Whitepaper

Avoiding the Pitfalls of Test Automation

Author: CodeFuse Technology Limited

Publication Date: 20h December 2013

Introduction

The use of Test Automation in a software development project offers significant efficiency gains, but needs business preparation as it can be fraught with risk. Without this preparation, project teams can end up wasting money on tools, licences and training among other areas and have nothing (or a white elephant) to show for their efforts at the end of the project.

There are a number of pitfalls which need to be avoided if time, effort and money is not to be wasted. This paper examines those pitfalls, discuss how they can be avoided and thus find routes to success. CodeFuse Technology Test Automation tool, CodeFuse, will be used to substantiate certain points within this paper.

Test Implementation Strategies

Firstly, we will exmaine strategies. There are two main ways in which firms typically try to use Test Automation as follows:

1. Test Automation After Development (TAAD)

This approach could be considered the "traditional" one, perhaps in part because it fits in well with the "Waterfall" Software Development Lifecycle (SDLC) in which testing takes place strictly post-development.

The thinking behind TAAD is that you wait for the software to be stable enough before you start writing automated tests so that you don't need to re-write the automated tests when the underlying application changes. In theory this should be a misnomer in Waterfall as all the specification is done up front, however, this is often not the reality. Furthermore, some of the traditional test automation tools do not give you a basis in which to allow you to write tests before hand. If there is no application, how on earth will you do "record and playback"?

TAAD is supported by Test Automation tools including CodeFuse. With CodeFuse, TAAD is delivered by the different team roles using CodeFuse features in the following way. First, a Project Manager can define the project type, for example, "Waterfall" SDLC. Next, a Business Analyst can enter the requirements into CodeFuse which are implemented by Developers. Finally, after development is complete, a Tester can then translate these requirements into executable tests.

2. Test Automation Before Development (TABD)

This approach has been gaining support as industry best practice for the last 10-15 years in the form of initially Test Driven Development (TDD) and more recently Behaviour Driven Development (BDD. It aligns itself closely to modern Agile SDLC practices.

Both TDD and BDD stipulate in slightly different ways that you are going to spend far longer thinking about the definition of what it is you are going to write, rather than actually writing it. The writing becomes far easier as you know exactly what to write and you can minimise your efforts based on satisfying the pre-defined tests. No more, no less.

Again, Test Automation tools support this testing methodology including CodeFuse. With CodeFuse, Project Managers would, as CodeFuse Administrators, define the development methodology, for example, “Behaviour Driven Development”. A Business Analyst would then enter the User Stories into CodeFuse. The Tester can then expand on the User Stories by mapping out the tests (or ‘behaviours’ in correct BDD parlance) step by step. The Tester in this scenario would have to leave out the Test Object information as the underlying application hasn't yet been developed. The Developer would then write code to satisfy the tests. The tester would then work closely with the developer to update Test Objects and schedule tests while the Developer refactors code. Once all tests have passed the software code is ready to accept.

Test Automation Pitfalls and Solutions

We’ve now looked at two different approaches to how test automation can be implemented. But what are the potential pitfalls for a team when they decide to use test automation? If you hear anything similar to the quotes in bold below, you need to start asking questions within your team!

“It’s something that resides entirely with the Test Team”

Test automation needs to be considered a project team activity. This is because the team needs to think about Test automation when looking at software design, coding standards and even the ways of working for the team. The whole team needs to buy-in to the fact that a new feature is not complete until the associated test automation is complete too, in the same way that manual testing, or a code review would be viewed as compulsory. Unless the team buys in to this, design decisions will be made that make test automation difficult or more expensive, coding practices will be used which are not conducive to test automation and features will be rushed though without test automation negating many of its benefits.

“We have an automation tool, we’ve solved the problem”

Test automation solutions are still viewed by some as a mystical silver bullet which enables development teams to operate without any testers. The issue is that someone needs to use the tool. As with PhotoShop or a hammer, a tool enhances productivity, it doesn’t remove human involvement entirely.

Additionally, as we’ve already read above, you can use test automation in different ways. Are your development team aware of how the tool fits into the development process? Who is meant to do what with it and at which stage?

There is also a key question for teams to define what it is exactly that you intend to replace via your automated tests. If you are using a TAAD approach, you’ll need to map out exactly what it is you want to automate. This could be done by some form of Application coverage matrix. On the other hand, if you are using a TABD approach, you will need to get team agreement that all tests will be written to support the requirements. This will mean code development will not start until that point. Above all, you do not want to be writing automated tests which are unclear in their purpose as you’ll probably end up running the same tests manually anyway!

The development team needs to put aside time to automate, and ideally automate-as-you-go whether you are using TAAD or TABD.

Finally, time needs to be put aside to learn how to use any tool. It is worth looking at tools which abstract the complexities of test automation away, so that this time is minimised.

“We have found a tool that is zero maintenance”

Aside from edge cases, this is highly unlikely, and the belief will store up problems for the future. Even if the tool were perfect, your application interface will most likely change over time, and your tests need to reflect that. Changes may be localised and most tests are likely to remain untouched, but there are likely to be some maintenance tasks, however small.

"The testers can just run the tests on their laptops"

Having a dedicated place to run your tests is a must. Being reliant on machines which have multiple purposes is a sure fire way to resource conflicts, unreliability in your tests and a fragile process. Unreliability in particular is a guaranteed way for team members to lose faith in the tool and test automation in general. Tools which leverage virtualised environments offer a clean resolution to this problem.

Summary and Conclusions

As we have discussed in this whitepaper, the use of test automation needs to be managed as a part of the software development process to enable it to be effective. The key points raised were:

- Get genuine involvement from the whole team early in the project.
- Ensure that people are trained in the tool, ideally making use of an easy to use tool.
- Ensure that all team members are aware of the points in the development process where the tool fits in.
- Ensure that you are very clear about exactly what it is you want to automate.
- Plan for time to use the tool as a part of your process, don't take a fire-and-forget attitude.
- Ensure that you have invested enough time in getting a reliable environment to run your tests.

The Test Automation Before Development (TABD) approach is the most modern and has a growing body of supporters, but whether this approach is right for you, depends on your organisation.

Test Automation is a realistic goal for software development teams. It needs to be managed correctly to be given the chance to succeed, but if you can do that within your team, the benefits are significant.

About CodeFuse Technology Limited

CodeFuse Technology produces tools for the software development industry. We are focussed on improving the quality of your software by enhancing your collaboration, simplifying your processes and streamlining your QA efforts through automation. We provide cloud based Software-as-a-Service (SaaS) solutions which are designed to minimise your efforts in set-up and maintenance so that you can concentrate on getting the job done. We were established in 2013 as CodeFuse Technology Limited.

Contacting CodeFuse

CodeFuse Technology Limited 33 Cross Deep Gardens, Twickenham, TW1 4QZ	Tel: 0207 873 2431	General: info@codefuse.io Support: support@codefuse.io Sales: sales@codefuse.io