September, 2009

testing experience

The Magazine for Professional Testers

printed in Germany
print version 8,00 €
free digital version
www.testingexperience.com
ISSN 1866-5705

Agile Testing

A way to get better software

by Henrik Andersson

Agile development has during the past years become more and more popular with many organisations involved with software. Unfortunately, however, many organisations tend to forget the QA department when developing software using Agile processes. I have worked as a consultant for many different companies and organisations where they *talked* about how 'agile' they were when developing software. If we scratch the surface, however, their development processes often turn out to be just another classic waterfall where the developer gets a bunch of requirements or features to implement before an end date, usually several months ahead. This gives us long development cycles.

The only real elements of agile in use are the daily meetings in which they discuss what has been achieved, what is outstanding and any impediments encountered. When the developers have implemented the requirements, the software is handed over to the QA department to test the software in a classic manner.

At the end of the cycle there is usually a release, a milestone taking pressure from two sides: On one side are the developers who want their new features in the release, and on the other side is the QA department that is interested in getting the bugs fixed before shipping to customers. Clearly, these two forces do not push in the same direction, which can lead to hostility between the two groups. To complicate the situation further, we then have the product management, marketing and sales pushing to get the new features onto the market as fast as possible, even if this means a reduction in quality. We end up with the business side butting heads with R&D.

This scenario describes a very common way of developing software today.

What is agile?

Before we get ahead of ourselves here, let's take a closer look at the agile manifesto, which was written in February 2001 by seventeen people that needed an alternative to the document-driven, heavy-weight software development processes that were predominant at that time.

What makes a process agile are the following values:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following plans

One of the principal misunderstandings of the agile manifesto is the part regarding documentation. Many think that agile processes are all about coding and not documenting anything. The agile manifesto says nothing about not writing documents, and many people in the agile community want to restore the credibility lost to the methodology due to this misunderstanding. It is about finding a balance between what documentation to write, and what *not* to write. In many companies large amounts of documentation is written, stored in some repository, and then left where it will often remain untouched, unread and (even worse) unmaintained.

Two of the twelve principles behind the agile manifesto are:

- Working software is the primary measure of progress
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter time scale

So we should deliver working software within a couple of weeks if we want to be agile. In short, we have to program and test the new features in very short time scales. Having both the development and testing teams work as one is a must and an accepted fact in the agile world.

The agile manifesto talks about cooperation within the team and that the team should have the flexibility to change direction when required. I have worked with many organisations seeing a connection between the agile manifesto and how they develop their software. But when it comes to testing, the connection is lost, and they stick to having a 'phase' between the point when development stops and the delivery of the product to the customer. This phase is the test phase where the QA team do their 'stuff'.

Classic testing

I am using the term 'classic testing' to make a distinction between agile testing and testing as many of us know it. Classic testing is a process that starts with planning the test, creation of test cases in form of written instructions, manual execution of the tests, creation of test reports, and finally evaluation to see if more testing is needed.

This process can be started as the project starts up, but often has no or minimal involvement with the development process. Rather, it forks off on a parallel track to the development.

Why doesn't this kind of testing work in a true agile project?

Let's take an example. In an agile project there is continuous integration, and we get a new piece of software every morning when we come to the office ready for being tested. If we are to follow the process taught by the ISTQB, the first thing to do would be to plan the test. After you haves planned, you will come to more insight regarding the test coverage. Are the existing test cases enough? Do we need to design new test cases for the new functionalities? If you decide to design new test cases, you have to do that before starting to test the new version of software. Please don't forget your regression testing, so that you know

whether previous features or requirements still work.

As the software grows, both in complexity and number of features, regression testing will consume more and more valuable testing time. To keep up the same pace as before, you can go to your manager to get other people from the team to help you, but this will slow down other parts of the team.

As my beloved testing friends try to say that we don't need to run the regression tests each day, run them as a last overall test at the end of each sprint. So don't we then have a classic waterfall, where we do development and then test?

By testing this way, you will have a long start time before you can start to test and a lot of overhead time in which to plan and design your test cases. However, we want to have as short a start time as possible. This cannot be done by this approach of testing.

How to test in an agile project?

Now finally the big questions: How can we make testing agile and what is agile testing?

The first thing, from my perspective, is that testing has to become a natural part of an agile project and not just a phase after the developers have stopped coding and before customers receives their software. It's about reaching the same goal as a team, i.e. to deliver working software within as short a space of time as possible.

With this short time scale, we as testers need to change our working methods. We need to step away from the nagging position we have in a waterfall process and become more active in the process of developing the software. Active does not mean that we should be sitting down hammering in code, but we should be involved from the beginning and be giving the developers feedback instantly on how the new feature works, instead of complaining at the end when everything is built.

To make this possible, we need to have frequent integration of new features, for example through nightly builds, where every morning there is new software ready to be tested. The testing should be performed manually as an exploratory test, since this is a rapid way of testing instantly, no extra documentation, no extra work, just the minimum effort required to succeed, namely testing the new features.

But how can we possibly do regression testing when we have no test cases?

When I talk to fellow testers who are into classic testing, we soon run into disagreements:

- We need the test cases for regression testing later on when the product has come in to a maintenances phase, they say.
- Let's spend time to automate these test cases, so we don't need to spend time running them manually later on, I reply.

Let me explain how I see things. Assume that we have a software project that is at the beginning of its life cycle. Not too complex, the GUI has started to fall in place, let's say it is a web application. We need to start to test automatically as soon as possible, which can be easily achieved using a tool that records our clicks and keystrokes. After each nightly run of the automated tests, we then add new test cases the next morning. This means that we get some extra time to add new automated test cases, or do some exploratory testing, since we don't need to test the existing ones manually.

The profile for an agile tester is not the same as the profile of a classic tester. I see the agile tester as a person with a more technically orientated profile, a person that enjoys both developing, debugging and testing. This profile is wanted in agile projects, since it will help the developers to find where the bug is, instead of just pointing out the symptom of the bug.

In summary, in agile testing you need to start to execute tests fast. Having working software early is a key aspect, and this means you have to have testable software early. Exploratory testing is ideal for you to be able to make a fast evaluation of the software with a minimum of preparation, and experience has shown that it is also very effective for finding problems in the software. Critical parts of the software should never fail, and you need to make sure that they are not broken in the fast development cycle. Because of this, it is worth putting effort into implementing automated regression tests.

As an agile tester you must be able to tackle these things, which calls for an interest in technology and a deeper technical understanding. Programming and debugging are key skills for the agile tester.

Biography

Henrik Andersson is ISTQB Certified Tester and test consultant at Testway, Malmoe, Sweden. His main focus is on automated tests in agile projects and on exploratory testing. After studying software engineering at the University of Skovde, Sweden and writing his diploma thesis on low-level testing, he started as a C++ developer at Tern System, Iceland. Henrik is also one of the founders of a test community in southern Sweden. In his spare time he loves riding his mountain bike and dancing "Lindy Hop" (better known as "Jitter Bug").