Borland® | White Paper

A MICRO FOCUS COMPANY

Adopting Agile Testing

A Borland Agile Testing White Paper

August 2012

Executive Summary

More and more companies are adopting Agile methods as a flexible way to introduce new software products. An important part of any software project is testing. In this white paper, you'll learn about the challenges of Agile testing and five ways to streamline your testing to ensure a quality result.

It's a common misconception that Agile projects don't need a rigorous approach to testing. The testers may be embedded in the project team along with the developers and business representatives, but this doesn't mean that you can avoid a structured, documented approach to carrying out system testing.

Using a test method with clearly defined metrics and analytics helps provide complete traceability of requirements, which in turn means that you can ensure that an Agile project is fit for purpose.

What is 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.

As projects have multiple releases which each need testing, Agile teams also frequently 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 of 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 five challenges teams face.

1. It's difficult to keep everyone on the same page.

Agile teams need an intuitive way of working to allow the planning and scheduling of activities. This is essential to the smooth progression of an iteration as the team is largely self-managing, so it's important that the required tasks can be planned appropriately. They also need to quickly see who is doing what, at a given moment, and what is being tested. At a higher level, they also need to fully understand the day-to-day progress.

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 an iteration. Keeping this information relevant and up to date is a challenge and can lead to an increased management overhead.

2. Test-driven development can be unpopular.

One approach to Agile testing is test-driven development (TDD). This is a practice where the tests are written before the program code, so the developers design the code for testing. In an ideal world, this guides the development so that the code is built to specification and testing then becomes more straightforward.

However, research from VTT Technical Research Centre of Finland shows that it can be difficult to get teams to adopt this way of working. There is an overhead before development begins to produce the tests, and the researchers found that it was difficult to get developers to both code for the tests and also run the tests.

Developers often like (and need) the flexibility to code the way that works for them and the software, so they can be reluctant to adopt any method that constricts this.

3. Lack of clarity around the role of testers.

A challenge facing managers of Agile teams is the question: Do we need testers at all? If developers are writing the test code and members of the project team are multi-skilled, managers often question the need to have dedicated testers. Can't developers do it all? Unfortunately, this is rarely the case.

Developers are highly skilled, but even if they are prepared to test the code that they have written, they are probably too close to it to do so objectively.

The challenge comes in adequately defining the role of a tester. In teams where the testing role is not valued, this can cause problems for the quality of the end product and additional tasks for people who are not fully skilled in testing practices.

4. Unit testing alone is not adequate.

Testing on Agile software projects is often very iterative. The maxim "test early and often" means that there isn't always a defined testing phase. Testing teams then face the challenge that unit testing—testing the individual components—will be good enough. Developers and managers can hold the view that an iterative development process will sort out the bugs inherent in the system. Unfortunately, without structured testing (even following the iterative model), unit testing alone is unlikely to deliver a quality end result.

5. Agile projects require multiple test cycles.

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 the testing overhead can be more substantial than in waterfall methodologies, and tests have to be run multiple times.

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

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 testing. 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.

1. Ensure team cohesion

Testing is no longer the responsibility of the Quality Assurance team. Consider 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. For testing purposes, this means clearly defining the role of a tester. For everyone else, it means ensuring that supporting testing—and even carrying out some testing—is part of their job description.

Making sure that all the team members are based together is also good practice and it will help the team work more closely, especially if some of them come from non-Agile backgrounds.

Teams work most effectively when they have clear goals. Ensuring that everyone knows that a quality outcome is an essential, and expected, part of this project is key to getting the result you want. 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.

A user story is a couple of sentences in non-technical language that describes how users will use the new software to do their jobs. User stories define the functionality of the system and are often written on small cards that can be pinned to a notice board. They form the basis of the requirements for the software, and are easier to produce than long, formal requirements documentation. As you don't need to spend much time producing them, it is quick to update them when the requirements change, as often happens in a fast-moving IT environment.

Testers, in conjunction with business users, can use the user stories to establish what needs testing. When the development of a piece of functionality based on a user story is complete, tests can be carried out that check whether or not that functionality fulfils the requirements of the user story.

This acceptance testing ensures that the test links back to requirements that are important to the end users. Using an automated system allows you to link assets and gain automatic traceability. Of course, you can manage these links manually. However you choose to do it, traceability relies on being able to match tests with user stories to give everyone confidence that the right things are being tested and that the objectives of the iteration will be adequately met.

3. Carry out risk-based testing

Risk-based testing is about being able to apply risk factors to the project requirements and use cases to determine where you should focus your testing effort. Agile projects work to fixed deadlines, and it is impossible to test everything as thoroughly as you would perhaps like. It's also costly to let the testing phases run on and on. As you can't test everything, risk-based testing helps you decide what to test, in what order.

The priority list comes from applying risk factors to the test itself and the requirements being tested. An automated testing tool allows you to schedule tests according to risk and quality factors so that the most important tests are done first. This allows the team to focus on any areas of high risk, ensuring that they have enough time to remedy any problems.

Another element of risk-based testing is for the business users to identify what risks there are to the project and the critical functions of the software. Test cases can then be developed around these to ensure that vulnerabilities and risks are checked.

For example, say one risk of a new piece of financial software is that people could maliciously log into another person's account. A test case would be created about ensuring that the login mechanism was appropriately secure. The test case would then be run to ascertain if the risk has been averted. In this situation, the tester would check that the security protocols around users signing in to the new system were robust enough to stop the wrong person from accessing a user's account.

Get the team all together to brainstorm the risks associated with this project and this piece of software. Risk identification is not something that the tester (or any other member of the project team) should do in isolation; everyone will have an opinion and something to offer to a discussion on project risk and the business users will no doubt have significant contributions to make around which features are business critical.

Once the risks have been identified, define the risk mitigation requirements that match each risk and then the quality criteria associated with them. In our example, the quality criteria could include things like two-factor authentication, the requirement to have complex passwords that expire regularly and so on. When you have identified your quality criteria, you can select an efficient set of tests that will help you categorically prove that the risk has been mitigated—or show you areas where the tests fail and the developers need to carry out further work.

4. Use pair testing

Pair testing is where a developer and a business analyst or test specialist work together to carry out testing. Test management software can be used to allocate test resources to test cycles during an iteration to make sure that you have the right people available together for a round of 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.

While many tests can be automated with the right software, pair testing cannot. This type of testing is exploratory, and will evolve organically from test cases as the pair uncovers errors. It is high-level testing, and as a result can generate test cases for use at a lower level, either on an automated basis or for use with manual testing. Pair testing alone won't identify all of the problems, so you can still use automation software for the majority of structured tests or carry out further manual testing to cover all the areas of risk.

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. Forrester recommends automating the "mundane" tasks because on an Agile project the testing time is compressed. The more automation you can adopt, the more your project team can focus on the tasks that really matter—and the essential tests that cannot be automated.

Layered test management tools bring together various disparate tools and manual testing methods across unit and functional testing, to provide a cohesive view of the testing phases. Developers may have their favorite tools to use, but a good test management tool can link these together to provide a centralized view of the software testing.

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

At the end of an 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.

Recommendations

The move to Agile software development helps you to test very early in the lifecycle and to produce incremental software deliveries that can be shared with customers on an iterative basis. Test automation and a mature test management tool can help project teams deliver more effectively, and in shorter timescales. Even if a lot of your testing is carried out manually, test management tools help streamline the testing overhead. The project team will deliver faster and in a more flexible way with less administration required. All testing progress reports can be produced automatically and in real-time, allowing the team to review the project progress and check at a glance that testing is on track. With more time to focus on the development and the needs of the users, your highly skilled project resources will avoid wasting time on tasks that test management software can do for them.

In order to reach the higher levels of maturity in the test process improvement model, Test Maturity Model Integration (TMMi), testing processes need to be properly measured and analyzed. Test management software can catapult Agile teams to the next level of testing maturity by providing the data, quality control and peace of mind required to advance testing practices at the organization. Whether you choose a test management tool or not, it is important to choose the right level of testing for the project. You should take into account the size of the team, the experience of the developers, the tools available, the testing methods in use and the support available. Larger projects will require more testing and more management overhead structuring the testing activities.

Essentially, robust, practical and mature testing methods and skilled testers will ensure that you deliver the best possible software, release after release.

About Borland

Originating in 1983, Borland Software Corporation is a Micro Focus company. The Borland brand identifies the requirements, test and change management solutions which help companies to build better software, faster.

Our world class software development products work across the entire Application Development Lifecycle to transform good software into great software. Our tools are open, Agile, and fit for enterprise. They:

- integrate with our customers' infrastructure and leverage open source
- work with our customers and bring new methodologies to existing processes
- scale appropriately across an enterprise

Our vision recognizes that developers and development organizations need to define, manage and measure the software delivery processes based on their unique needs, tools and preferences. We partner globally with a broad range of leaders in technology, services and distribution to deliver that vision. Our partnerships enable us to help organizations make good software great, using the processes, tools and platforms of their choice.

About Silk Central™

Silk Central is an open test management solution that reflects the way today's organizations work. As a scalable, flexible test management engine, it ensures that development teams deliver higher quality software into the hands of their users faster, ensuring greater value in a shorter timeframe.

Optimized to test both Agile and traditional projects, Silk Central speeds up test execution and increases collaboration on quality assurance activities. Silk Central manual execution planning supports Agile development through progress reporting, burndown charts, planning and personal dashboards. Silk Central is deeply integrated with Rally®, the leader in Agile project management.