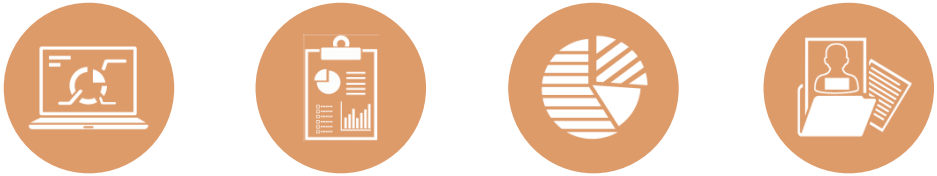
PMI®—Agile Certified Practitioner (PMI-ACP)®

Agile Quality



Objectives

After completing this lesson, you will be able to:

- Explain Agile embeds quality throughout the project lifecycle
- Describe Agile Test-First Development and its processes
- Explain Agile Acceptance Test-Driven Development and its stages
- Define the criteria for Done in Agile
- Identify the best practices of Continuous Integration

Quality is defined as “conformance to requirements and fitness of use”. Quality can be broadly classified into the following two categories:

Customer quality

Customer quality is extrinsic or external and delivers value in the short term.

Technical quality

Technical quality is intrinsic or internal and enables continuous delivery of value over time.

Poor quality work results in unreliable products, but more critically, it results in products that are far less responsive to future customer needs.

Agile Frequent Verification and Validation

The objective of an iteration in Agile must be to produce code that is of 'near releasable' or 'potentially shippable' quality. This requires the code to have passed through the verification and validation steps.

Verification is looking for defects in the product to analyze how it differs from intended functionality. Product inspection and peer review can be done to test the product against the stated requirements. Unit or statistical or system tests will be the final measure to help remove defects.

Validation is repeatedly checking with the stakeholders if the product meets their requirements. Selected users can perform acceptance testing of ready components or UI prototypes that wire together before writing the code pieces.

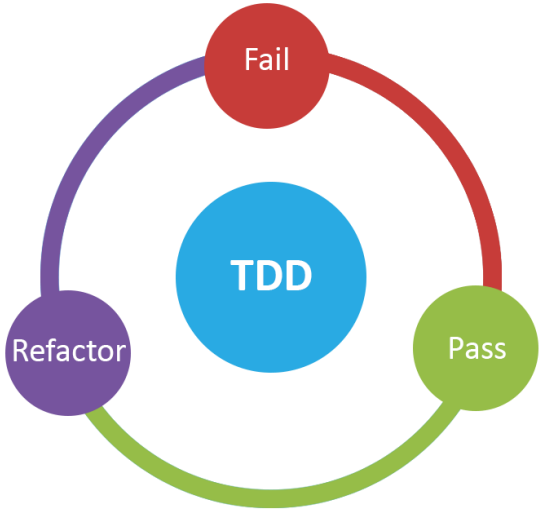
Agile Frequent Verification and Validation (contd.)

An iteration in agile development will produce artifacts (typically code, but could be documents) that require verification and validation.

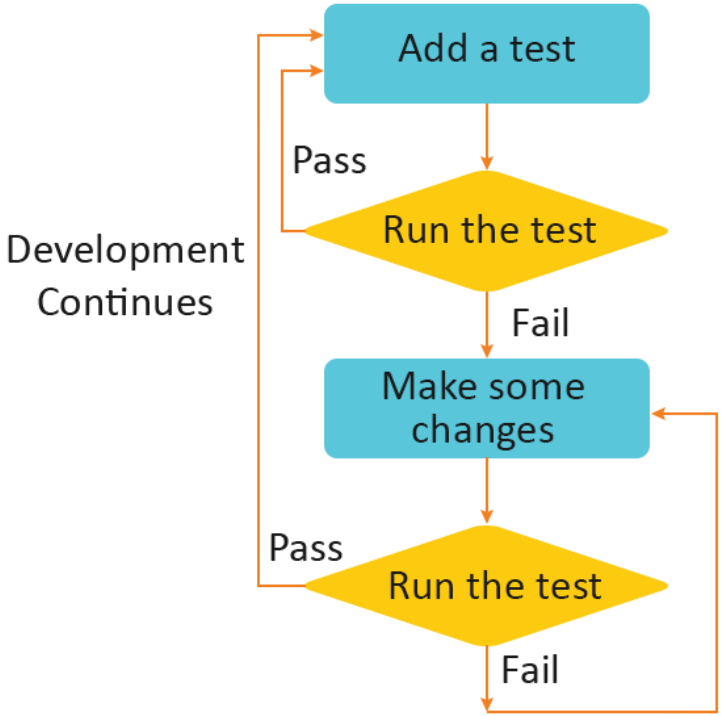
- For requirements and design, the verification and validation is the result of peer reviews with team members and with the customer.
- For coding, verification and validation is done by code reviews, unit testing, and functional and non-functional testing.

Test-First Development (TFD) is also known as Test-Driven Development (TDD) or test-first programming.

- It was first introduced with Extreme Programming (XP).
- It is an evolutionary (iterative and incremental) approach to programming where Agile software developers must first write a test before they write code to make the test pass.
- This develops a rich collection of tests for all code developed.
- These tests are executed by the Integration Server every time a code is compiled.
- The more tests available, the more 'courage' testers (developers) have to refactor.



The steps involved in TFD process are as follows:



The advantages of TDD are as follows:

- TDD ensures developers to do detailed design just in time (JIT) before writing the code.
- It ensures that agile developers have testing code available to validate their work, ensuring that they test as often and early as possible.
- It allows developers to refactor their code to keep it in the highest quality possible, as the test suite will detect if they have 'broken' anything as the result of refactoring.
- TDD substantially reduces the incidence of defects [Janzen & Saiedian].
- It improves design, documents public interfaces, and guards against future mistakes.

Acceptance Test Driven Development (ATDD) focuses on writing requirements in a testable form.

- Unit tests in TDD focus on building the code right; Acceptance Tests focus on building the right code.
- Each requirement is expressed in terms of inputs and expected outputs.
- ATDD requirements are documented in Wiki pages that can be executed once the code is complete.
- ATDD involves creating tests before code, and those tests represent expectations of behavior the software should have.
- Acceptance tests are ‘black box’ tests. They test the behavior of functionality of the system.
- Unit tests are ‘white box’ test. They test the internals of the system.

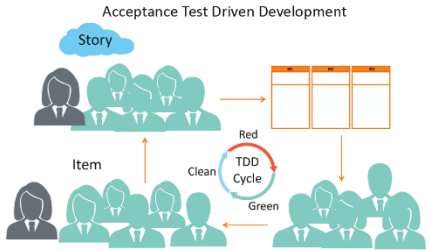
The ATDD Cycle comprises four stages, discuss, distill, develop, and demo.

Discuss

- A discussion is held with the business stakeholder, requesting the features.
- An understanding of the stakeholder expectations is established.
- Acceptance tests in collaboration with the stakeholder can then be created.

Distill

- The tests are captured in a format that works with the test automation framework.
- A variety of test automation frameworks that support defining the tests before the implementation: FIT, Fitnesse, Concordian, and Robot Framework.



Demo

- Acceptance tests for all the completed stories are executed to demonstrate that they have passed.
- Manual exploratory testing to reveal gaps in the acceptance criteria and to discover defects that might occur when multiple user stories are executed in a scenario.

Develop

While implementing the code, developers will follow the Test Driven Development techniques of writing tests,

- watching them fail,
- write codes,
- watch the tests pass, refactor, etc.

IBM team has sustained the use of TDD for five years and over ten releases of a Java-implemented product. A set of experiments was conducted with professional programmers to validate the effectiveness of TDD.

Experiment 1: Develop a short program in Java to automate the scoring of a bowling game.

The results of the experiment conducted are given below:

Team using TDD Technique

- Developers took 16% more time to write the tests and then develop the code
- Passed 18% more functional black box test cases
- Developers came up with good quality automated test cases

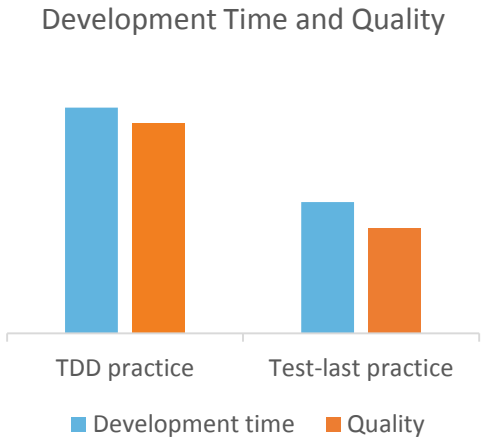
Team not using TDD Technique

- Developers took relatively less time to develop the code
- Passed relatively fewer black box test cases
- Developers did not write any worthwhile automated test cases

Experiment 2: To determine the effectiveness of TDD, each practitioner was asked to complete a programming task using both TDD practice and test-last practice, each taking approximately 5 hours.

Result: TDD requires more development time. But the improved quality could offset this initial increase in development time. The test-last practice resulted in poor quality, though the initial time taken is less.

Conclusion: Agile focusses on value delivery rather than tasks completed. The TDD technique leads developers to design precise and accurate test cases.



Definition of Done

Every project will have its own definition of Done. It's important for the team to agree to the definition of Done and commit to it. Often, the definition of Done is an Information Radiator that is posted in the team room.

Definition of 'Done' should include the following criteria:

- It has been tested and has passed user acceptance or client approval.
- It has passed an in-house iteration review.
- It is of a 'shippable' or 'deliverable' quality.

Checklist for Story Completion

Following is a sample checklist for story completion criteria. Teams must come up with their own checklists, before marking a story 'complete'.

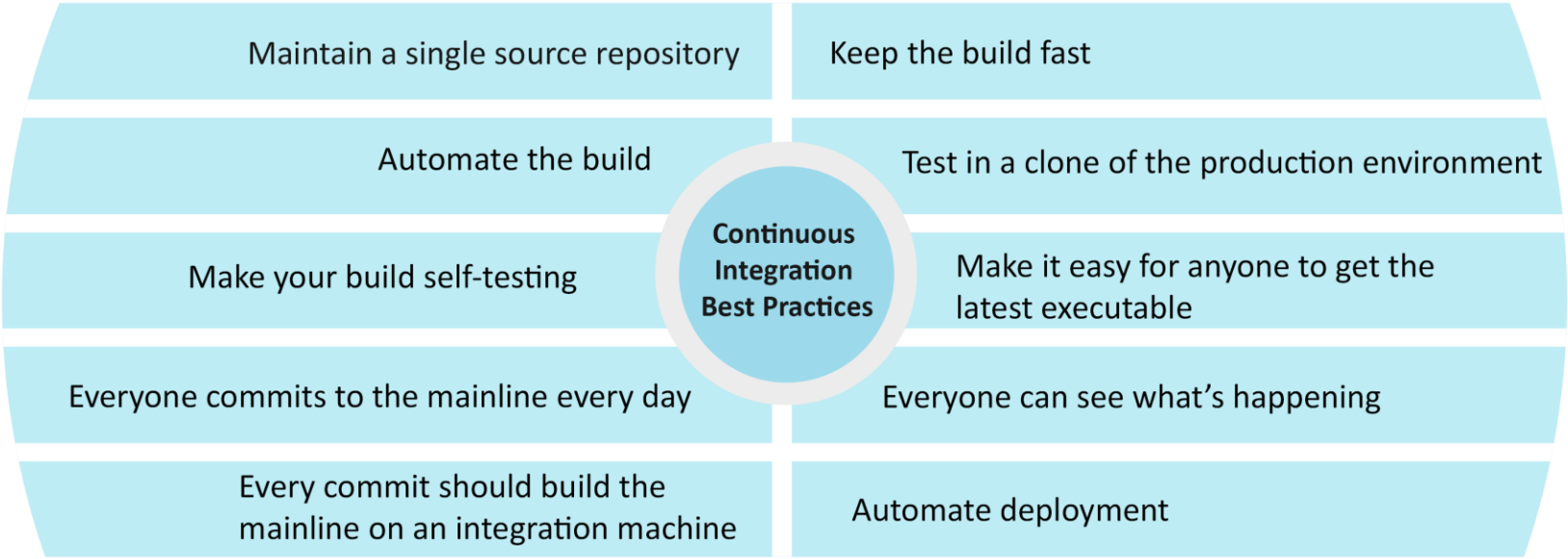
- Story is **tested**.
- Functional **code** is written.
- **Design** is complete and refactoring is done.
- Story is **integrated** across different components.
- Story is integrated into the build system and is a part of the **build**.
- Story is available through the **installer**.
- Story can be **migrated** if required.
- Story is **reviewed** by stakeholders and confirmed that it meets their requirements.
- All the identified bugs have been **fixed**.
- Customers have **accepted** that the story is complete.

Continuous Integration is one of the twelve practices of Extreme programming.

- It is an approach that keeps all the code integrated and builds release infrastructure along with the rest of the application.
- It can often be used as a visual information radiator by attaching lava lamps or some other signaling device to the integration server. The red lava indicates the build is broken and the green lava lamp means the build is successful.

Best Practices of Continuous Integration

Best practices of Continuous Integration are as follows:



QUIZ 1	Which of the following is not true about Test-First Development?

a. It is also known as Test-Driven Development (TDD).

b. Is an evolutionary approach to programming where agile software developers must first write a test that fails before they write new functional code.

c. It involves updating the functional code if the new test is passed.

d. It passes user acceptance or client approval.

QUIZ 1	Which of the following is not true about Test-First Development?

a. It is also known as Test-Driven Development (TDD).

b. Is an evolutionary approach to programming where agile software developers must first write a test that fails before they write new functional code.

c. It involves updating the functional code if the new test is passed.

d. It passes user acceptance or client approval.

Answer: d.

Explanation: It passes user acceptance or client approval is the definition of 'Done' and not Test-Driven Development.

QUIZ 2	Which of the following correctly states the components and sequence of the ATDD cycle?

a. Discuss, Distill, Develop, Demo

b. Distill, Develop, Discuss, Demo

c. Discuss, Develop, Demo, Acceptance

d. Develop, Distill, Demo, Acceptance

QUIZ 2	Which of the following correctly states the components and sequence of the ATDD cycle?

a. Discuss, Distill, Develop, Demo

b. Distill, Develop, Discuss, Demo

c. Discuss, Develop, Demo, Acceptance

d. Develop, Distill, Demo, Acceptance

Answer: a.

Explanation: Discuss, Distill, Develop, and Demo are the four stages in the ATDD cycle.

QUIZ 3	What does customer quality deliver?

a. Product

b. Video

c. Value

d. Demo

QUIZ 3	What does customer quality deliver?

a. Product

b. Video

c. Value

d. Demo

Answer: c.

Explanation: Customer quality delivers value.

QUIZ 4	What of the following does not mean done?

a. When code is developed.

b. It has been tested and has not passed user acceptance or client approval.

c. It has been tested and has passed user acceptance or client approval.

d. It is “shippable” or “deliverable”.

QUIZ 4	What of the following does not mean done?

a. When code is developed.

b. It has been tested and has not passed user acceptance or client approval.

c. It has been tested and has passed user acceptance or client approval.

d. It is “shippable” or “deliverable”.

Answer: a.

Explanation: Only the code being completed does not signify the completion or done.

QUIZ 5	How does TDD help developers?

a. It prevents them from avoiding the difficult task of writing tests.

b. It ensures that their code is always defect free.

c. It reduces the questions from management on whether the code has been tested properly.

d. It reduces the re-work developers would have to do and gives them the courage to refactor.

QUIZ 5	How does TDD help developers?

a. It prevents them from avoiding the difficult task of writing tests.

b. It ensures that their code is always defect free.

c. It reduces the questions from management on whether the code has been tested properly.

d. It reduces the re-work developers would have to do and gives them the courage to refactor.

Answer: d.

Explanation: Refactoring is most effective with Test Driven Development.

QUIZ 6	Can Continuous Integration have an Information Radiator?

a. Yes, lava lamps can be used to show the status of the builds by wiring them to the integration server.

b. No, information radiators are for project progress indicators only.

c. Yes, but only if each team member agrees to update a whiteboard with status every time they commit code.

d. Yes, information from Continuous Integration can be radiated using osmosis.

QUIZ 6	Can Continuous Integration have an Information Radiator?

a. Yes, lava lamps can be used to show the status of the builds by wiring them to the integration server.

b. No, information radiators are for project progress indicators only.

c. Yes, but only if each team member agrees to update a whiteboard with status every time they commit code.

d. Yes, information from Continuous Integration can be radiated using osmosis.

Answer: a.

Explanation: Many development teams have hooked their integration server to lava lamps, with green lava lamps for success and red for failure.

Here is a quick recap of what was covered in this lesson:

- Customer quality delivers value in the short term while technical quality enables continuous delivery of value over time.
- The objective of an iteration in Agile must be to produce code that is of 'near releasable' or 'potentially shippable' quality. This requires the code to have passed through the verification and validation steps.
- TDD is an iterative and incremental approach to programming where Agile software developers must first write a test before they write code to make the test pass.
- ATDD focuses on writing requirements in a testable form.
- For a story to be done, it has to be tested, passed user acceptance test and an in-house iteration review, and should be of a 'shippable' or 'deliverable' quality.
- Continuous Integration is an approach that keeps all the code integrated and builds release infrastructure along with rest of the application.

THANK YOU