Essay 1: Overview of Agile Testing

Lisa Crispin and Janet Gregory explore software testing from an agile approach in their book More Agile Testing (2014).

What is Agile Testing, anyway?

In agile development, development works in shorter cycles, delivering smaller units of business value each cycle, compared to a more traditional development approach where testing is left as an activity at the end of the development process, such as in the Waterfall model. We have previously discussed whether testing improves quality and noted that it does indirectly. It is easy to see how leaving all testing activities to the end of the development process makes it impossible to improve the quality, as bug fixing costs increase during the development process. An iterative process enables earlier incorporation of insights gained during testing. Technically, almost all elements of a traditional software development life cycle are used in iterations, but activities happen simultaneously.

The chapter also explores the roles in an agile software development process. Both testers and programmers are developers, and quality is a shared effort within the team. In addition to developers, a customer team works more directly with the customer. Even if testers are part of the development team, they work closely with domain experts and the different stakeholders.

Crispin and Gregory emphasize that testing is not limited to testers in agile teams, but imply that it is beneficial to have designated testers. All levels of testing help us understand the application that is being built. This chapter visualized what testing in agile teams might look like, and it provided a sobering perspective on the importance of designated test experts in the context of James Bach's conference speech for the warm-up essay. Bach argued that early agile philosophy made some companies question the need for testers, as testing is a shared responsibility in agile.

Ten Principles for Agile Testers

This perspective of the importance of designated testers in agile teams is elaborated on in this chapter. An agile tester is a team member who drives agile testing, sees the bigger picture, and understands the customer perspective. The focus is not on policing programmers and protecting the customer from bad code, but on sharing knowledge and collaborating with different stakeholders, not limiting themselves to activities that are traditionally deemed as testing.

- **Provide continuous feedback**: Testers' traditional role as information providers makes them invaluable in agile teams.
- **Deliver Value to Customer**: Limit the customer team's desire for new features by recognizing change impacts to stories and testing repercussions. Automate tests for the "happy path" and identify test cases beyond it.
- **Enable face-to-face communication**: Find unique ways for communication and identify key persons. Help bridge the gap between developer and customer languages.
- **Have courage**: Automated regression suites give programmers confidence to make changes in the code. Dare to get out of the silo and to endure failure.
- **Keep it simple**: Learn how to test just enough – even test automation suites should be limited.
- **Practice continuous improvement**: Look for new tools, skills, and practices. Learn new skills and grow as a tester by networking and studying.
- **Respond to change**: Adapt to changing requirements; preparing in advance might be a waste of time. Automate what you can.
- **Self-organize**: Collaborate with your team to find better practices.
- **Focus on people**: True agile philosophy gives team members equal weight – testers are an important part of the team.
- **Enjoy**: Tester's work adds real value to teams – enjoy it.

Agile testers should ask questions of the customers and developers early on and often; the answers are shaped to tests. Testers should understand multiple perspectives, such as business, end user, production support, and programmer. Testers help customers be clear when requirements are formulated, so that they can be turned into tests. I've specifically underlined the "tests drives development" part of the chapter, as it summarizes the lightbulb moments I've experienced reading these chapters. I think it is easy to see that testing is used to make sure we have built the right product, but these chapters have deepened my understanding of how testing affects different stages of the agile development process.

The Agile Testing Quadrant

This quadrant helps us tackle the question of when we are done with testing and how to structure the testing.

- **Quadrant 1: Test-driven Development (TDD)**
  - Writing tests that help programmers write their code well with automated unit and component tests. Internal quality.
- **Quadrant 2: Support Developers at a Higher Level**
  - Focuses more on customer/business-facing tests and external quality.
  - Tests are derived from examples from the customer team, describing details of stories written in business domain language. Tests illustrate and confirm the system behavior.
- **Quadrant 3: Does the product meet the description?**
  - Make sure that the business-facing tests are the correct ones – critique of the product is necessary. How do real users use the product? Exploratory testing is essential.
- **Quadrant 4: Critique product characteristics**
  - Easier to perform this type of testing if we know nonfunctional requirements in advance, such as performance and security.
  - Information from these tests should be used to create new tests in the first quadrant.

The quadrants are not ordered from one to four, but help teams and testers plan and execute tests. Combining tests from all four categories is needed to know when we are done testing.

All sections were interesting and provided new insight, and many "of course, it has to be like this" moments while reading. I was pleasantly surprised that test automation was mentioned so often, as I am looking into automation as a starting point for a software tester career. From previous courses, I have gained insight into unit and component testing, but I noticed I have a lot to learn in the second quadrant – probably with tools like Selenium and Robot Framework.

It is also surprising that critique, as a term, is often viewed negatively, but it might be that understanding the role of critique in an improvement is not as natural for people without an art background.

These chapters helped me understand the multifaceted role of a tester in agile teams, a perspective that often seems to be omitted when discussing agile development in some courses.