Trinitatum

10 reasons why test automation fails (and how to make yours a success)

Jed Dalton
Professional Services Director

Test automation – why does it so often fail to keep its promises?

Few CIOs, program or project managers doubt the need for automated testing for all but the most trivial projects. In particular, key enterprise systems that are regularly enhanced and updated to satisfy the never-ending stream of business change requests, require a rigorous validation process to ensure they will still work after each update.

Business users have an especially low tolerance for bugs that are introduced with software and such bugs will usually be costly to rectify and/or have a profound effect on your reputation.

Similarly, when you are building a new system, and particularly when you have an agile development methodology, you need a means to assure the software quality throughout the development lifecycle. We are all familiar with systems that only revealed their shortcomings too late in the project to be rectified or, perhaps worse, the 'great landing, wrong destination!' scenario.

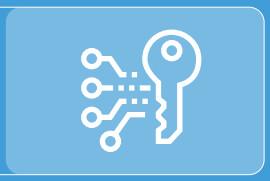
Automated testing seeks to address both of these situations. When test automation is prioritized at the project's inception, the same tests that proved the system was meeting its requirements whilst in development can continue to assure the system through subsequent upgrades and enhancements once the system is 'live'.

It all sounds simple, but have you ever found yourself asking any of the following (real- life) questions:

- Why did my well-intentioned investment in test automation fail to deliver a return?
- I paid for automated testing, so why does the system we built have so many bugs?
- Why are the users complaining that it doesn't meet their requirements?
- Where are the automated tests that I was promised now that the system is in production and I want to upgrade?
- Why does no one know how to run the tests or what functionality they cover?
- Why am I spending more time maintaining my automated tests than I do maintaining my application?
- Why did my automated tests not cover system performance?
- Why were the automated tests switched off when they started failing?

This paper seeks to address the common reasons that test automation fails to deliver on its promises and to provide guidance in avoiding the most common mistakes.

01



Lack of domain knowledge

‘They just didn’t understand the business’

Lack of domain knowledge

While there are generic testing skills that are transferrable between domains, just as you wouldn’t be happy to have your GP performing a keyhole surgery, your automation effort will benefit from testers who also understand the business and application context in which they are operating.

Testers without business awareness will suck time from the rest of the project trying to understand the system; tests constructed without this detailed understanding may execute perfectly but fail to adequately reflect the way your business actually works and hence be irrelevant. Make sure your automation experts have actually created an automation suite for a similar application and have more than a passing knowledge of the field.

02

My testers were theorists not pragmatists

‘I needed doers more than thinkers!’

Theorists not pragmatists

In common with most areas of IT, there are trends and trend-setters in the world of software testing. Such ideas and people are valuable in introducing new techniques and theories. However, beware the automation consultant whose theorizing fails to develop beyond ‘vapourware’. Ask for an early demonstration of an automated test. Request a business review of the test scripts and artefacts.

Demand an account of the percentage of business process coverage and an explanation of the handover approach and deliverables for transition to ‘BAU’.

03

Failure to focus on outcomes

‘We bought the pitch, not the delivery’

Focus on outcomes

In the world of medicine, there’s been a revolution brought about by focusing on the outcomes of clinical procedures rather than simply trusting the reputation or self-publicity of the practitioner. In the same way, when evaluating your investment in automation expertise you should be targeting the end result and also checking you haven’t employed an ENT specialist for your hernia op (cf. ‘1:Domain Knowledge’). What will actually be delivered and how will it help meet your project’s objectives?

What percentage of code or business functionality will be automated? What artefacts will be delivered and how will they be handed over to your team? Can your consultant demonstrate a track record of successful operations in the same area?

04

Missing ‘buy- in’ from the wider team

‘There was only one person committed to the automation effort’

Buy-in from the whole team

To develop a meaningful automated test suite requires the will of the entire team. Where automation requires ‘hooks’ into the system under test, the development team must be prepared to create them. Such tasks must be prioritized alongside the development of the actual software. At the same time, analysts must be prepared to cast business requirements as ‘acceptance criteria’ which, as acceptance tests, will drive the development effort. Ask the project manager how much time has been dedicated to automation tasks, a complex system may need up to 50% of the development time assigned for this purpose. Ask the analysts how they will be documenting requirements - will they be expressed as testable system behaviors?

05

Poor test design

‘The artefacts they created weren’t high enough quality’

Poor test design

Just as there is variability in the quality of automation testers, so there is a world of difference between a well-designed automated test suite that continues to deliver benefit for years and a ‘brittle’ one that doesn’t. A poorly constructed test fails to properly initialize its context or to tear-down on completion.

If it’s reliant on specific environment settings or on data that was expected to remain static but changes, it may throw false negatives like the boy who cried wolf. I’ve seen teams who start to ignore or disable the failing tests or the more conscientious, who become stuck in an expensive ongoing maintenance and debugging cycle.

06

Tests can’t be maintained

‘I’ve been left with a maintenance nightmare’

Test maintainability

A common corollary of poor test design, is a lack of test maintainability. When tests aren’t sufficiently self-describing it can be the devil’s job to even identify what’s causing a failure or to debug a failing test in order to verify the issue.

A high quality automation suite should make obvious what it is testing and ideally, the business context should be manifest for all tests. Pose the following challenges: ‘By reading the test artefacts alone can I understand what this automated test is actually doing?’ and ‘For a sample failing test, is it obvious what has failed / where the problem lies?

07

The automation ‘withers on the vine

‘What happened to those automated tests we used to run?’

The automation ‘withers on the vine’

Sometimes as a consequence of 5 & 6, this is a common failing of test automation projects. The automation fruit that has been cultivated with care, whose benefits have been reaped at ‘go-live’ is then left to languish and rot. Even the best automation assets will still incur a maintenance overhead after the project has delivered its initial phase.

Make sure that an adequate handover plan is in place. Mandate that the recipients of any automated tests are able to run, debug and enhance the automation tests as a condition of exiting the test phase. This will likely involve validating specific software and access requirements. Understand in advance what level of documentation will be provided with the tests.

08

Unstable environment

‘When are those automated tests going to be automated?’

Unstable environment

So you’ve selected the best supplier, an automation tool has been chosen and you have excellent engagement from development, analysis and the business. You will still need to make sure the environment is right. Will your team have suitable access and rights to the machines in the development estate: PCs; application and build servers; source control repositories and databases? Is there a mature development process and support for continuous integration and ‘on-demand’ software deployments?

The benefit of automated testing is only realized where the host organism is welcoming and collaborative rather than perceiving these demands as a parasitic drain.

09

Unaccountable opinions

‘People exercised influence who weren’t accountable’

Unaccountable opinions

It’s a sad fact that vested interests will often find positions of power and influence in the test arena. These people may try to mandate an approach, tool or even the resources for testing without taking responsibility for these decisions. I suggest an ‘arms-length’ approach.

If they persist, ask whether they are willing to underwrite the test results, to support the ‘live’ system or to be held accountable for any overruns in this area. It’s your project, your budget, and the quality of the end product, including its test artefacts, is your responsibility.

10

Inappropriate technology selection

‘The automation tools suited the consultant but not the project’

Inappropriate technology selection

Correct selection of technology for test automation is crucial. Certain tools are more appropriate for different types of testing and there are on-going costs for licensed 3rd party products to be considered. Often a bespoke solution or one leveraging freeware will better fit the task at hand than the ‘preferred’ tool in which your consultant happens to be proficient. Understand the reason for selection of a specific tool, are alternatives available?

How much of the functionality of the automation tool is being used? What are on-going costs for support and maintenance?

Summary

The ten pitfalls we've explored show that test automation rarely fails solely because of the testing tool selection – it fails because of how it's planned, designed, and managed. The lesson is clear: treat automation as a strategic capability, not a side task. Success depends on getting the right people, processes, and environments in place from the start, and ensuring your automation strategy evolves with your business.

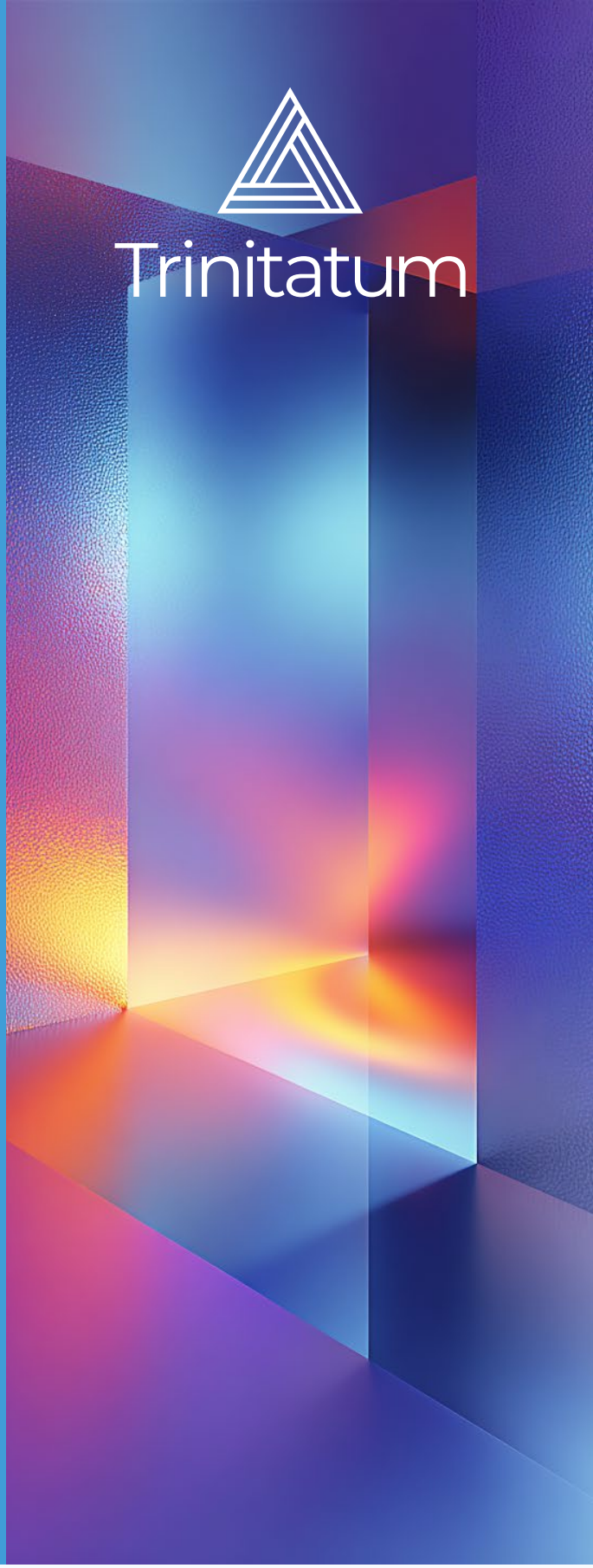
At Trinitatum, we've helped energy and capital markets organizations build automation frameworks that deliver lasting value – not just at go-live, but throughout every upgrade and release cycle. Our approach blends intelligent automation and deep domain understanding with practical, proven test assurance expertize to strengthen quality and confidence across complex trading systems.

If your automation programme isn't delivering the results you expected, or you want to make sure it does – we can help.

Get in touch for expert advice on building a stronger, more sustainable test automation strategy.

FOR MORE INFORMATION

W: Trinitatum.com
E: Hello@trinitatum.com



Registered addresses

UK: 4th Floor, 18 St. Cross Street, London, EC1N 8UN
US: 8 The Green, STE R, City of Dover, Kent, 19901