L & T Infotech | White Paper

PROJECT ID: E9953

Adopting Automation in Agile Testing

An L & T Infotech Agile Testing White Paper

January 2014

Deepti Waila
Garima Sharma
Mohammad Abuzar

Executive Summary

Agile test automation may sound like a contradiction in terms, but it is a new testing methodology that can revolutionize the world of application testing, bringing with it order of magnitude gains in time, cost and risk management.

Traditional automated testing methods, such as record/replay tools, provide a useful, but rigid approach for application testing. These tools generate custom script code that must be developed and maintained along with the application under test. Conventional wisdom holds that these tests should not be automated until the application is stable, resulting in a primary focus on regression tests.

In general, traditional record/replay tools do not lend themselves well to automating tests in a rapidly changing environment. The cost of maintaining test scripts for an application with a high rate of change would exceed the benefit gained from automated execution. However, a newly developed methodology—Agile Test Automation—is providing more flexible test management strategies.

What is Automation in Agile Testing?

The objective of any software testing is to check that it works before it is shipped to the customer, and that it meets the business needs. In an Agile environment, the customers work closely with the development team and there are a number of software releases before the final version. On an Agile project, the purpose of testing is to uncover any errors so that these can be fixed and to check that the release works the way it was intended to.

Agile testing may have similar aims as traditional software testing, but the structure of the team is different. Testers are normally embedded in the Agile project team, working alongside the developers and business users to ensure each release is tested early and often. Agile test automation can improve the probability that important defects get reported in a timely manner

As projects have multiple releases which each need testing, Agile teams make use of test automation as a way to cut down testing time. The more the team can automate the testing, the faster they can move on to the next developments—so automation can cut delivery times and remove some of the mundane, repetitive work, leaving your skilled resources to focus on the more difficult and valuable tasks. It is impossible to automate everything, so this also frees up more time to spend on manual testing.

The Challenges in Automation in Distributed Agile Testing

Agile teams agree that testing is an essential part of a software release, but testing in an Agile way is not without its challenges. And as Agile teams work differently from other software teams, they need to adapt their ways of working to manage the challenges. Here are few challenges teams face.

1. Time Constraint

In a span of two weeks (iteration duration) the testing team has to relay the information across locations.

2. Tools and Frameworks

The challenge comes in adequately defining the type of tool to be used; this can cause problems for the quality of the end product and additional tasks for people who are not fully skilled in testing using those tools.

3. Knowledge sharing

Teams have to find a suitable way to share this information that works for them. It has to show who is allocated to what, what development and testing tasks are in progress and how this compares to the overall progress expected as they get closer to the end of iteration. Keeping this information relevant and up to date is a challenge and can lead to an increased management overhead.

4. Test Data Management

In a traditional project, there is normally just the one testing phase. However, the iterative nature of Agile means that there are multiple testing phases as each iteration requires testing before being released to the customer. This means Test Data is required to be changed after every Iteration.

Spending too much time on changing test data can lead to issues with team morale, slow progress and communication challenges with senior managers and business users. Without proper test planning, and test data management repeated testing phases can bloat the project and slow progress down.

5. Reusability

The test cases created for one environment should be reusable for the different environment. In Agile projects teams are required to test right from low level to production environment; so creating separate test scripts for separate environment can be time consuming and redundant.

6. ROI

The selection of the frame work and type of planning can raise a serious question of obtaining the required Return on Investment.

5 Ways to Streamline your Agile Testing

Faced with these challenges, it can be daunting for a new member of an Agile team, or a new manager leading an Agile project, to get to grips with Automation. There are ways to address these issues and to ensure that the software your project is producing is the best possible quality. If the challenges above are not addressed, you risk ending up with a project that is poorly planned, poorly scheduled and bloated—not very Agile at all. Here are five ways to streamline your Agile testing to maintain your team’s flexibility.

Agile automation progresses according to this cycle: Understand how the testing is done, identify some technology that would significantly improve it in the opinion of the tester, deliver that solution in less than a week, repeat.

1. Ensure team cohesion

Automation Testing is no longer the responsibility of one individual. Consider Automation testing the responsibility of everyone on the project.

While you may find that having multi-skilled individuals on the team helps progress things faster and in a more flexible way, having each person’s main role clearly defined really will make a difference. Individual team members should be clear about the contribution that they are expected to make, and the roles that others in the team are making as well.

Making sure that the programmer is responsible for only script writing and others are responsible to execute the script and fix any bugs in the script can be fixed by other team members. As a result, everyone should get involved with testing early and often, drawing on the specialist testing resource and tools in the team as necessary.

2. Ensure traceability

Make sure that you are testing what is important to the end users. Each test should link back to a user story to ensure that both the tests and the functionality that is being tested meet the user requirements.

The manager can also request the team members to create Capability wise test cases so that each test case can be traced back to every component of the Application.

3. Maintaining Test Data

Test data can be stored in an excel and reused by the test script and calling the source of the excel. This excel can be placed in a common shared folder or can be updated using any configuration management software like SVN

As Agile projects are incremental and iterative so the test data is likely to change more often as compared to other SDLC project; hence programmer can in corporate database queries in the testing script so that there is less test data to be maintained in the excel. This can cut out time spent on storing data in excel and updating it every now and then.

4. Use pair testing

Having two people working together can mean a sharper focus on testing, bug discovery and issue resolution. The tester in the pair can ask questions from the user's perspective, so the developer can see the impact of the code from a different angle. Pair testing breaks down the silos between testers and developers. It also helps both parties link seemingly random bugs into a pattern, as both individuals will bring specialist knowledge that can uncover more errors than if they worked alone.

The Distributed teams can create their automation test cases after their discussion with their On-Shore counter parts. This will reduce dependency on a particular programmer

Doing automation in the same iteration with distributed locations will save the regression manual efforts. Paired and team wise automation testing makes more productive, faster, or reliable the kinds of useful testing already being done, or otherwise contributes to a more successful and cost effective test project in long term.

5. Use layered tools

Agile projects are iterative, and each release needs structured testing. As a result, there are more testing phases in Agile projects than traditional, waterfall projects. Each time a new release is tested, the team reviews the code from the previous phases to ensure that everything still works as intended. Many of these testing tasks are repetitive.

At the end of iteration, you should know how many tests were carried out, the defects to be rectified and their severity. Some types of testing can be run almost continuously (for example, some integration tests). Automating these means that code can be checked overnight with the results available for the team in the morning. Test tools can help collate this data and track the metrics against previous iterations: the trend, of course, should be downward.

The best test management tools are flexible enough to interface with other products and to support Agile ways of working. It is really important that the tool you choose fits well with your current infrastructure and suits the needs of a multi- functional team to ensure collaboration and control.

Test management tools can also help with test automation, both by running tests and by storing the results of automated tests.

In our project we are using HP-QTP v 11.0 and it is merged with HP Quality Centre which performs the above mentioned tasks.

Moreover we have utilized a Framework known as TRAVTest that can effectively perform the test case configuration management and automation tasks

About TRAVTest

TRAVTest is a tool designed by Travelers to generate the automation test script easily and run them effectively across platforms.

It has feature to initialize objects and during the run time those objects combine together to form a perfect syntax.

These objects tend to change after two or more iteration as new objects are added as we progress so TRAVTest allows the team to expire an object and then again renew that same object once the two iteration old code which was deployed in lower environment, is now deployed in UAT environment.

This ensures configuration management and reusability of the previous configuration.