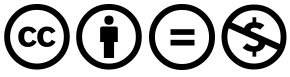
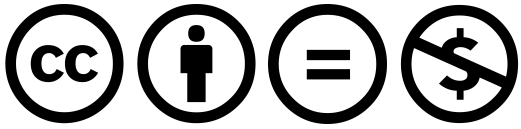
Agile Testing Learning Outcomes



ICAgile®
International Consortium for Agile

LICENSING INFORMATION

The work in this document was facilitated by the International Consortium for Agile (ICAgile) and done by the contribution of various Agile Experts and Practitioners. These Learning Outcomes are intended to help the growing Agile community worldwide.



SPECIAL THANKS

ICAgile would like to thank the contributors to the
Agile Testing Learning Outcomes:
Janet Gregory • Jeff Payne • Sharon Robson

CONTENTS

HOW TO READ THIS DOCUMENT

This document outlines the Learning Outcomes that must be addressed by accredited training organizations intending to offer ICAgile's Agile Testing Certification.

Each LO follows a particular pattern, described below.

0.0.0. Learning Outcome Name

Additional Context, describing why this Learning Outcome is important or what it is intended to impart.

The Learning Outcome purpose, further describing what is expected to be imparted on the learner (e.g. a key point, framework, model, approach, technique, or skill).

LEARNING OUTCOMES

1. AGILE TESTING MINDSET

1.1. OVERVIEW OF AGILE TESTING

1.1.1. Origins of Agile Testing

Many people who hear about Agile testing for the first time assume that it was created as part of the Agile movement. In actuality, much like Agile itself, many of the Agile testing techniques where espoused well before the Agile Manifesto was created.

Explain and anchor the ideas of Agile testing in earlier work, giving the learners continuity from the past to the present.

1.1.2. Agile Testing vs. Traditional Approaches

Agile testing is much different from testing performed during traditional software development approaches.

Compare and contrast the major differences between Agile testing and testing performed as part of traditional (phased-based) software development approaches in which testing is primarily performed by software testers who are often in their own organization and sometimes only involved late in the development life-cycle.

1.2. MINDSET & CULTURE

1.2.1. Agile Testing Principles

The Principles behind the Agile Manifesto establish guiding principles for not only the Agile movement but Agile testing as a discipline.

Explain how the Agile Manifesto is realized within an Agile testing process and approach.

1.2.2. Whole Team Approach

Quality is not "owned" by a particular role in Agile. It is a property of software that the entire team must make sure is present before software is released to customers.

Explain that quality is everyone's responsibility during Agile projects and everyone is involved in software testing. Testers are often ideally suited to guide the team toward achieving its quality goals and its definition of "done" based on team definitions of the quality attributes for the product. The role each team member typically plays for quality will be discussed.

1.2.3. Building Quality In

The role of a tester shifts in Agile from that of quality gate keeper to a facilitator who supports the team through asking questions to help clarify understanding, as well as testing and critiquing the product.

Demonstrate the mindset shift of testers from that of an independent group responsible for gating the development process to a collaborative team member focused on improving the product and releasing value to the customer.

1.2.4. Continuous Improvement and Feedback

Agile testing provides critical insights and feedback into the software process that can be used to drive team and quality improvements and assist the organization in making informed business decisions regarding software release.

Explain that Agile testing is a critical feedback component when seeking to improve operational effectiveness.

1.2.5. Ingraining the Agile Mindset (Hands-on Exercise)

Practicing behaviors can solidify the mindset and culture of Agile testing.

Practice situations in which the Agile testing mindset is likely to be different, so the learner can internalize the difference experientially, not just in concept.

2. TESTING TECHNIQUES

2.1. CATEGORIES OF TESTING

2.1.1. Agile Testing Quadrants or Categories

Testing activities can be broken into various categories of testing based upon their purpose and value. Types of testing are often broken into categories that include: testing that supports project team development efforts, testing that looks at quality from a business perspective, testing that critiques the product and testing that exercises the relationship between software and its deployment platform.

Explain and categorize the purpose of various testing techniques so they can be applied appropriately and at the right time within an Agile environment.

2.1.2. Automation Pyramid - Introduction

Automated testing can be performed at various levels within a software application. An automation pyramid or structure describes these various levels and discusses the approach and likelihood of automating tests within each of them.

Explain the various types of testing that can be automated and how decisions get made regarding what to automate during an Agile project.

2.1.3. Testing Techniques

Test design techniques are necessary in an Agile environment, they need to incorporate existing techniques and extend and apply them to the collaborative methods.

Apply some simple ideas about approaching test design; For example mind-mapping, context diagrams and other methods conducive to a collaborative environment.

2.2. COLLABORATING WITH DEVELOPERS

2.2.1. Unit and Component Testing

Developer testing of individual software units and associated components is critical to detecting implementation defects within software. Unit and component tests are leveraged within TDD as well.

Demonstrate the purpose and approach to successfully implementing unit and component testing on Agile projects and show how testers support developer testing during development cycles.

2.2.2. Pairing between the Developer and Tester

Additional testing techniques beyond unit/component testing are often necessary to test beneath the UI level.

Explain how developers and testers pair during fixture / test method development and API-level testing.

2.3. EXAMPLE DRIVEN DEVELOPMENT

2.3.1. Acceptance Test-Driven Development (ATDD)

ATDD is a common technique for assuring that Stories are implemented in a manner that satisfies the exit criteria defined for Story completion. It is often used as a technique to test Stories but in actuality includes the testing of key business processes and non-functional requirements as well.

Explain the purpose and approach to successfully implementing Acceptance Test-driven Development (ATDD) on Agile projects.

2.3.2. Behavior-Driven Development (BDD)

BDD is an alternative approach to ATDD that is sometimes used to test Stories, Business Process and non-functional Requirements based upon an understanding of user behavior.

Explain purpose and approach to successfully implementing Behavior-driven Development on Agile projects.

2.3.3. Spec by Example

Specification by example uses the 3 amigos idea in a workshop environment to express examples.

Explain the terminology and how it is different / same as ATDD and BDD.

2.4. FEATURE AND STORY TESTING

2.4.1. User Story Testing

Testing of User Stories is critical to successful development of software within an Agile project. This testing is often performed using the techniques above but can be done in other ways as is appropriate or necessary.

Demonstrate User Stories are tested during software development; this is an extension to ATDD to include boundary conditions and other types of testing such as exploratory testing.

2.4.2. Feature Testing

While the above techniques are the most common, there are a variety of other testing techniques that can be applied to test software features.

Explain and categorize the bigger picture beyond a user story (work-flows, end-to-end scenarios).

2.4.3. Exploratory Testing

Exploratory testing provides a mechanism for additional testing to be performed on Stories or Business Processes based upon a tester's intuition and knowledge about the product.

Describe and apply Exploratory Testing techniques and approaches and explain how they are best applied to an Agile project. Introduce the idea that automation can support ET.

2.4.4. Non-Functional Testing

Non-functional, or Quadrant 4 tests are sometimes ignored by customers. There is no clear understanding of who is responsible for it.

Explain how to include testing these quality (non-functional) attributes in our testing.

3. AGILE TESTING PROCESS

3.1. ROLES AND RESPONSIBILITIES

3.1.1. Team-Based Testing Approach

Testing during an Agile project is team-oriented wherein it is common for every member of the team to provide some level of testing support.

Explain that within an Agile project, the entire project team is responsible for test plans, test design, test cases, test automation and test reporting.

3.1.2. Typical Business Representative Role in Testing

Business Representatives typically provide guidance on acceptability and provide examples of what Stories are intended to accomplish.

Explain the common test activities that a business representative is involved with during an Agile project.

3.1.3. Typical Programmer Role in Testing

Software programmers typically build, automate and run a variety of tests at a variety of levels as part of their development process. TDD and ATDD leverages this testing to improve design and development.

Explain the role software programmers play in Agile testing.

3.1.4. Typical Tester Role in Testing

Software testers typically work hand-in-hand with the product owner and programmers to plan, execute and report on the testing that is performed at all levels.

Explain and classify the role software testers play in Agile testing. Skills should include technical awareness; T-shaped skill-set.

3.1.5. Role of Test Managers in Agile

When an organization has decided to organize around products or projects, test managers are often left to wonder what their role within this new structure will be.

Explain the new role that a test manager plays and the reason behind the shift. For example, mentoring test communities of practice and helping coordinate post-development testing activities.

3.2. TEST STRATEGY AND PLANNING

3.2.1. Different Strategies Based on Levels of Precision

Lightweight planning is typically part of the release planning done prior to associated iterations.

Demonstrate how lightweight test strategy and planning is performed during release planning and how decisions are made regarding what type of test documentation is needed and how much is enough.

3.2.2. During Iteration Planning/Kickoff

Test planning at iteration kickoff focuses on detailing acceptance criteria and examples for Stories.

Demonstrate how tests are developed prior to implementation.

3.2.3. Lightweight Test Plan Documentation

Test planning in Agile is different from traditional development approaches as the goal is to provide the least amount of documentation needed to get the job done.

Explain how to determine the amount of test documentation necessary for a given environment or situation.

3.2.4. Defect Tracking and Management

The amount of defect tracking that is performed during an Agile project depends upon what works best for the team.

Explain the key trade-offs for determining which defects to track and which to rely upon team communication to correct without tracking.

3.2.5. Results Reporting

Test reporting during Agile projects depends upon what works best for the team.

Explain the key trade-offs between documented test results and team communication of those results.

3.2.6. Test Metrics

Metrics collected to support test completeness and release readiness decisions. Example: determining whether a story, feature or/ iteration is "Done".

Explain which metrics make sense to collect and report on for both test completeness and release readiness within an Agile project.

3.2.7. Regression Tests

Automated regression tests are essential to reducing the cost of change and providing real-time feedback during the development process.

Show how to best leverage tests that have been automated during development within future iterations and releases.

3.3. SUCCESSFUL DELIVERY

3.3.1. Time-Boxed Delivery

Time-boxed delivery means that at the end of every iteration, there is a potentially shippable product, including testing.

Explain that all work on a story needs to be completed within the iteration and time must be allocated by the team to allow that to happen. Testing and coding should be planned for concurrency not sequential work. Testing and coding are to be considered one process not two steps.

3.3.2. Continuous Delivery

Continuous delivery means that every build has the potential to be deployed to production.

Explain what this means to testing. Inclusion of automation, responsiveness to defects and change. Explain how aspects of testing that enable continuous delivery in a team are captured and incorporated into the work flow.

3.3.3. Post-Development Test Cycles

In Agile, testing is as early as possible for fast feedback, but during the endgame, there is often final testing that needs to happen.

Describe the interaction of testers in post-development test cycles. This includes integration test teams that take advantage of economies of scale for security testing, browser compatibility, interoperability, etc. in an integrated environment. It also includes testing in the End Game for final UAT, or final performance testing.

3.3.4. Iteration Wrap-Up

Wrap-up activities during an iteration include a product demo, retrospective and sometimes a User Acceptance Test.

Explain of the role that software testers play during iteration wrap-up activities.

3.3.5. Definition of a Release/End Game

A release process (aka “end game”) is performed whenever a decision has been made by the business to release software to customer(s).

Explain how a release decision is made and what testing activities are typically part of the release process.

3.3.6. User Acceptance Testing (UAT)

User Acceptance Testing is used within Agile to gain customer feedback on a working piece of software before its release.

Demonstrate user acceptance testing (UAT) techniques and approaches and how they are best applied to an Agile project.

3.3.7. System-Wide and Cross-team Testing

The focus needs to be on products and solutions not on projects. Therefore testing needs to maintain the big picture that spans beyond the individual project and crosses over to multiple teams or projects to a more system-wide view of testing

Explain how to coordinate testing across Agile projects and associated testing activities when the project must interact/interface with other projects and IT technology release schedules.

3.3.8. Post-Release Testing

Testing after software release typically consists of testing "hot fixes" for critical defects identified in the field and ongoing testing of bug fixes not fixed prior to release.

Explain the types of testing that is performed post release and how continuous testing supports a continuous delivery process.

3.3.9. Documentation for Regulatory Requirements

Demonstrate how regulatory requirements can still be fulfilled when following an Agile development process.

Prepare a set of tests based on regulatory requirements and show how they can be incorporated into the testing cycles.

3.4. TEST ENVIRONMENTS AND INFRASTRUCTURE

3.4.1. Typical Environments for Test

Describe typical test environments that must be set up and maintained to support testing activities during iterations and releases. This needs to include information about the tool and data setups as well.

Discuss the characteristics and typical definitions test environments. Justify why each type of testing needs a defined environment to match the purpose of the tests being run.

3.4.2. Build Pipeline

Explain how the build pipeline works to help determine which testing is appropriate for which test environment. What needs to happen before the build is deployed to the next stage?

Discuss the product maturity as it moves through the levels of test automation that can be applied.

3.4.3. Automated Builds

Explain how automated builds are set up to support a continuous testing process.

Explain why automated build processes are valuable and how they support continuous testing.

3.4.4. Testing the Proper Build

Discuss the best practices associated with choosing a build for test and keeping development and testing in sync during the process.

Describe how to choose a build for testing and explain the factors influencing the decision.

3.4.5. Test Data Management

Discuss traceability and techniques for building the relationship between tests, data, environments and features.

Explain the effective ways to generate and manage test data during an Agile process.

3.5. WORKING ON DISTRIBUTED TEAMS

3.5.1. Distributed Team Communication

Discuss common problems with distributed teams. Provide examples of how tests can become the common language of the team, bridging ambiguity and confirming understanding no matter location or time zone.

Explain how communication can be most effective on distributed teams.

3.5.2. Distributed Team Coordination

Discuss coordination, liaison overheads, hand over methods for the engagement of the testing activities within the iteration.

Explain ways that testing activities can be coordinated when the team is distributed.