TESTING IN THE AGILE ENVIRONMENT

Agile testing is a software testing practice that follows the Agile software development methodology. In Agile development, projects tend to evolve during each sprint among collaborators and shareholders. Agile testing focuses on ensuring quality throughout the Agile software development process.

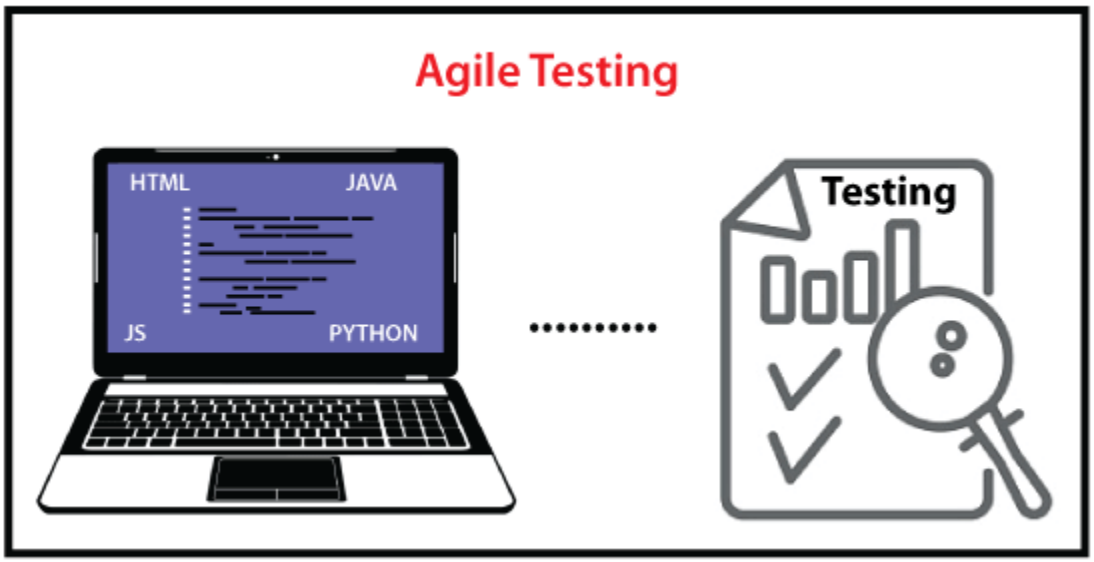
Agile test levels ensure thorough software testing at different stages: unit (individual components), integration (collaboration), system (entire system), and acceptance (end-user validation).

Agile testing follows the standards of agile software development.

Agile testing is an **iterative and incremental method,** and the necessities, which develop during the cooperation between the **customer** and **self-establish teams**.

In agile testing, the word "**Agile**" primarily signifies something that can be performed quickly and immediately, but also in the area of **software development**.

The core-functional agile team implements it in order to test the software product and its several modules. The implementation of agile testing makes sure to deliver a high quality product as bugs or defects get deleted in the initial stage of the project itself.



Unlike the Waterfall model, Agile Testing can create at the beginning of the project with endless incorporation between development and testing. It is not a sequential but the continuous process.

The agile testing process is a smart way of testing complicated software, which accepts more effective results as compared to the traditional testing procedures.

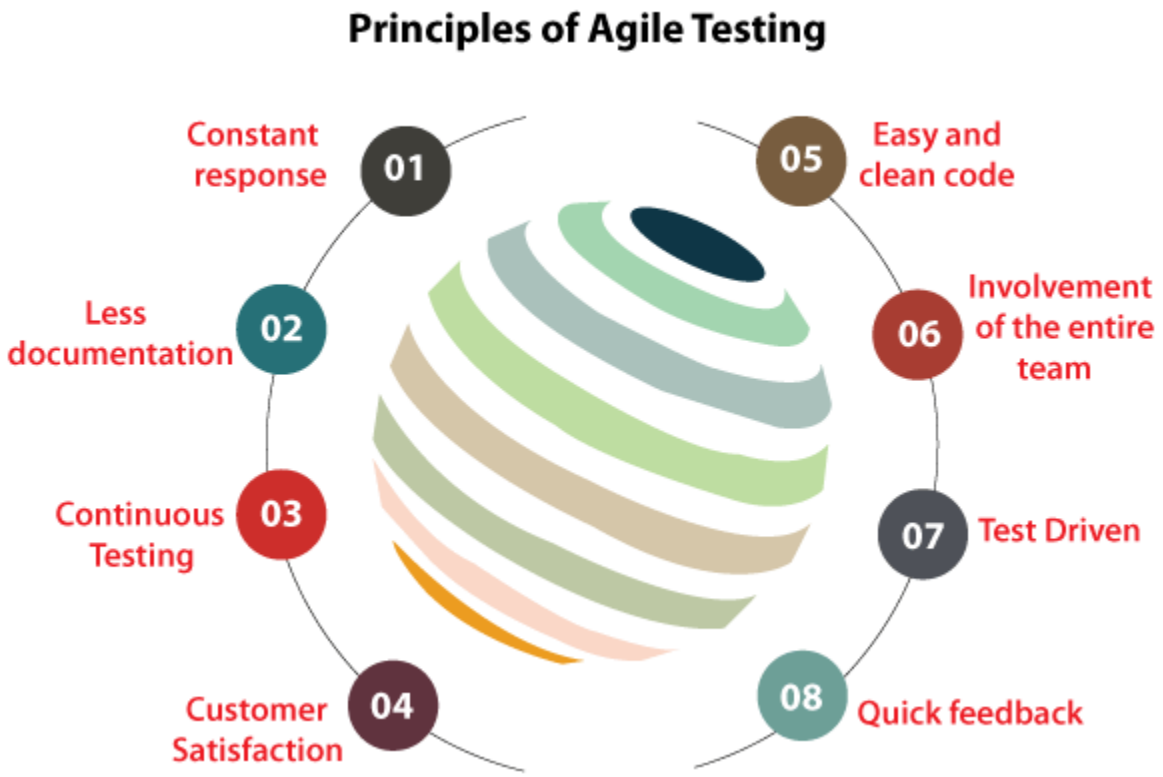
In the modern days of software testing, agile testing has gained a lot of acceptance and significance. The execution of agile testing will help us identify the initial error and elimination, giving better results in less development time and costs.

Principles of Agile Testing

Agile Testing includes various different principles that help us to increase the productivity of our software.

1. **Constant response**
2. **Less documentation**
3. **Continuous Testing**
4. **Customer Satisfaction**
5. **Easy and clean code**
6. **Involvement of the entire team**
7. **Test-Driven**
8. **Quick feedback**

For our better understanding, let's see them one by one in detail:



1. Constant Response

The implementation of Agile testing delivers a response or feedback on an ongoing basis. Therefore, our product can meet the business needs.

In other words, we can say that the Product and business requirements are understood throughout the constant response.

2. Less Documentation

The execution of agile testing requires less documentation as the Agile teams or all the test engineers use a reusable specification or a checklist. And the team emphases the test rather than the secondary information.

3. Continuous Testing

The agile test engineers execute the testing endlessly as this is the only technique to make sure that the constant improvement of the product.

4. Customer Satisfaction

In any project delivery, customer satisfaction is important as the customers are exposed to their product throughout the development process.

As the development phase progresses, the customer can easily modify and update requirements. And the tests can also be changed as per the updated requirements.

5. Easy and clean code

When the bugs or defects occurred by the agile team or the testing team are fixed in a similar iteration, which leads us to get the easy and clean code.

6. Involvement of the entire team

As we know that, the testing team is the only team who is responsible for a testing process in the Software Development Life Cycle. But on the other hand, in agile testing, the business analysts (BA) and the developers can also test the application or the software.

7. Test-Driven

While doing the agile testing, we need to execute the testing process during the implementation that helps us to decrease the development time. However, the testing is implemented after implementation or when the software is developed in the traditional process.

8. Quick response

In each iteration of agile testing, the business team is involved. Therefore, we can get continuous feedback that helps us to reduces the time of feedback response on development work.

How is Agile Methodology used in Testing?

Agile Testing is a fast and informal testing process. In simple terms, we can say that it is specified as an advanced and **dynamic type of Testing** that is performed regularly throughout every iteration of the **SDLC (Software Development Life Cycle)** by the agile test engineers.

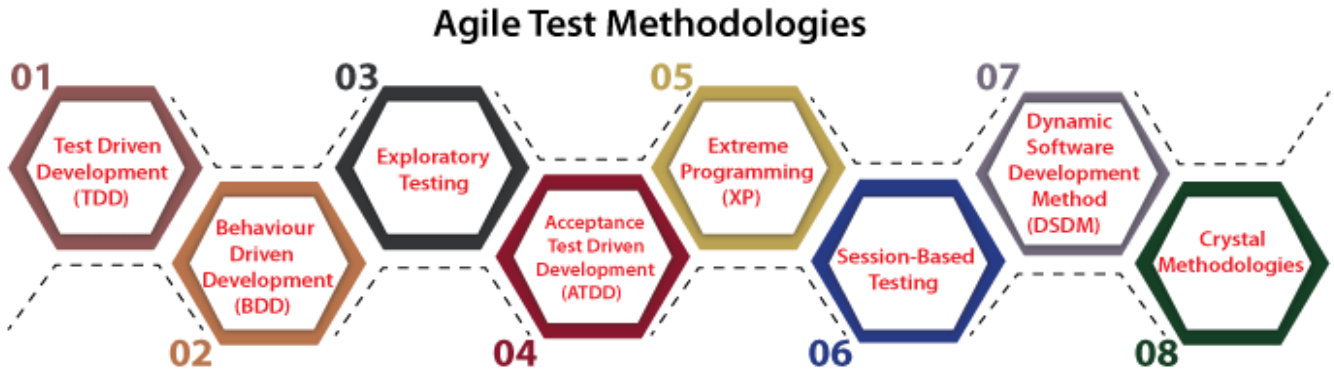
If we deliver the software quickly with the best of the attributes, and the customer satisfaction is the primary concern at some stage in the agile testing process.

Agile Testing Methodologies

When we are executing the agile testing, the team takes help from several agile methodologies, which support them in accomplishing the precise results.

Some of the effective agile testing methodologies are as follows:

- **Test-Driven Development (TDD)**
- **Behavior Driven Development (BDD)**
- **Exploratory Testing**
- **Acceptance Test-Driven Development (ATDD)**
- **Extreme Programming (XP)**
- **Session-Based Testing**
- **Dynamic Software Development Method (DSDM)**
- **Crystal Methodologies**



Test-Driven Development (TDD)

The test-driven development method begins with the test itself. As the name proposes, the TDD varies upon the repetition of the development cycle.

We already knew that the first step in of development cycle is to create **a unit test case.** And in the next step, we will be designing the code that fits the test case in order to execute the test cases.

Hence, the whole code is designed until the unit test passes. Generally, the test-driven development is executed by using the automated testing tools and implement on units and components of the code.

Behavior-Driven Development (BDD)

The following method in agile testing is **behavior-driven development**. The BDD enhances the communication between the project stakeholders to facilitate the members adequately and understood all the components before the development process begins.

It is constructed on the same rules as TDD and ATDD. Therefore, the code is developed as per the test case designed in this testing methodology too.

The primary purpose of this development is to emphasize the identification of business needs and outputs. And the development should be consistent to a business output.

In behavior-driven development, we need to follow the below steps:

1. Describe the behavior.
2. Generating the test case.
3. Writing code as per the test case is specified.
4. Continuing the process until the code passes the test case.

EXPLORATORY TESTING

In Software testing, exploratory testing is one particular type where the test engineers have the fundamental freedom to explore the code and create the most effective software.

In simple words, we can say that if we don't have the requirement, then we do one round of exploratory testing.

Exploratory testing is a very significant part of the agile test as it helps discover the unknown risks from the software that a simple testing approach could not have noticed.

To explore each aspect of the software functionality, the test engineer creates various test cases, executes different tests, and records the process to learn it and understand its particular flow.

While performing the exploratory testing, we need to follow the below steps:

- Exploring the application in all possible ways
- Understanding the flow of the application
- Preparing a test document
- Testing the application

Acceptance Test-Driven Development (ATDD)

Another methodology of agile testing is Acceptance Test-Driven Development (ATDD). The ATDD approach emphasizes the customer's requirements by involving the team members with different viewpoints.

The team's members of development, testing, and the customers come together in order to develop the acceptance test from the customer's perspective.

In Acceptance Test Driven Development, the code is acquired along with the developed acceptance test case.

It is a very customer-centered methodology; the primary objective of using the ATDD methodology is to develop a program based on the user's view.

Extreme Programming (XP)

The next significant agile methodology is **Extreme Programming** which is denoted as **XP.** When there is a continuous modification in the user requirements, we will go for the extreme programming methodology.

Just like other agile testing methodologies, **Extreme Programming** is also a **customer-centric** methodology.

The XP will help us deliver a quality product, which meets the customer's requirements that are made clear throughout the process of development and testing.

SESSION-BASED TESTING

In the row of various agile testing methodologies, the next methodology is Session-based testing. It is mainly is created on the values of exploratory testing.

Though session-based testing contains some structure and on the other hand, exploratory testing is performed unexpectedly without any planning. It is used to help us identify the hidden bugs and defects in the particular software.

The session-based testing structure is prepared by executing the tests in continuous sessions where test engineers must report the tests, which took place throughout the process.

Dynamic Software Development Method (DSDM)

Another effective method of agile testing is **Dynamic Software Development Method.** It is a **Rapid Application Development** (RAD) approach that offers a delivery framework to agile projects.

In other words, we can say that the **Dynamic Systems Development technique (DSDM)** is a correlate degree agile code development approach, which gives a framework for developing and maintaining systems.

The **Dynamic Software Development Method** can be used by users, development, and testing teams.

Crystal Methodologies

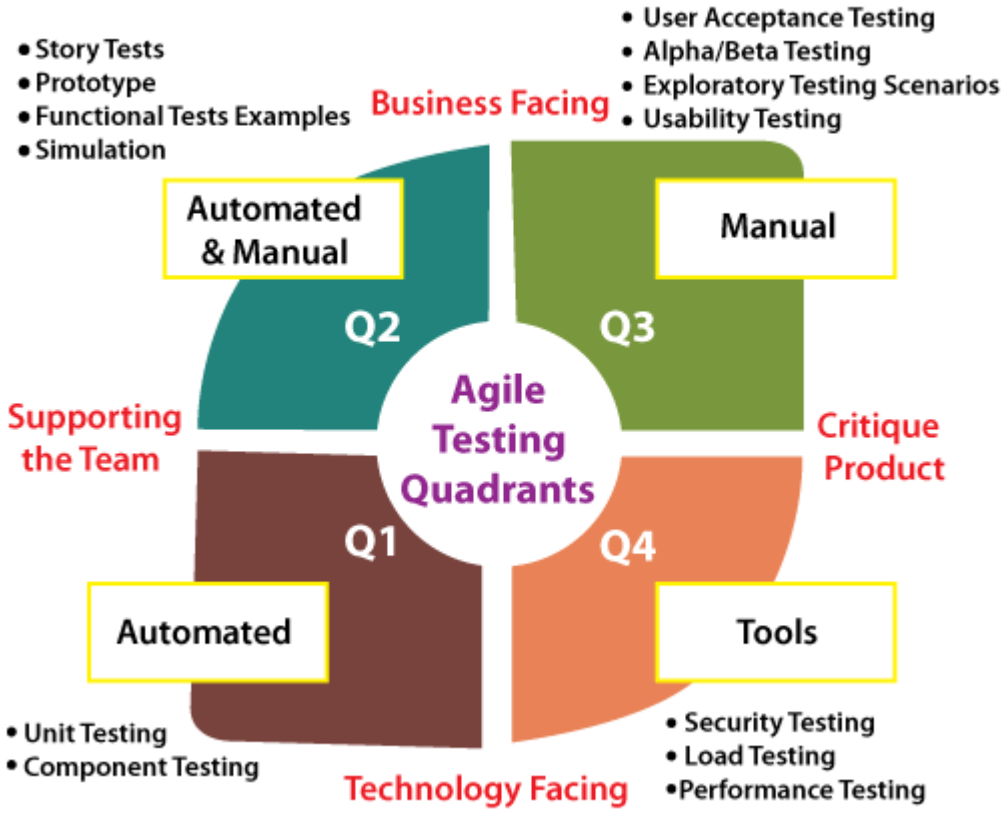
The subsequent agile testing is **crystal methodologies**. This methodology mainly emphasizes recording, cyclic delivery, and wrap-up, which is to make sure during the various analyses.

AGILE TESTING QUADRANTS

It has different quadrants to easily understand agile testing, which divides the whole testing process into **four** parts.

In addition to the four quadrants, the left two specify the test engineer that code to write, and the right two quadrants help them understand the code improved with the support of response to the left quadrants.

These agile testing quadrants may be understood as a traditional process or strategies to perform the end-to-end agile testing of a software application in four different stages, as we can see in the following image:



Let us discuss them one by one to understand the process of agile testing:

1. **Quadrant 1 (Automated)**
2. **Quadrant 2 (Automated and manual)**
3. **Quadrant 3 (Manual)**
4. **Quadrant 4 (Tools)**

Quadrant 1 (Automated)

In the first quadrant of Agile testing, we will see mainly emphasis on the quality of the code. We can say internal code quality, which contains the test cases and test components that is executed by the test engineers.

And these test cases are technology-driven and used for automation testing in order to enhance the code and support the testing team to perform their tasks.

All through the first quadrant of agile testing, we can execute the following testing:

- Unit Testing
- Component Testing

QUADRANT 2 (AUTOMATED AND MANUAL)

In the second quadrant of Agile testing, we will see mainly emphasis on the customer requirements given to the team before and throughout the testing procedures, which expands the business results of the newly created software.

The test case involved in this second quadrant is **business-driven**, usually manual and automated functional tests, prototypes, and examples of test scenarios performed by the testing team.

In quadrant 2, we can execute the following tests:

- Testing scenarios which may occur and workflow
- Implementing the pair testing
- Testing the user story and experiences like prototypes.

Quadrant 3 (Manual)

The third quadrant of agile testing primarily emphasizes the response for the previous two phases (**Quadrant 1 and Quadrant 2**).

The execution of agile testing involves many iterations. And in this quadrant, these reviews and responses of the particular iterations are sustained that helps to strengthen the code.

To test the user experience and determine business results allows the testing team to learn as the test develops.

The team, business owners, and even customers realistically use the product. In the third quadrant, the test cases have been designed to implement automation testing, which helps us develop certainty in the particular product.

In quadrant 3, below types of testing can be executed:

- **Usability testing**
- **Collaborative testing**
- **Exploratory testing**
- **User acceptance testing**
- **Pair testing with customers**

Quadrant 4 (Tools)

The last and fourth Quadrant of agile testing primarily emphasizes the product's non-functional requirements, including compatibility, performance, security, and constancy.

In other words, we can say that the fourth Quadrant ensures that the code fulfils all the non-functional requirements.

Like other Quadrants, various types of testing are performed in quadrant 4 to deliver the non-functional qualities and the expected value.

- **Non-functional testing such as Stress testing, performance testing**, and **load testing, etc.**
- Scalability testing
- **Security Testing**
- **Data Migration Testing**
- Infrastructure testing

AGILE TEST PLAN

As compared to the waterfall model, the agile test plan is created and updated for every release. Furthermore, the agile test plan contains those types of testing executed in a specific iteration, such as test environments, test data requirements, test results, and infrastructure.

The agile test plans emphasize the following:

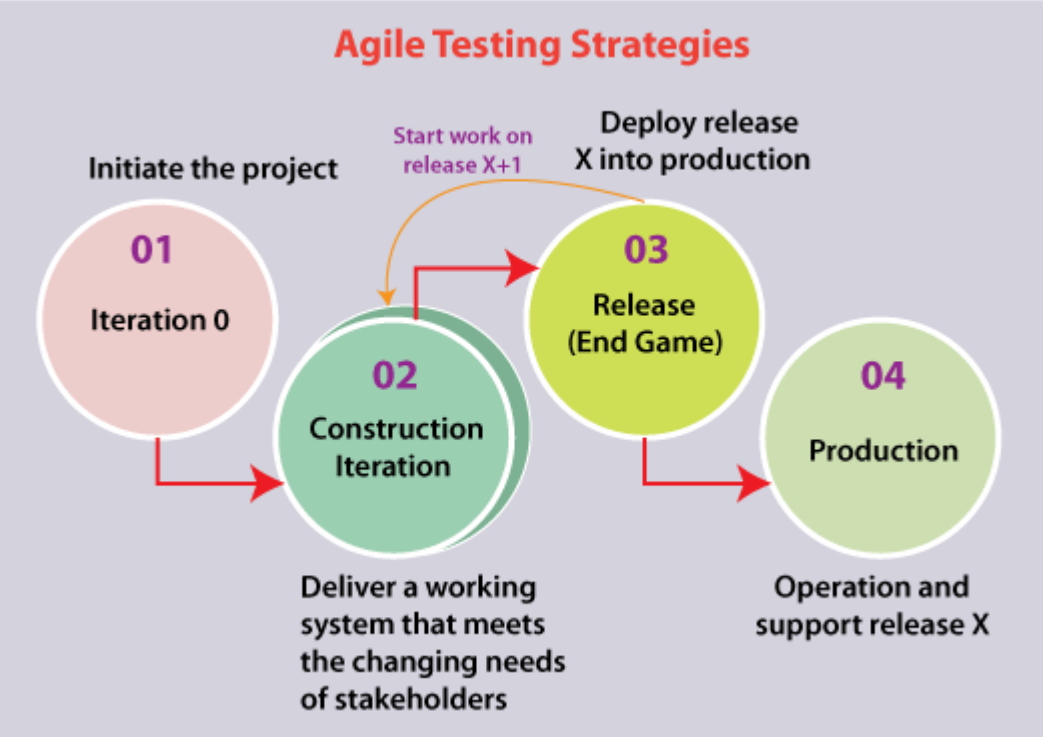
- **Testing Scope:** The testing scope specifies the sprint goals, test scope, and test coverage in which the test will be implemented.
- **Performance and Load Testing:** Here, it specifies the different testing methods and procedures.
- **Types of testing or levels as per the feature's complexity**: It defines those types of testing or levels of testing which are going to be used. And also specifies the data and configurations for the test and the environment in which the test will be executed.
- **Mitigation or Risks Plan:** It defines the backup plan prepared to overcome the risks or issues. And it also identifies the challenges which might face at the time of testing of the application in the current release.
- **Deliverables and Milestones:** It sets the deliverables and milestones of the tests as per the customer's perspective.
- **Infrastructure Consideration:** It governs the infrastructure which is required to perform the tests.
- **Resourcing:** It lists out the test tasks and the occurrence of tests, which defines how many times the tests will be executed.
- **Establish the New functionalities** which are being tested.

AGILE TESTING STRATEGIES

Agile testing has four different approaches, which help us to enhance our product quality.

1. **Iteration 0**
2. **Construction iteration**
3. **Release End Game or Transition Phase**
4. **Production**

Let us discuss them one by one in detail:



1. Iteration 0

The first strategy or approach of agile testing is **iteration 0.** In this, we execute the preliminary setup tasks such as **finding people for testing, establishing testing tools, preparing resources or usability testing lab, etc.**

In Iteration 0, the below steps are accomplished:

- Verifying a business case for the project and boundary situations, and the project scope.
- Summarize the important requirements and use cases that will determine the strategic trade-offs.

- Plan the initial project and cost valuation
- Detecting the risk.
- Outline one or more candidate designs

2. CONSTRUCTION ITERATION

The next strategy of agile testing is **Construction Iteration**. During this approach, the majority of the testing is performed.

The construction iteration is performed as a set of iterations in order to create an increment of the solution.

In simple words, we can say that the agile team follows the listed requirement within each iteration where they can acquire the most significant business needs or requirements left behind from the work item stack and then execute them.

The construction iteration process divided into the following two types of testing:

- **Confirmatory Testing**
- **Investigative Testing**

1. CONFIRMATORY TESTING

To ensure that the product meets all the stakeholders' requirements, we will execute the confirmatory testing.

Confirmatory testing can be further separated into another two types of testing, which are as follows:

- **Agile Acceptance Testing**
- **Developer Testing**

Agile Acceptance Testing: It is a combination of functional testing and acceptance testing. The agile acceptance testing can be executed by the development team and stakeholders together.

Developer Testing: It is a combination of unit testing and integration testing. And it validates both the application code as well as the database schema.

Note: We can automate both (agile acceptance testing and developer testing) to ensures that continuous regression testing has occurred.

2. Investigative Testing

In order to test deep and identify all the issues, which are overlooked in **confirmatory testing,** we will execute the **investigative testing.**

3. RELEASE END GAME OR TRANSITION PHASE

The third approach of agile testing is **release**. The objective of this specific approach is to implement our system effectively in production.

The test engineer will be working on its defect stories in the end game. In the release end game or transition stage, we have the following activities:

- Support Individuals
- Training of end-users
- Operational People

Similarly, it involves some additional activities as well:

- Back-up and Restoration
- Marketing of the product release
- User documentation
- Completion of system

The last agile methodology testing stage encompasses whole system testing and acceptance testing. To complete our final testing phase without having any

difficulties, we should have to test the product more thoroughly in construction iterations.

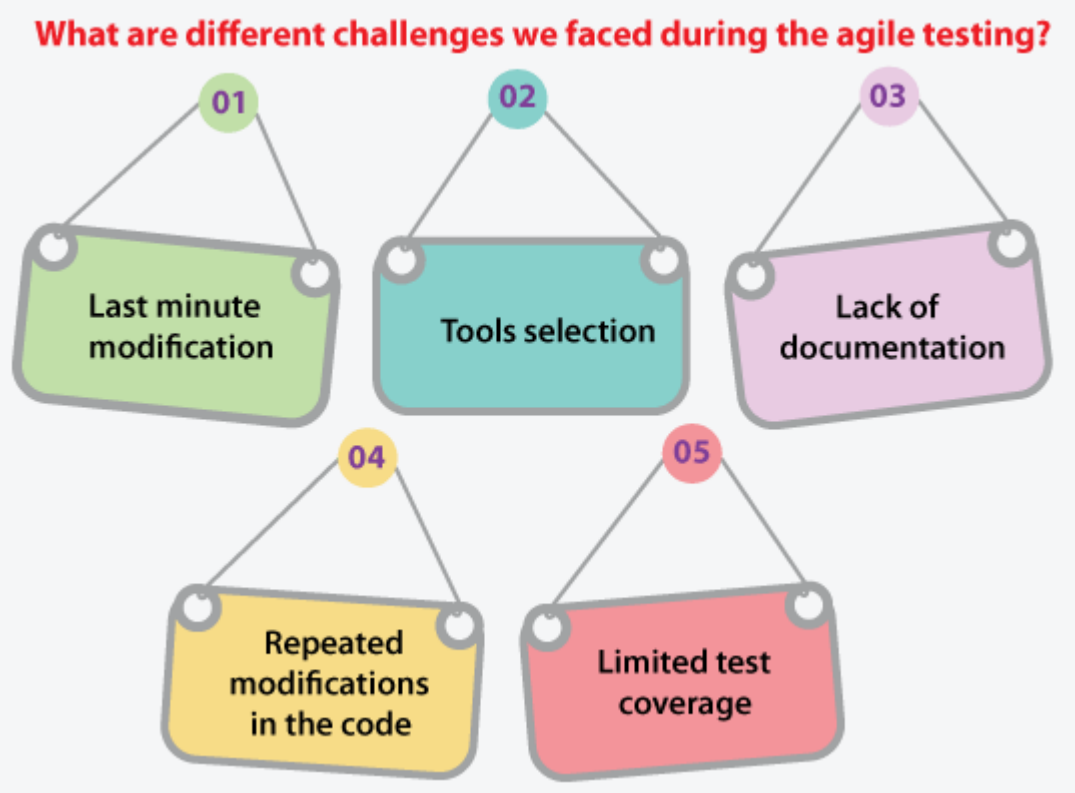
4. Production

The product will move on to the **production stage** as soon as the **release stage** is completed.

WHAT ARE THE DIFFERENT CHALLENGES WE FACED DURING THE AGILE TESTING?

Generally, while performing agile testing, a testing team may face some challenges. Let see those challenges all together for our better understanding:

- **Last-minute modification**
- **Tools selection**
- **Lack of documentation**
- **Repeated modifications in the code**
- **Limited test coverage**



- **Last-minute Modification**

The most faced challenges during the agile testing are last-minute modifications by the client, which gives significantly less time to the testing team to design the test plan, which may affect the product quality. And sometimes, the test engineer is often required to play a semi-developer role.

- **Tools Selection**

The selection of tools during agile testing is essential because if we select the wrong tool, it will waste our time as well as money.

As we already knew, the Test execution cycles are highly reduced, and for the regression testing, we will have minimal timing.

- **Lack of Documentation**

Another frequently faced challenge while executing agile testing is the lack of documentation. The probabilities of error are more agile as documentation is given less importance and ultimately puts more burden on the testing team.

- **Repeated Modifications in the code**

In an agile method, requirement modification and updation are fundamental, making it the major challenge for the Quality assurance team.

- **Limited Test Coverage**

In agile testing, new features are initiated quickly, decreasing the available time for the testing teams to find whether the latest features are as per the requirement and address the business suits.

How do we overcome Agile testing challenges?

As we understood from the definition of Agile Testing, it comprises less or no documentation, which creates problems for the testing team to predict the expected results and becomes the obstacle in the testing process.

And it also makes it challenging to choose the direction and path of the testing to be performed. Hence, to overcome the agile testing challenges, we can implement the following best options:

- We can execute the Exploratory Testing to conquer the agile testing challenges.
- We can perform the automated unit tests to speed up the agile testing process.
- Test-Driven Development could be a good option in order to overcome the agile testing challenges.
- Furthermore, we can overcome these issues or challenges with the help of the agile testing specification and make sure to perform improved and qualitative Testing in a guided way.
- We can implement automated Regression Testing.