Preventing Defects over Finding Failures

Here are some of the topics related to agile testing that we'll cover in this tutorial.

In agile, the litmus test is whether we produce potentially shippable code every iteration. Often, when teams start agile, the tester works pretty much as before, but within much shorter timescales. This is the mini-waterfall pattern – where each iteration is like a small waterfall project. Trouble is, if the tester is doing their testing at the end of the sprint, and are dedicated to a team, then they are probably the bottleneck and will come under a lot of pressure, with little to do at the beginning of the sprint, but too much to do at the end. This is a form of waste – unevenness in flow and unreasonableness. Also, what if we discover a problem, a bug? Its probably too late to fix and retest, so that story or piece of work can't be considered done, and we can't demo it or take credit for it.
So to work in an agile team, testers need to approach their jobs in a very different way. This change includes mindset, skills and practices:

Mindset:
As a tester, you probably acknowledge that we can't 'test in quality'. Testing finds defects and gives us information to help assess the quality of the product – it doesn't on its own improve the quality or eliminate bugs. So just finding bugs after they've been created is not enough on its own to attain quality. It's the debugging process that improves quality. And developers do debugging. Therefore, if we, as testers, want to contribute to improving quality, we need to work alongside testers, to collaborate with them. But in agile teams, we don't have time to do this after they have created, and we have found, bugs. Instead we need to work with them right throughout the sprint, from the very first day, to prevent bugs – to be proactive in helping developers not create bugs in the first place.
But testers do testing – this doesn't sound like testing! Well, it's all in the timing – in traditional project environments, testers usually get involved AFTER the code has been written, the UI designed, etc. They bring an independent interpretation of the system requirements, and design and run tests to validate that the system provides the necessary functionality, and verify that it does this with an acceptable level of reliability, performance, etc. But one of the ways testers on agile teams work is to design and implement their tests

BEFORE the code is written, even before the design is complete. Then they can collaborate with developers by testing what is developed on a continuous basis, guiding the developer to an acceptable implementation by making the tests themselves act as the requirements to be met – effectively providing 'executable requirements'.

But what of Tester independence? If I work with developers closely won't I be influenced by their interpretation of the requirement? Won't I end up just testing that what they build does what they intended, rather than what the requirements expressed that it should do? This is a danger – but one that can be avoided by another aspect of agile testing – the tester working proactively with the business or product owner as well as with the developer – in fact, before they work with the developer. I agile teams, the tester is key to bridging the communications gap between the business and understanding of the business problem to be solved, and the IT folks who are tasked with finding a solution. So testers need to concern themselves with understanding what a requirement is, helping the business or product owner define acceptance criteria and developing acceptance tests based on these, from the very earliest stages. This is what we call a Test First approach. We focus on fully understanding what a requirement is, or a feature should do, before we begin to implement it in code. The discipline of designing and implementing tests BEFORE we start developing a solution ensures that we fully understand WHAT should be built before we engage the developers in designing HOW it should be built.

The mindset change for an agile tester is that they are not just responsible for designing and running tests after the development work is complete. Nor are they just involved in early static testing of documents and designs before development. Instead, they are involved in the earliest discussions with the business, understanding the requirements, helping the business define what will be acceptable, and implementing tests that can be executed by themselves and, potentially, by developers, to ensure what is built is of high quality from the beginning. The mindset must change from 'I'll be seen to be a good tester if a find bugs' to 'I'll be seen to be a good tester if I don't find bugs (because there are none – I've made sure of it). This is the idea of 'Building in Quality'

Skills:
Testers bring many skills to a waterfall type environment: an independent, critical and unbiased mindset to the testing task – they have not been intimately involved in the design and development of a solution and may see flaws that others who have been involved in this work are blind to. They will approach the testing task with the presumption that there are bugs to find – that misinterpretations and other errors have been made. They will assume that not all possible interactions, inputs and situations have been considered and that there will be combinations of these where the system will fail. They will probe, putting the system under pressure in various ways to try to 'break it'.
All the above skills are still valid for testers working in an agile team – however, rather than applying these skills independently after the code has been written, the agile tester brings these skills to a collaborative, multi-disciplined 'all at once' development team.

Practices:
Here are some of the ways agile testers apply these skills in this new environment:

- Test First - Defining tests which can validate and verify features before the solution is designed. This maintains tester independence but applies it much earlier in the development lifecycle
- Automation – understanding how to automate effectively, avoiding overlapping tests, deciding what needs to be included in regression tests and where exploratory tests should be used. Working with developers to build and deploy automated test harnesses.
- Working with the business directly to validate requirements and define user stories and acceptance criteria, and thereby influencing the requirements themselves. Identifying concrete, precise examples of use using techniques such as Specification by Example.
- Working with developers to ensure easily testable code
- Performing effective exploratory & regression testing to ensure fast bug detection and sustainable quality levels with minimal cost of delay