Original Software

Throwaway Test Automation

originalsoftware.com

Introduction

Software test automation has been available for over a quarter of a century but the practice has a mixed track record and often falls into disuse due to the effort required to maintain the created scripts. Achieving just a moderate level of automation coverage requires considerable investment of budget and resource. With increasing software development complexity fighting a business and IT drive for agility, traditional test automation has become too cumbersome for many to contemplate or sustain.

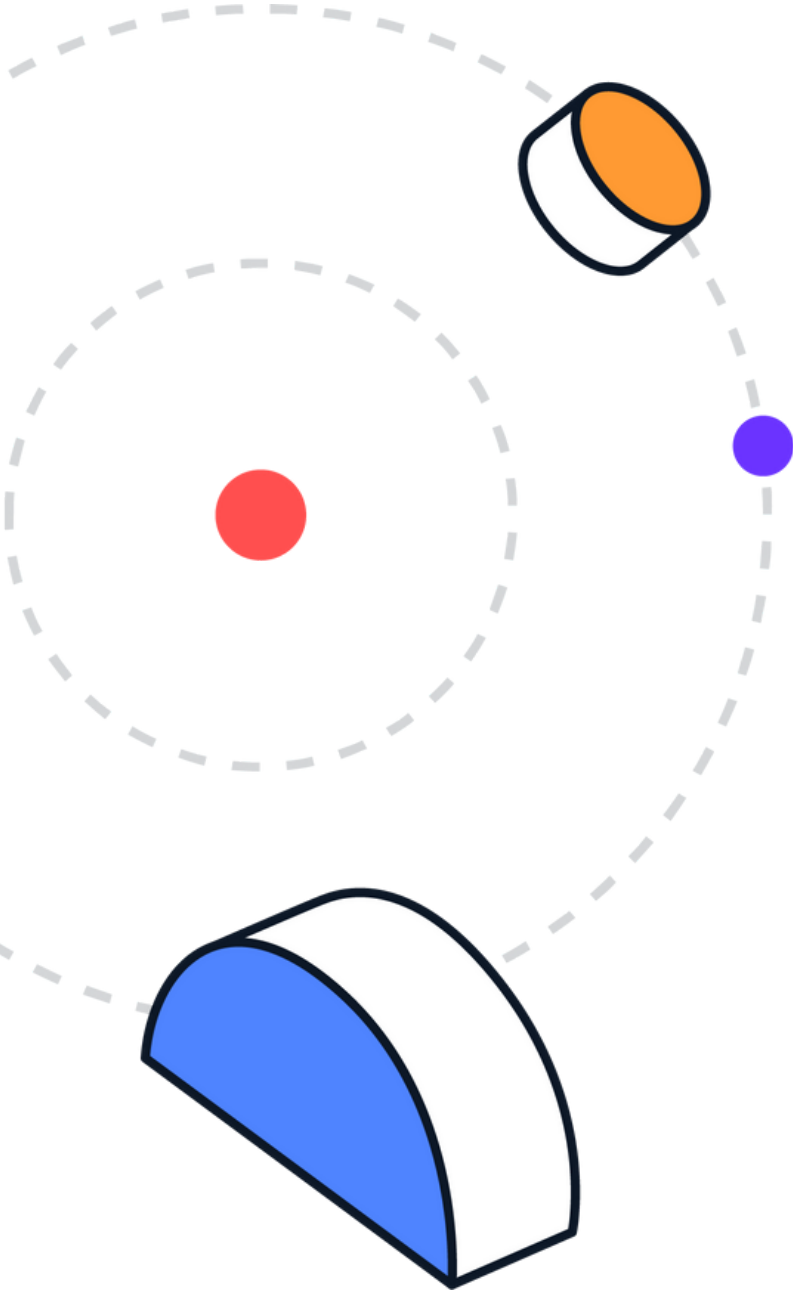
"The biggest barrier to test automation remains the level of maintenance required to sustain it."

But why is Test Automation so cumbersome?

Traditional test automation systems originated in a world that moved at a much slower pace, where waterfall developments were the only game in town and no-one attempted to tackle fast moving, mission-critical applications – they knew that the technology simply couldn't keep up.

These products all get their capabilities from powerful scripting languages; something that sounds good in a presentation but has become a horror in the real world, requiring highly skilled and expensive test automation engineers, to build and maintain the test automation framework and assets.

Other 'benefits' of a coded approach were rapidly found to be of little practical use. The theory was that a coded automation script could be developed in parallel with code development.



The truth was somewhat different as these test tools required knowledge of how the developers were naming the visual components – something that was neither consistent nor predictable.

Because of this, the code-based tools reverted to a "record' mode to establish the initial script, which made them only usable once the application was complete.

This was more practical, but now the automation coding effort couldn't even commence until sections of the code were complete and stable.

It got worse. Most of each script that needed to be coded had nothing to do with testing the application. The engineers had to overcome many challenges before they could even get that far – handling unpredictable response times, retrieving displayed data needed for validation and establishing checkpoints to signify when the application had completed a logical step.

But the death knell was what happened when the application to be tested changed. Suddenly all these laboriously created 'assets' were worth nothing and would not execute until the entire process had been repeated.

What happened next ranged from the sane to the almost comical. The sane organisations did what came naturally and gave up. Others were not to be defeated and threw even more expensive resources at the problem, some hiding the failure by outsourcing the entire test burden – often to companies who did most of the testing manually. All this for an initiative which was meant to reduce the need for resources, save time and improve quality!

To put this in perspective, industry analysts state that the high-water mark in automation success is when 20% of an application has been automated. This is the high-water mark, mind you, not the average; 20% is the peak of what you can expect after a financial investment measured in hundreds of thousands of dollars and an effort investment measured in many man years.

"With all the benefits of a more fluid, flexible process, come challenges in how to assure the quality and governance of these ever-changing application."

The Current Need for Speed

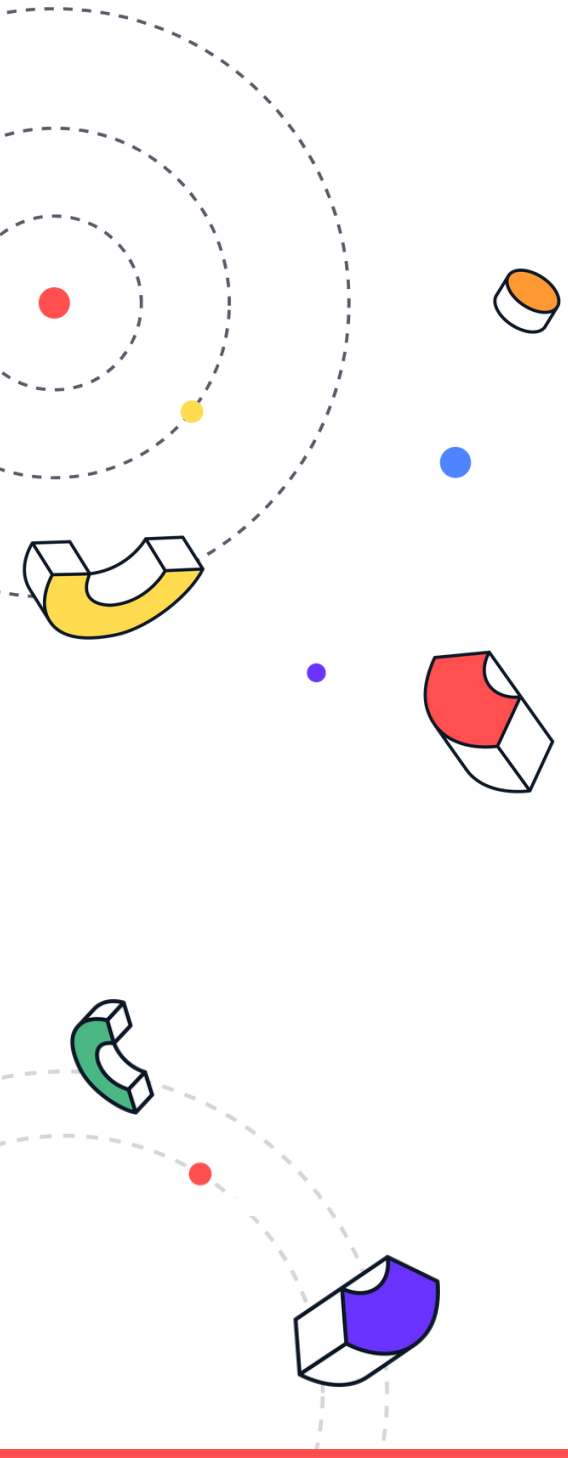
The way we work has also changed. In the last decade, the rate of business change has risen beyond anything we could have expected. The availability of new technology and the strategic advantage that it can potentially provide businesses has fuelled this, along with the need to adapt quickly to changing market requirements.

As the fortunes of markets change and move with a frighteningly sudden pace, every business finds itself needing to achieve more, with static or reduced budgets and resources.

Agile and hybrid development is now well established, although many waterfall still happily co-exist. What is common to all is that today's fast-paced business environment demands an organisation's development process to be flexible and adaptable to changing needs.

Any agile model provides frequent delivery, increased customer involvement and helps to deal with the problem of rising complexity in systems.

"How much real innovation have we had in this [testing] discipline that has actually stood the test of time? I argue that we've thrown most of it away"



QA teams now have to accept that requirements can often change during and after each iteration, depending on the feedback from the customer.

These changes in requirements are consequently reflected in the code and the tests that QA teams have to develop, which in turn can lead to a large amount of rework and script maintenance.

With delivery cycles getting shorter and with security concerns and new regulations to manage, applications are becoming more like living things; beings that grow and mature, morphing from new-born status to an almost unrecognisable fully-grown adult with all the associated trappings and documents that adults tend to collect throughout their lives.

How on earth is outdated and cumbersome test automation technology supposed to cope with this level of change and complexity? It simply can't.

Micro Focus bought HP... nothing changed

HP in a refreshing burst of honesty now states that unless you will run a script a minimum of seven times, there will not be any payback from automation. That is one heck of a statement. Any part of an application that needs to be tested at least seven times suggests an almost static application, not one that is subject of active development efforts.

This sort of automation is fine for regression tests but will not make any impact on current QA bottlenecks. The need is for a solution that is faster, lighter and better able to respond to dynamic application developments.

In short, the modern business, with all its need for speed and agility just has no place left for these types of solutions, regardless of how much organisations have already invested in them and regardless of how much resource is tied up in trying to maintain them. The need for change is now.

"Having trouble keeping up? The pace of innovation and disruption is accelerating."

So says Barry Ritholtz in an article entitled "The world is about to change even faster".

He's right. Technology changes too quickly for any one company to stay on top of it. New software is released so regularly that it is already out of date by the time it is launched– consider the frequency of SAP or even Microsoft updates; keeping on top of these pose a real headache to most IT departments.

We've got to a state where traditional testing processes and tools are too cumbersome and development is pulling away. Testing becomes the bottleneck. We don't need test assets which have cost companies thousands of dollars and man-hours to develop. What the business needs now is test assets that are quick and easy to develop, that can be re-used or adapted easily, or can be discarded without a second thought.

By freeing automation from the burden of a script based on code, we can begin to imagine a solution that could be used by subject matter experts and not limited to frustrated developers, a solution that could adapt to changes in the application under test

www.altimetergroup.net

"Think about your own test assets. Can you even estimate the cost in time and effort in building them? How constricted are you by that investment?"

So why throw such perfection away?

Now it would be foolish to dispose of the entire test automation concept purely based on what came before. We need to recalibrate our expectations and remind ourselves of excellent potential benefits from test automation if only the capabilities were delivered in a form usable by all.

The ability of any automation technology to adapt to changes in the underlying application will always have some limitations. Is it reasonable to expect a test script created for a legacy mainframe application to remain valid on the replacement responsive web architecture? Can you expect the test scripts created for the English version of your website to be applicable to the newly developed Japanese version?

By freeing automation from the burden of a test script based on code, we can begin to imagine a solution that could be used by subject matter experts and not limited to frustrated developers, a solution that could adapt to changes in the application under test, an intelligent solution that inherently understood the application under test, removing the need to develop logic in addition to the validation itself.

The exciting thing is that modern automation goes a surprisingly long way towards addressing these needs. But do not let that optimistic outlook hide the core issue – at some point, the application, environment or business will change in such a fundamental way that the existing test assets have little or no value.

If that loss represents an investment in intellectual property, resource and time at a level so large that there is no appetite to redevelop those assets for the application version, then automation will have failed. Thus we arrive at the acid test – if it is deemed easier to return to a manual test approach, then automation has failed and deserves to be thrown.

Summary

Test automation has failed to date simply because we could not afford to throw it away. Creating any form of automation takes effort and time; when the application under test changes and the automation ceases to work you are faced with a stark choice – either maintain it at additional effort and time or abandon it. If you abandon it you are also writing off the effort and time you invested in creating it, thus bringing the whole concept into question.

The reality is that you have to be in a position to throw away the automation you have created, as sooner or later the application will change in such a way that no amount of manual or automatic healing can tackle.

So, by definition, the creation of the automation must have been so fast and painless and the investment minimal, that you can afford, both financially and emotionally, to throw it away.

Think about your own test assets. Can you even estimate the cost in time and effort in building them? How constricted are you by that investment? Can you hand on heart claim that you don't grimace every time you have to throw them away?



This guide has been brought to you by Original Software. With the happiest customers in software testing, we've been helping businesses meet their objectives and deliver quality software through our range of innovative code-free testing solutions for over 25 years.

originalsoftware.com

Original Software