed in the test tool that they bought. Is it the tool's fault? Is it the tester's fault? Or is it something else?

All too often automated testing is viewed as the solution to all testing woes yet is rarely given the resources or backing required for success.

In this paper, we have identified five areas that should be taken into consideration when looking to implement any type of test tool. These are based on experience gained over many test automation engagements. These areas are summarised in the diagram below and will be addressed in the sections which follow.

Within many Agile and DevOps teams, it is clear that the drivers and organisational structures differ from those in traditional models. Nevertheless, there is value to be found in the following approaches.



EXPECTATION

Many management teams have a certain expectation of automated testing. Often, the expectation is that it will enable testing to be quicker, cheaper, and more comprehensive (i.e. "test everything")

This expectation needs to be managed both as part of the project process but also through the tools available to the test manager, which include:

- **Test Policies**
- **Test Strategies**
- **Test Plans**

By utilising a test policy, the test manager can set out to senior management and stakeholders the high-level approach that is going to be taken for automation as well as the level of initial and ongoing support that is required to make automated testing work in the organisation. It should be made clear to the senior stakeholders that by signing off on this document they commit to the ongoing investment in the automation approach and toolset that will be defined.

Through the test strategy, the test manager can clearly define the approach taken for automation testing in the organisation, and can further define expectations of what automated testing can achieve, what areas will be covered and what investment will be required. It is also a document where areas of non-coverage can be outlined, so further setting expectations at this point.

The test plan is the final document where detailed expectations can be set, the test manager can outline what functions will and will not be covered by automated testing as part of an individual project.

PROCESS

Automated testing needs to be taken into consideration throughout the lifecycle of a project, from inception to closure (and often beyond in the case of automated regression packs).

For example, when an automated test fails and is reported as a defect, that defect is then rejected as expected behaviour because a new change has been implemented without the automation team being told that one was due. Those responsible for the test automation are then in catch up mode to try to bring the automation test pack up to date.

The earlier that the test team (including the automation tester/team) are included, the more likely the automated tests are to succeed. For agile teams, a working, proven automated test(s) should be part of the Definition of Done for the scrum. This means it can be introduced into an automated regression pack without further rework. For DevOps and Continuous Integration systems, this definition should be extended to include integration of the automated tests into the automated build and deployment scripts.

RESOURCE

The test engineers who write the automated tests are not magicians; they cannot write tests without background information, working tools or applications under test.

Too often, management thinks that their delivery team will produce all the automated tests required with little or no guidance. Similarly, the environments that the actual tool resides on and also the environment under test needs to be adequately sized and supported.

It is imperative that test tools have similar technical debt requirements as the actual systems under test. Operating systems, web browsers, and protocols to mention but a few, change overtime and will impact any tools ability to run.

Like the systems under test, the test tool will also need to upgrade and maintenance planned into any release/roll out schedule, with adequate time for regression testing and fixing any issues identified.

So often test tools are initially housed on substandard hardware, as management teams want neither the initial nor the ongoing expense of setting up and maintaining the environments until they have proven a return on investment (ROI).

In many cases, automated test runs execute at a faster rate than manual tests through screens and, in many cases, any API. This is often misunderstood by management who often will only pay for test environments that are half or quarter size (or less) of the final production version.

Management are often surprised when the environment under test fails due to the level of automated tests executed against them. So expectations need to be managed. For example, if you have a test pack of 500 tests that are expected to be executed overnight but the environment can only support the execution of 100, the test plan needs to be adjusted to reflect that the task will now take five days instead of one day.

Similarly, both the tool and test environments need to be given support to provide an effective test environment. This can include operating system releases, Windows upgrades and other housekeeping tasks to ensure that the environments stay in sync with production code and also ensure that testing is executed on a like-for-like environment.

THE TEST TOOL:

Different expected outcomes can require different test tools to deliver what is required.

The range of tools available stretches from packaged applications to open source to individual bespoke applications, all with different levels of cost and support requirements.

Seldom will one tool be able to deliver all that is expected of automated testing and time needs to be taken to select the correct tool sets in order to select those that will help deliver the level of testing that is required, rather than remain "package-ware".

Building and running a proof of concept (POC) is a useful way to identify the suitability of a tool because you can prove if a tool is really able to execute the testing and subsequent reporting that is required. A POC should focus on automating the system under test and trying to automate those complicated areas that form the core functionality. However, before embarking on a POC, the scope of automation needs to be carefully thought through. It is not always possible or financially viable to automate everything, so a clear definition of what will be automated is important.

THE OUTCOME:

As previously mentioned, the perceived successful outcome of automated testing depends on what expectations have been agreed at all levels across an organisation and what is needed on a project by project basis.

Once agreement has been reached about what is expected from automated testing and the required level of investment/support is in place, then the automation testing in any organisation has a good chance of succeeding in delivering both actual results and value for money, as well as meeting perceived expectations.

SUMMARY

To ensure maximum benefit from automated testing, the process cannot be left to operate in a vacuum and needs to be integrated into an organisation's development approach and lifecycle. To not do so can prove costly in both time and expense.

Setting and agreeing expectations up front as well as having the right resource in terms of environments, initial and ongoing support, documentation, tools and team members, can greatly contribute to the success of automation and the value it delivers to an organisation.

useMango™ Industrial Test Automation

useMango™ is a functional automated test tool and test automation framework that offers the benefits of test automation at a lower risk compared to traditional test automation approaches such as accelerator, keyword or action word frameworks. Designed with pre-built agile libraries/components, multi-platform Inspector tools, tooling for automation assets management and integration to HPE ALM, useMango™ reduces both test execution time and cost.

Learn more about useMango™

ABOUT INFUSE

Infuse is a UK software testing company that provides modern software testing, transformation consulting and test environment management. We specialise in test automation and performance engineering.

Our strong alliance and partner network enables us to deliver the right solution for every client. Infuse is an Hewlett Packard Enterprise Gold Partner in Application Delivery Management (ADM), an Oracle Gold Partner in Application Quality Management (AQM), a CA Partner in Dev and Test and SAP Partner. We have a global partner network to enhance our delivery capabilities beyond that of a typical UK software testing company.

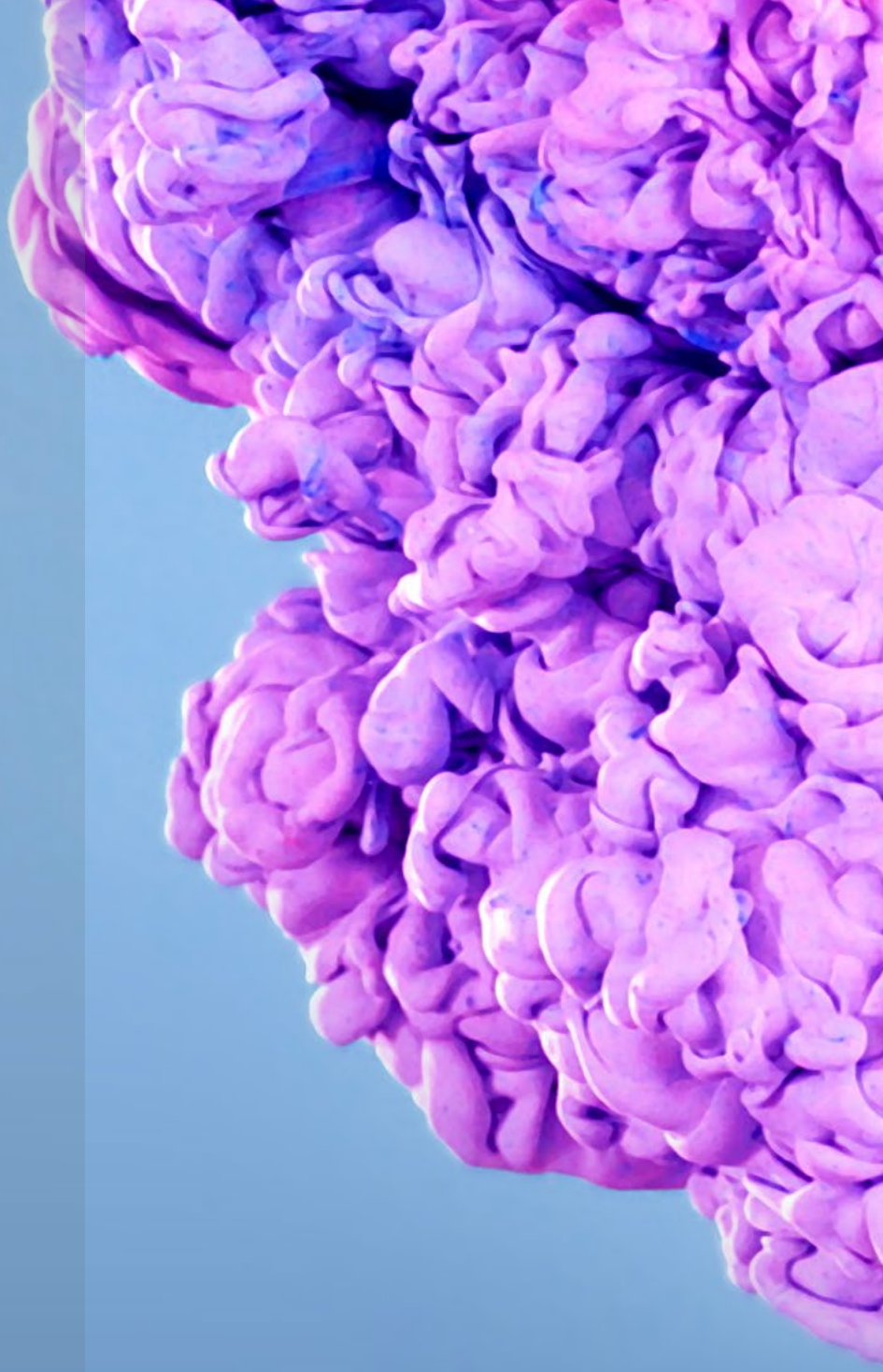
To learn more about Infuse, please visit **Infuse.it**

ABOUT THE AUTHOR

JAMES MILNE

James is a Technical Test Manager for Infuse Consulting, with over 20 years of testing experience.

His experience includes building and running test teams mainly in Financial Services on large high value programmes. When not at work he is a keen walker and trekker with recent a summiting of Stok Kangri in India, which stands at just over 20,000ft.



INFUSE

For more information, email _info@infuse.it or visit infuse.it

Infuse Consulting Ltd | QEII Conference Centre
Broad Sanctuary | London SW1P 3EE | UK
Tel: +44 (0)20 3755 5135

facebook.com/infuse-consulting
@infusegroup
linkedin.com/company/infuseconsulting

infuse.it