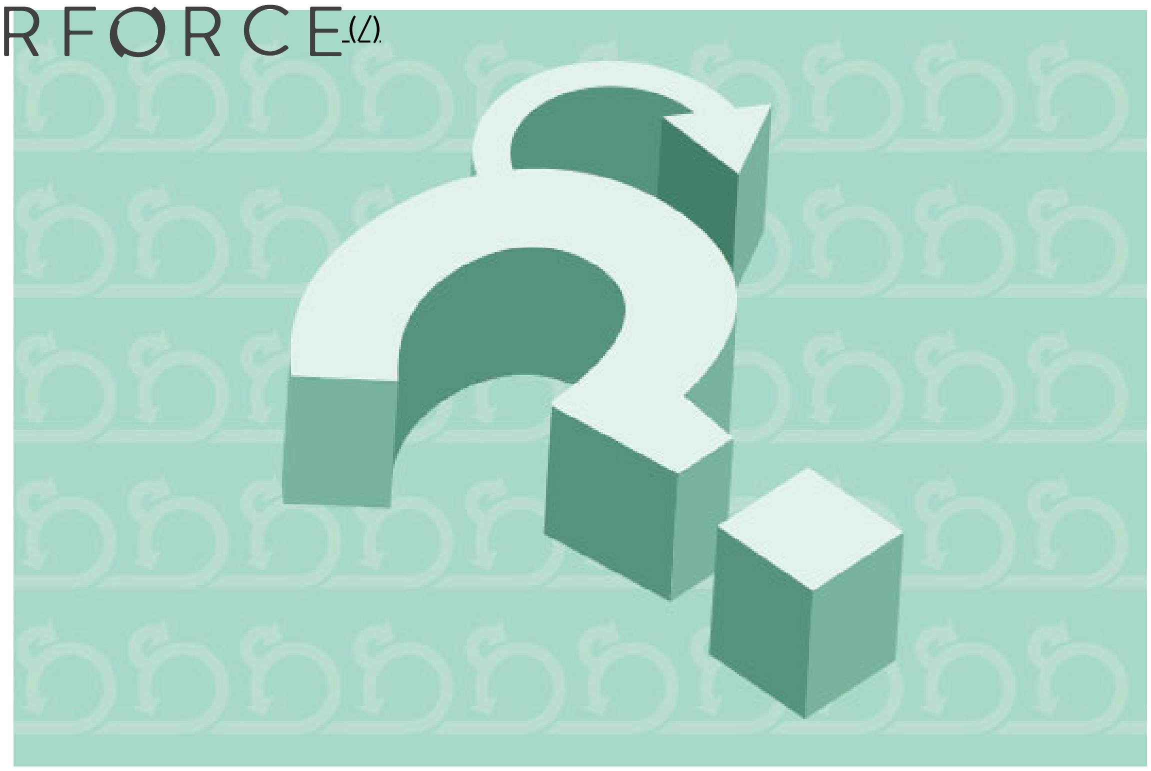

March 19, 2023

Agile Testing Methodology: 5 Examples for the Agile Tester

AGILE | TEST MANAGEMENT

By Gerhard Krüger

SEND FEEDBACK

Agile testing (https://www.blazemeter.com/blog/what-is-agile-testing) is software testing that follows the best practices of the Agile development framework.

Agile development takes an incremental approach to development. Similarly, Agile testing includes an incremental approach to testing. In this type of software testing (https://www.perforce.com/resources/alm/types-of-software-testing), features are tested as they are developed.

While the Agile methodology is fairly common at this point, many organizations – particularly organizations in regulated industries that require more formal documentation and traceability – still use waterfall or hybrid development frameworks. And some teams are in the process of transitioning to Agile.

Regardless of where you are in your Agile journey, this article will help you better understand the basics of Agile testing and how to easily transition to an Agile testing methodology. And we'll show you the best way to create and implement an Agile testing strategy.

SEE AGILE TESTING IN ACTION (https://www.perforce.com/resources/alm/helix-alm-demonstration)

Read along or skip to the section that interests you most:

SEND FEEDBACK

Is your organization in a highly regulated industry? Learn how to transition to Agile without losing traceability. Download the white paper: Transitioning to Agile in a Safety-Critical Environment >> (https://www.perforce.com/resources/alm/transitioning-agile-safety-critical-environment)

What is an Agile Testing Methodology?

Just as the Agile development methodology breaks down larger elements of feature development into smaller, more manageable pieces (such as user stories, tasks, and technical requirements), an Agile testing methodology breaks down larger testing elements into smaller, more specific tests with a narrower focus on a specific user story, task, technical requirement, etc.

By breaking testing down into these "bite-size" blocks, Agile testing allows testing to happen in a more timely and efficient manner.

Want a deep dive on Agile testing? Watch our on-demand webinar on Test Management in an Agile Reality >> (https://www.perforce.com/webinars/alm/test-management-agile-reality)

Agile Testing Supports Continuous Testing

The Agile testing methodology supports DevOps (https://www.perforce.com/solutions/devops) and continuous testing (https://www.perfecto.io/solutions/continuous-testing-for-devops). And continuous testing is important to improving product quality.

In Agile development, testing needs to happen early (https://www.perforce.com/blog/alm/what-shift-left-testing) and often. So, instead of waiting for development to be finished before testing begins, testing happens continuously as features are added. This is also referred to as "shift left" testing (https://www.perforce.com/blog/alm/what-shift-left-testing).

SEND FEEDBACK

Tests are prioritized just like user stories. Testers aim to get through as many tests as they can in an iteration. Adding automated testing (https://www.perforce.com/blog/alm/best-practices-automated-testing-webinar-recap) tools can help testers get through more of the testing backlog.

How does Agile testing and continuous testing help improve product quality? For starters, by testing earlier in the development process, issues and bugs can be discovered sooner. So you don't end up with costly last-minute surprises that can delay your time to market. And it's easier to fix bugs when they're still fresh in the developers' minds.

Continuous testing can also improve test coverage, and better test coverage results in better product quality and product safety.

>> *Learn how to transition to Agile testing.* *Get the white paper >>* (https://www.perforce.com/resources/alm/agile-development-methodologies-testers)

What Does an Agile Tester Do?

Testing and QA is everyone's responsibility in Agile. So testers and developers working in an Agile environment need to work closely together. Communication and collaboration are key.

Agile development is often driven by tests. Developers use Agile testing methods like test-driven development (TDD) to write the test first. Then they write the code that will be verified by the test. And developers and Agile testers should collaborate before user stories (e.g., requirements) are set.

Once development and testing are underway, close communication and collaboration remain important. Agile testers should be testing as developers write code. Plus, developers will probably do some testing. And Agile testers will probably do some coding.

Understanding the Definition of Done

In Agile development, the definition of done is a shared, standardized understanding among the team that a particular user story has been completed. The acceptance criteria in a user story are what will help drive the definition of done. If the user story passes the acceptance criteria, it can be considered done. This includes testing or validating the acceptance criteria. So, a test verifies that you've completed the user story.

SEND FEEDBACK

It's important that both Agile testers and developers know what has been tested and what defects still need to be resolved. Jump to the "How to Manage Agile Testing" section below to learn how to ensure visibility and alignment around testing and the definition of done.

Agile Testing Methodology: 5 Examples

Tests come first in Agile development. When you create a user story, you need to define the acceptance criteria. This drives testing and validation of the user stories.

It doesn't matter which Agile test methodology you use — Scrum, XP, Kanban, or even hybrid Agile. The following testing methods are typically used in Agile testing.

Test-Driven Development (TDD) (https://www.perforce.com/blog/alm/what-test-driven-development)

Test-driven development (TDD) starts with tests. This type of development begins by discussing what you want to test and then creating a user story. So, you start by writing a unit test. Then you write the user story. Finally, you write the code until the unit test passes.

Test-driven development is typically used on unit and component tests — which can be done with automated testing tools. TDD makes sure the features are working as they should be.

Acceptance Test-Driven Development (ATDD)

Acceptance test-driven development (ATDD) is similar to test-driven development. But acceptance test-driven development starts with customer input on functionality. This type of development begins by discussing how the product will be used. So, you write a user acceptance test (UAT). And then you write the code until it passes the test.

Acceptance test-driven development is typically used for acceptance tests. It verifies that the product functions as users would expect.

Behavior-Driven Development (BDD)

Behavior-driven development (BDD) often stems from test-driven development and acceptance test-driven development. In behavior-driven development, the purpose of development needs to be tied to a business outcome. So you'll have a user story — but the user story needs to answer why, in business terms, this feature is being developed. And in BDD, tests are included in user stories as scenarios or specifications.

SEND FEEDBACK

Behavior-driven development is also used for acceptance tests. It verifies that the product functions are necessary for the desired business outcome.

Exploratory Testing (https://www.perforce.com/blog/alm/what-is-exploratory-testing)

Exploratory testing is a style of testing that lets testers follow their intuition, rather than a predefined path. It's typically manual. Testers record what they're doing and save it as a test. They then figure out what exactly it is that they're testing as they go.

Exploratory testing is typically used to find hidden risks within a product. These would be bugs that are missed in functional tests done in test-driven development.

Session-Based Testing

Session-based testing has some similarities to exploratory testing. But there's a little more structure in session-based testing. Instead of testers figuring out what they're testing as you go, they start with a mission in mind.

Session-based testing is also used to find hidden bugs within a project.

Learn how to reduce risk with exploratory testing. Get the white paper >> (https://www.perforce.com/resources/alm/reducing-risk-exploratory-testing)

Agile Testing Strategy

Testing in traditional development typically includes a test plan (https://www.perforce.com/blog/alm/test-plans-basics-best-practices). But a well-documented test plan is not common in Agile development. Instead, Agile testers need to be flexible and ready to respond to shifts in requirements.

So there needs to be an Agile test strategy, rather than an Agile test plan.

There are many ways to outline your Agile test strategy. You may simply outline the strategy in a document. You might create a test matrix. Or, you might use a Kanban board.

No matter which Agile testing methodology you use, your strategy should include:

- Purpose (defined by the user story).
- Objectives (test cases).
- Scope (what needs to be tested).

- Methods (how tests will be run).

Creating an Agile test strategy can be tricky. But it's easy when you're using the right testing tools, such as Helix ALM (https://www.perforce.com/products/helix-alm).

Here's how you create an Agile test strategy in Helix ALM:

- Start with a purpose.
- Decide what you want to test.
- Write a user story, including acceptance criteria (your definition of done).
- Create a test case from that user story. This happens automatically in Helix ALM (https://www.perforce.com/resources/alm/helix-alm-demonstration).
- Generate a test run to validate the user story.

Want to see Helix ALM in action? Watch this free 20-minute demo >> (https://www.perforce.com/resources/alm/helix-alm-demonstration)

Benefits of Agile Testing

Agile testing improves product quality and enables development teams to release software on shorter cycles. And effective test case management (https://www.perforce.com/products/helix-test-case-management) helps Agile testers.

You'll find and fix errors faster. So you'll lower the risk of finding a bug at the very end of testing — and missing a release deadline.

You'll make customers happy by delivering reliable products and regular, stable releases. And you'll be able to better manage the scope of each release. That helps you prioritize features for each iteration and deliver the most important ones first. All of which will improve customer satisfaction and retention rates.

How to Manage Agile Testing

As mentioned above, the definition of done is a key element of Agile testing. It's critical for everyone on your team - Agile testers, developers, project managers and other stakeholders - know the current status of what is complete and what issues or defects still need to be resolved.

SEND FEEDBACK