tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Agile Testing is a software testing practice that follows the principles of agile software development.

Agile Testing involves all members of the project team, with special expertise contributed by testers. Testing is not a separate phase and is interwoven with all the development phases such as requirements, design and coding and test case generation. Testing takes place simultaneously through the Development Life Cycle.

Audience

The target audience for this tutorial is Software Testing Professionals, Software Quality Experts, and Software Developers.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of software development life cycle (SDLC). A basic understanding of software testing (manual or automation) will be beneficial.

Copyright & Disclaimer

Table of Contents

1. Agile Testing – Overview

Agile is an iterative development methodology, where both development and testing activities are concurrent. Testing is not a separate phase; Coding and Testing are done interactively and incrementally, resulting in quality end product, which the meets customer requirements. Further, continuous integration results in early defect removal and hence time, effort and cost savings.

Agile Manifesto

The Agile Manifesto was published by a team of software developers in 2001, highlighting the importance of the development team, accommodating changing requirements and customer involvement.

The Agile Manifesto is:

We are uncovering better ways of developing software by doing it and helping others do it. Through this work, we have come to value-

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

That is, while there is value in the items on the right, we value the items on the left more.

What is Agile Testing?

Agile Testing is a software testing practice that follows the principles of agile software development.

Agile Testing involves all members of the project team, with special expertise contributed by testers. Testing is not a separate phase and is interwoven with all the development phases such as requirements, design and coding and test case generation. Testing takes place simultaneously through the Development Life Cycle.

Furthermore, with testers participating in the entire Development Lifecycle in conjunction with cross-functional team members, the contribution of testers towards building the software as per the customer requirements, with better design and code would become possible.

Agile Testing covers all the levels of testing and all types of testing.

Agile Testing Vs. Waterfall Testing

In a Waterfall Development methodology, the Development Life Cycle activities happen in phases that are sequential. Thus, testing is a separate phase and gets initiated only after the completion of the development phase.

Following are the highlights of differences between Agile Testing and Waterfall Testing-

Agile Testing	Waterfall Testing
Testing is not a separate phase and occurs concurrently with development.	Testing is a separate phase. All levels and types of testing can begin only after the completion of development.
Testers and developers work together.	Testers work separately from developers.
Testers are involved in coming up with requirements. This helps in requirements mapping to the behaviors in the real world scenario and also framing the acceptance criteria. Also, logical Acceptance Test Cases would be ready along with the requirements.	Testers may not be involved in the requirements phase.
Acceptance Testing is done after every iteration and customer feedback is sought.	Acceptance Testing is done only at the end of the project.
Every iteration completes its own testing thus allowing regression testing to be implemented every time new functions or logic are released.	Regression Testing can be implemented only after the completion of development.
No time delays between coding and testing.	Usual time delays between coding and testing.
Continuous testing with overlapping test levels.	Testing is a timed activity and test levels cannot overlap.
Testing is a best practice.	Testing is often overlooked.

Agile Testing Principles

The principles of Agile testing are-

- **Testing moves the project forward**: Continuous testing is the only way to ensure continuous progress. Agile Testing provides feedback on an ongoing basis and the final product meets the business demands.

- **Testing is not a phase**: Agile team tests alongside the development team to ensure that the features implemented during a given iteration are actually done. Testing is not kept for a later phase.

- **Everyone tests**: In agile testing, the entire team including analysts, developers, and testers test the application. After every iteration, even the customer performs the User Acceptance Testing.

- **Shortening Feedback Loops**: In Agile Testing, the business team get to know the product development for each and every iteration. They are involved in every iteration. Continuous feedback shortens the feedback response time and thus the cost involved in fixing it is less.

- **Keep the Code Clean**: The defects are fixed as they are raised within the same iteration. This ensures clean code at any milestone of development.

- **Lightweight Documentation**: Instead of comprehensive test documentation, Agile testers-

    - Use reusable checklists to suggest tests.
    - Focus on the essence of the test rather than the incidental details.
    - Use lightweight documentation styles/tools.
    - Capture test ideas in charters for exploratory testing.
    - Leverage documents for multiple purposes.

- **Leveraging one test artifact for manual and automated tests**: Same test script artifact can be utilized for manual testing and as an input for automated tests. This eliminates the requirement of Manual Test Documentation and then an equivalent Automation Test Script.

- "**Done Done," not just done**: In Agile, a feature is said to be done not after development but after development and testing.

- **Test-Last vs. Test Driven**: Test Cases are written along with the requirements. Hence, development can be driven by testing. This approach is called Test Driven Development (TDD) and Acceptance Test Driven Development (ATDD). This is in contrast to testing as a last phase in Waterfall Testing.

Agile Testing Activities

The Agile Testing Activities at Project Level are-

- Release Planning (Test Plan)
    - For every Iteration,
    - Agile Testing Activities during an Iteration

- Regression Testing
- Release Activities (Test Related)

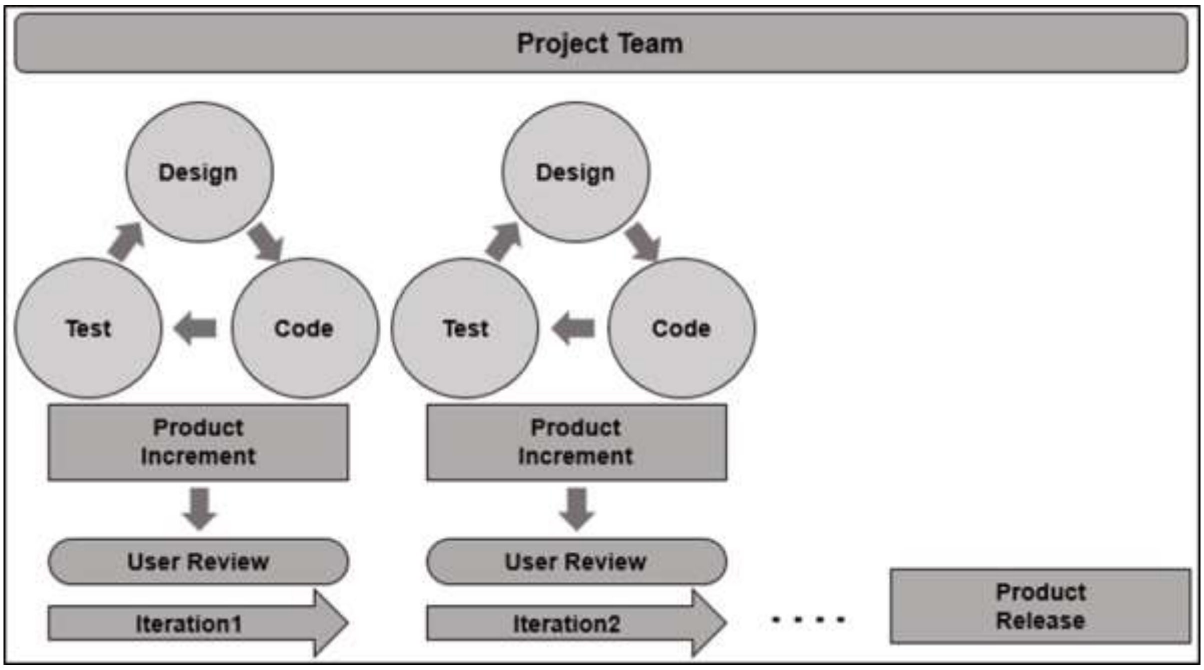
The Agile Testing Activities during an iteration include-

- Participating in iteration planning
- Estimating tasks from the view of testing
- Writing test cases using the feature descriptions
- Unit Testing
- Integration Testing
- Feature Testing
- Defect Fixing
- Integration Testing
- Acceptance Testing
- Status Reporting on Progress of Testing
- Defect Tracking

2.Agile Testing – Agile Methodologies

Agile is an iterative development methodology, where the entire project team participates in all the activities. The requirements evolve as the iterations progress, through collaboration between the customer and the self-organizing teams. As Coding and Testing are done interactively and incrementally, during the course of development, the end-product would be of quality and ensures customer requirements.

Every iteration results in an integrated working product increment and is delivered for User Acceptance Testing. The customer feedback thus obtained would be an input to the next / subsequent Iterations.



Continuous Integration, Continuous Quality

Continuous Integration is the key for Agile Development success. Integrate frequently, at least daily such that you are ready for a release as and when required. Testing in Agile becomes an essential component of all the phases of the development, ensuring continuous quality of the product. Constant feedback from everyone involved in the project adds to the quality of the product.

In Agile, communication is given utmost importance and the customer requests are received as and when necessary. This gives the satisfaction to the customer that all the inputs are considered and working quality product is available throughout the development.

Agile Methodologies

There are several Agile Methodologies that support Agile Development. The Agile Methodologies include-

Scrum

Scrum is an Agile development method that emphasizes on team-centric approach. It advocates participation of the entire team in all the project development activities.

XP

eXtreme Programming is customer-centric and focuses on constantly changing requirements. With frequent releases and customer feedback, the end-product will be of quality meeting customer requirements that are made clearer during the process.

Crystal

Crystal is based on chartering, cyclic delivery and wrap up.

- Chartering involves forming a development team, carrying out a preliminary feasibility analysis, arriving at an initial plan and the development methodology.
- Cyclic Delivery with two or more delivery cycles focuses on the development phase and final integrated product delivery.
- During Wrap up, deployment into the user environment, post-deployment reviews and reflections are performed.
-

FDD

Feature Driven Development (FDD) involves designing and building features. The difference between FDD and other Agile Development Methodologies is that the features are developed in specific and short phases separately.

DSDM

Dynamic Software Development Method (DSDM) is based on Rapid Application Development (RAD) and is aligned to the Agile Framework. DSDM focuses on frequent delivery of the product, involving users actively and empowering the teams to make quick decisions.

Lean Software Development

In Lean Software Development, focus is on eliminating waste and giving value to the customer. This results in rapid development and product of value.

Waste includes partially done work, irrelevant work, features that are not used by the customer, defects, etc. that add to delays in delivery.

The **Lean Principles** are-

- Eliminate Waste
- Amplify Learning
- Delay Commitment
- Empower the Team
- Deliver Fast
- Build Integrity in
- See the Whole

Kanban

Kanban focuses on managing work with an emphasis on just-in-time (JIT) delivery, while not overloading the team members. The tasks are displayed for all the participants to see and for the Team Members to pull work from a queue.

Kanban is based on:

- Kanban Board (Visual and Persistent across the Development)
- Work-in-progress (WIP) Limit
- Lead Time

Agile Testing Methodologies

The testing practices are well defined for every project, whether Agile or not, to deliver quality products. Traditional Testing principles are quite often used in Agile Testing. One of them is Early Testing that focuses on-

- Writing Test Cases to express the behavior of the system.
- Early Defect Prevention, detection and removal.
- Ensuring that the right test types are run at the right time and as part of the right test level.

In all the Agile Methodologies we discussed, Agile Testing in itself is a Methodology. In all the approaches, Test Cases are written before Coding.

In this tutorial, we will focus on Scrum as the Agile Testing Methodology.

The other commonly used Agile Testing Methodologies are-

- **Test-Driven Development (TDD):** Test-Driven Development (TDD) is based on coding guided by tests.

- **Acceptance Test-Driven Development (ATDD):** Acceptance Test-Driven Development (ATDD) is based on communication between the customers, developers and testers and driven by pre-defined Acceptance Criteria and Acceptance Test Cases.

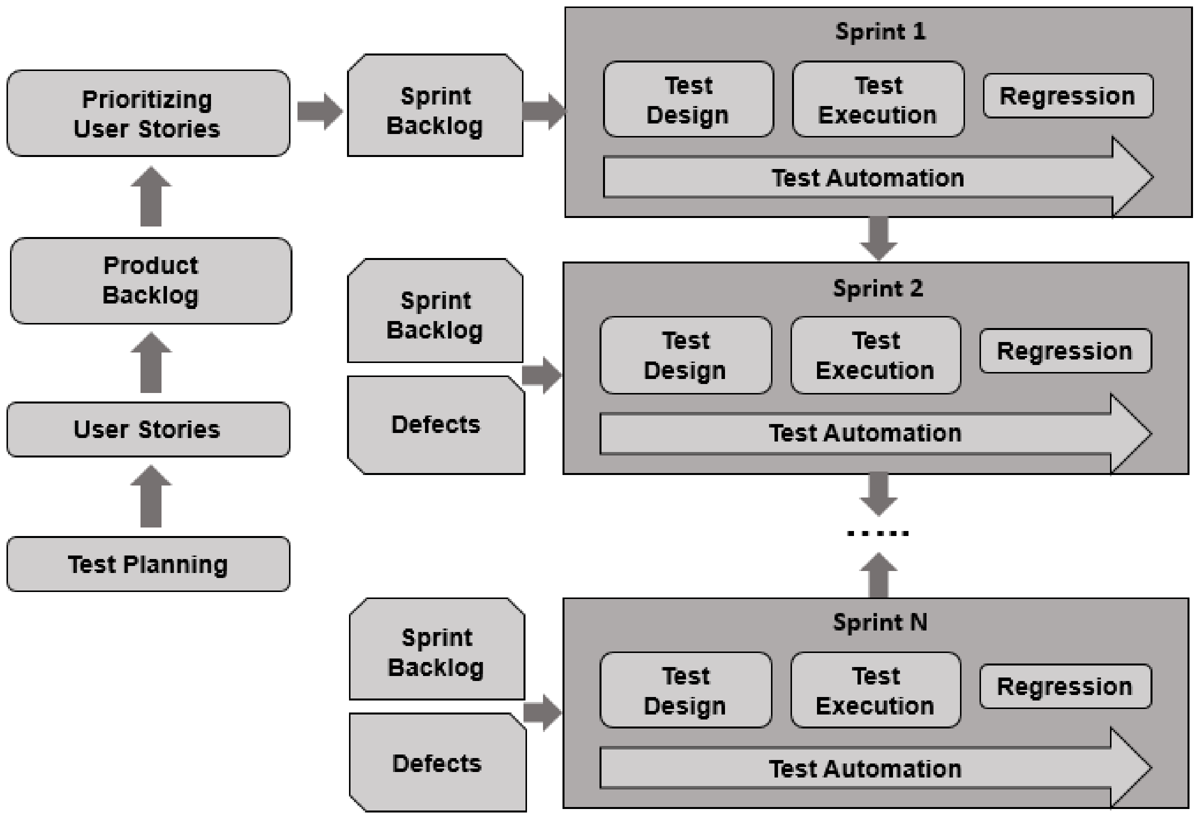
- **Behavior-Driven Development (BDD):** In Behavior-Driven Development (BDD) testing is based on the expected behavior of the software being developed.

Agile Testing Lifecycle

In Scrum, the Testing activities include-

- Contributing to User Stories based on the expected behavior of the System depicted as Test Cases
- Release Planning based on Test Effort and Defects
- Sprint Planning based on User Stories and Defects
- Sprint Execution with Continuous Testing
- Regression Testing after the completion of Sprint
- Reporting Test Results
- Automation Testing

Testing is iterative and sprints based as depicted in the diagram given below-



3. Agile Testing – Tester in Agile Team

Agile Development is team-centric and developers and testers take part in all the project and development activities. The Teamwork maximizes success of testing in Agile projects.

A Tester in Agile team has to participate and contribute to all the project activities and at the same time has to leverage the expertise in testing.

An Agile tester should have traditional testing skills. In addition, Agile tester needs-

- Good interpersonal skills.
- Ability to act positive and solution-oriented with team members and stakeholders.
- Ability to display critical, quality-oriented, skeptical thinking about the product.
- Aptitude to be pro-active to actively acquire information from the stakeholders.
- Skills to work effectively with customers and stakeholders in defining testable User Stories, the Acceptance Criteria.
- Talent to be a good team member working with developers in producing quality code.
- Usability of testing skills to have the right test cases at the right time and at the right level and executing them well within the duration of the sprint.
- Ability to evaluate and report test results, test progress and the product quality.
- Openness to respond to changes quickly, including changing, adding or improving test cases.
- Potential to self-organize work.
- Enthusiasm to continuous skill growth.
- Competency in Test Automation, Test-driven Development (TDD), Acceptance Test-driven Development (ATDD), Behavior Driven Development (BDD) and experience based Testing

Role of Tester in Agile Team

Tester in Agile Team participates in all the project and development activities to contribute the best of the testing expertise.

Agile Tester Activities include-

- Ensuring proper use of testing tools.
- Configuring, using and managing the test environments and the test data.
- Mentoring other team members in relevant aspects of testing.
- Ensuring that appropriate testing tasks are scheduled during the release and sprint planning.
- Understanding, implementing and updating test strategy.
- Collaborating with developers, customer and stakeholders in clarifying requirements, in terms of testability, consistency and completeness.
- Performing the right tests at the right time and at right test levels.
- Reporting defects and working with the team in resolving them.
- Measuring and reporting test coverage across all applicable coverage dimensions.
- Participating in sprint retrospectives, proactively suggesting and implementing improvements.

In the Agile Lifecycle, a tester plays a significant Role in-

- Teamwork
- Test Planning
- Sprint Zero
- Integration
- Agile Testing Practices

Teamwork

In Agile Development, teamwork is fundamental and hence requires the following-

- **Collaborative Approach**: Working with cross-functional team members on Test Strategy, Test Planning, Test Specification, Test Execution, Test Evaluation, and Test Results Reporting. Contributing the testing expertise in conjunction with other team activities.
- **Self-organizing**: Planning and organizing well within the sprints to achieve the targets of testing by amalgamating expertise from other team members as well.

- **Empowerment**: Making appropriate technical decisions in achieving the team's goals.
- **Commitment**: Committing to understanding and evaluating the product's behavior and characteristics as required by the customers and stakeholders.
- **Transparent**: Open, Communicating and Accountable.
- **Credibility**: Ensuring the credibility of the test strategy, its implementation, and execution. Keeping the customers and stakeholders informed on the test strategy.
- **Open to Feedback**: Participating in sprint retrospectives to learn from both successes and failures. Seeking customer feedback and acting quickly and appropriately to ensure quality deliverables.
- **Resilient**: Responding to changes.

Test Planning

Test Planning should start during the release planning and update during each sprint. Test planning should cover the following tasks-

- Defining test scope, extent of testing, test and sprint goals.
- Deciding on the test environment, test tools, test data and configurations.
- Assigning testing of features and characteristics.
- Scheduling test tasks and defining frequency of tests.
- Identifying test methods, techniques, tools and test data.
- Ascertaining prerequisites such as predecessor tasks, expertise and training.
- Identifying dependencies such as functions, code, system components, vendor, technology, tools, activities, tasks, teams, test types, test levels and constraints.
- Setting priorities considering the customer/user importance and dependencies.
- Arriving at the time duration and effort required to test.
- Identifying tasks at each sprint planning.

Sprint Zero

Sprint Zero involves preparation activities before the first sprint. A tester needs to collaborate with the team on the following activities-

- Identifying scope
- Dividing user stories into sprints
- Creating system architecture

- Planning, acquiring and installing tools (including testing tools)
- Creating the initial test strategy for all the test levels
- Defining test metrics
- Specifying the acceptance criteria, also called the definition of "Done"
- Defining exit criteria
- Creating Scrum board
- Setting the direction for testing throughout the sprints

Integration

In Agile, a quality working product should be ready for release at any point of time in the development lifecycle. This implies continuous integration as a part of development. An Agile tester needs to support continuous integration with continuous testing.

To accomplish this, a tester needs to-

- Understand the integration strategy.
- Identify all dependencies between functions and features.

Agile Testing Practices

An Agile tester needs to adapt Agile practices for testing in an agile project.

- **Pairing:** Two team members work together at the same keyboard. As one of them tests, the other reviews/analyzes testing. The two team members can be
    - One tester and one developer
    - One tester and one business analyst
    - Two testers
- **Incremental Test Design**: Test cases are built from user stories, starting with simple tests and moving to more complex tests.

- **Mind Mapping**: A mind map is a diagram to organize the information visually. Mind mapping can be used as an effective tool in Agile testing, using which information regarding the necessary test sessions, test strategies and test data can be organized.

4. Agile Testing – Tracking Testing Activities

Test Status can be communicated

- During daily stand-up meetings
- Using standard test management tools
- Via messengers

Test status determined by test passing status is crucial in deciding the whether the task is “Done”. Done means all the tests for the task pass.

Test Progress

Test Progress can be tracked using-

- Scrum Boards (Agile Task Boards)
- Burndown Charts
- Automated Test Results

Test Progress also has a direct impact on development progress. This is because a User Story can be moved to **Done** status only after the Acceptance Criteria is reached. This, in turn, is decided by Test Status as the Acceptance Criteria is judged by a Test Status.

If there are any delays or blockages in test progress, the entire team discusses and works collaboratively to resolve the same.

In Agile Projects, changes take place quite often. When many changes take place, we can expect that the Test Status, Test Progress and Product Quality to evolve constantly. The Agile testers need to get that information to the team so that the appropriate decisions can be made at the right time to stay on track for successful completion of each iteration.

When changes happen, they can affect existing features from previous iterations. In such cases, manual and automated tests must be updated to deal effectively with regression risk. Regression testing is also needed.

Product Quality

Product Quality Metrics include-

- Tests Pass / Fail
- Defects Found / Fixed
- Test Coverage
- Test Pass/Fail Rates
- Defect Discovery Rates
- Defect Density

Automating the gathering and reporting of product quality metrics helps in-

- Maintaining transparency.
- Gathering all the relevant and required metrics at the right time.
- Immediate reporting without communication delays.
- Allowing testers to focus on testing.
- Filtering misuse of metrics.

To secure overall product quality, the Agile team needs to obtain customer feedback on whether the product meets customer expectations. This needs to be carried out at the end of each iteration, and the feedback will be an input for subsequent iterations.

Key Success Factors

In Agile projects, quality products can be delivered if Agile testing is successful.

The following points need to be considered for the success of Agile testing-

- Agile testing is based on test first and continuous testing approaches. Hence, the traditional testing tools, which are built on test-last approach, may not be suitable. Hence, while choosing the Testing Tools in Agile projects, the alignment to Agile testing needs to be verified.

- Reduce total testing time by automating tests earlier in the development lifecycle.

- Agile testers need to maintain their pace to align to the development release schedule. Hence, proper planning, tracking, and re-planning of the testing activities need to be done on the fly with product quality as the goal.

- Manual testing accounts to 80% of the testing in the projects. Hence, testers with expertise are need to be part of the Agile team.

- Participation of these testers with expertise throughout the development lifecycle makes the entire team focus on quality product meeting customer expectations-

    - Defining user stories emphasizing product behavior expected by the end users

    - Identifying Acceptance Criteria at user story level / task level as per customer expectations

    - Effort and duration estimation for testing activities

    - Planning testing activities

    - Aligning with the development team to ensure production of code that meets the requirements with an upfront test design

    - Test first and continuous testing to ensure that done status is reached meeting the acceptance criteria at the expected time

- Ensuring testing at all levels within the sprint
- Regression testing at the end of each sprint
- Collecting and analyzing product metrics that are useful for the success of the project
- Analyzing defects to identify which need to be fixed in the current Sprint and which can be delayed to subsequent Sprints
- Focusing on what is important from the Customer's point of view

Lisa Crispin has defined seven key Factors for Agile Testing Success-

- **Whole Team approach**: In this kind of approach, the developers train the testers and the testers train other team members. This helps everyone to understand every task in the project, thereby collaboration and contribution will have maximum benefit. Collaboration of testers with customers is also an important factor to set their expectations right at the beginning and translating the acceptance criteria to the required to pass the test.

- **Agile Testing Mindset**: The testers are proactive in continually improving the quality and collaborating constantly with the rest of the team.

- **Automate Regression Testing**: Design for testability and drive development with tests. Start simple and allow the team to choose the tools. Be ready to provide advice.

- **Provide and Obtain Feedback:** As this is a core Agile value, the entire team should be open for feedback. As the testers are expert feedback providers, need to focus on relevant and necessary information. In return, on obtaining feedback should accommodate test case changes and testing.
- **Build a Foundation of Core Agile Practices:** Focus on testing alongside coding, continuous integration, collaborative test environments, working incrementally, acceptance for changes, maintaining synergy.

- **Collaborate with Customers:** Elicit examples, understanding, and checking the requirements mapping to the product behavior, setting up Acceptance Criteria, obtaining feedback.

- **Look at the Big Picture:** Drive development with business-facing tests and examples using real world test data and thinking about impacts on other areas.

5.Agile Testing – Significant Attributes

In this chapter, we will see some significant attributes of Agile Testing.

Agile Testing Benefits

The benefits of Agile testing are-

- Customer satisfaction by quick, continuous completely tested product and seeking customer feedback
- Customers, developers, and testers continuously interact with one another, thereby reducing the cycle time
- Agile testers participate in defining requirements contributing their testing expertise to focus on what is workable
- Agile testers participate in estimation assessing testing effort and time.
- Early test design reflecting Acceptance Criteria
- Testing requirements consolidated by the whole team, avoiding drawbacks.
- Constant focus on quality of the product by the entire team
- Definition of **Done** status reflecting tests pass ensures that the requirement is met.
- Continuous feedback on delays or blockages so that resolution can be made immediately with effort from the whole team
- Quick responses to changing requirements and accommodating them soon
- Continuous integration driven regression testing
- No time delays between development and testing. test first, continuous testing approaches are followed
- Automation testing implemented early in the development lifecycle, thereby reducing total testing time and effort

Best Practices in Agile Testing

Follow the best practices given below-

- Inclusion of testers with expertise in all types of testing at all levels
- Testers participating in the definition of requirements, collaborating with customers on the expected behavior of the product

- Testers sharing feedback continuously with the developers and customer
- Test first and continuous testing approaches to align to the development work
- Tracking test status and test progress promptly and constantly with focus on delivering quality product
- Automation testing early in the development lifecycle to reduce cycle time
- To perform Regression Testing leverage Automation Testing as an effective way

Challenges in Agile Testing

The following challenges exist in Agile testing-

- Failure to understand the Agile approach and its limitations by the Business and Management can lead to unachievable expectations.
- Agile follows whole-Team approach, but not everyone knows the essentials of Testing Practices. Testers are advised to coach the others, but in real scenario can be impracticable with time-boxed Sprints (Iterations).
- Test First Approach requires Developers to base the coding on Tester Feedback, but in real scenarios Developers are more accustomed to base the coding on the Requirements coming from Customer or Business.
- Accountability for the Quality Product is with the entire Agile Team, but in initial stages the Developers may not Focus on Quality as they are more into the implementation mode.
- Continuous Integration calls for Regression Testing that requires considerable effort, even if it has to be automated.
- Testers can be adaptable to changes with the Agile mind-set, but accommodating the resulting Test Changes and Testing can be impracticable to target to finish during the Sprint.
- Early Automation is advised so that Manual Testing Effort and Time can be reduced. But, in the real scenario, arriving at the Tests that can be automated and automating them require Time and Effort.

Agile Testing Guidelines

Use the following guidelines while performing Agile Testing.

- Participate in Release Planning to identify the required Testing activities and come up with the initial version of test plan.
- Participate in estimation session to arrive at testing effort and duration so that testing activities are accommodated in the iterations.
- Participate in User Story Definition to arrive at Acceptance Test Cases.
- Participate in every Sprint Planning Meeting to understand the scope and update Test Plan.
- Continuously collaborate with the Development Team during the Sprint to make Testing and Coding a success well within the Sprint.
- Participate in Daily Stand-up Meetings and convey Test Delays or Blockages if any, to receive immediate resolution.
- Track and Report Test Status, Test Progress and Product Quality regularly.
- Be ready to accommodate changes, responding with modifications to Test Cases, Test Data.
- Participate in Sprint Retrospectives to understand and contribute the Best Practices and Lessons Learned.
- Collaborate in obtaining Customer Feedback at each Sprint.

6. Agile Testing – Quadrants

As in the case of Traditional Testing, Agile Testing also need to cover all the Test Levels.

- Unit Testing
- Integration Testing
- System Testing
- User Acceptance Testing

Unit Testing

- Done along with Coding, by Developer
- Supported by Tester who writes Test Cases ensuring 100% Design Coverage
- Unit Test Cases and Unit Testing results need to be reviewed
- Unresolved major defects (as per priority and severity) are not left
- All Unit Tests are automated

Integration Testing

- Done along with Continuous Integration as the Sprints progress
- Done at the end after all the Sprints are completed
- All Functional Requirements are tested
- All Interfaces between Units are tested
- All the Defects are Reported
- Tests are automated where possible

System Testing

- Done as the Development progresses
- Users Stories, Features and Functions are Tested
- Testing done in Production Environment
- Quality Tests are executed (Performance, Reliability, etc.)
- Defects are reported
- Tests are automated where possible

User Acceptance Testing

- Done at the end of each Sprint and at the end of the project
- Done by the Customer. Feedback is taken by the Team

- Feedback will be an input to subsequent Sprints
- User Stories in a Sprint are pre-verified to be testable and are with defined Acceptance Criteria

Test Types

- Component Tests (Unit Tests)
- Functional Tests (User Stories Tests)
- Non-functional Tests (Performance, Load, Stress, etc.)
- Acceptance Tests

Tests can be fully Manual, fully Automated, Combination of Manual and Automated or Manual supported by Tools.

Support Programming & Critique Product Tests

Tests can be for-

- **Supporting Development (Support Programming)**: Support Programming Tests are used by the Programmers-
    - To decide on what code they need to write to accomplish a certain behavior of a System
    - What Tests need to be run after Coding to ensure the new Code does not hamper the rest of the behaviors of the System
- **Verification only (Critique Product)**: Critique Product Tests are used for discovering inadequacies in the finished Product

Business Facing & Technology Facing Tests

To decide on what tests to be performed when, you need to determine whether a test is-

- Business Facing, or
- Technology Facing

Business Facing Tests

A Test is a business-facing test if it answers the questions framed with words from business domain. These are understood by the business experts and would interest them so that behavior of the system can be explained in the real time scenario.

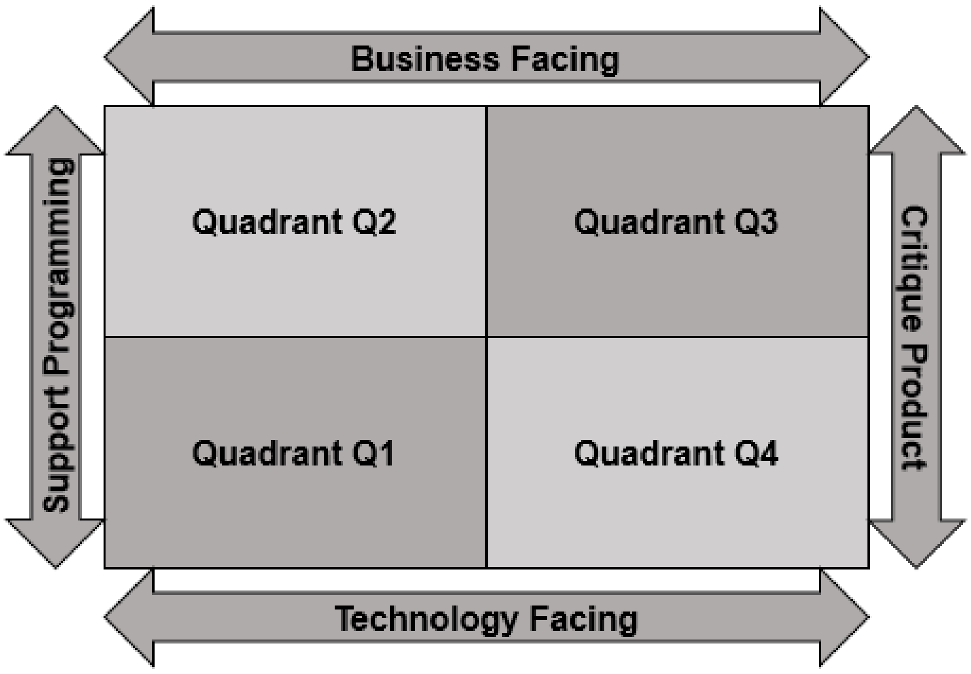
Technology Facing Tests

A Test is a technology-facing test if it answers the questions framed with words from technology domain. The programmers understand what needs to be implemented based on the clarifications on technology.

These two aspects of test types can be viewed using the Agile Testing Quadrants defined by Brian Marick.

Agile Testing Quadrants

Combining the two aspects of Testing Types, the following Agile Testing Quadrants are derived by Brian Marick -



The Agile Testing Quadrants provide a helpful taxonomy to help teams identify, plan and execute the testing needed.

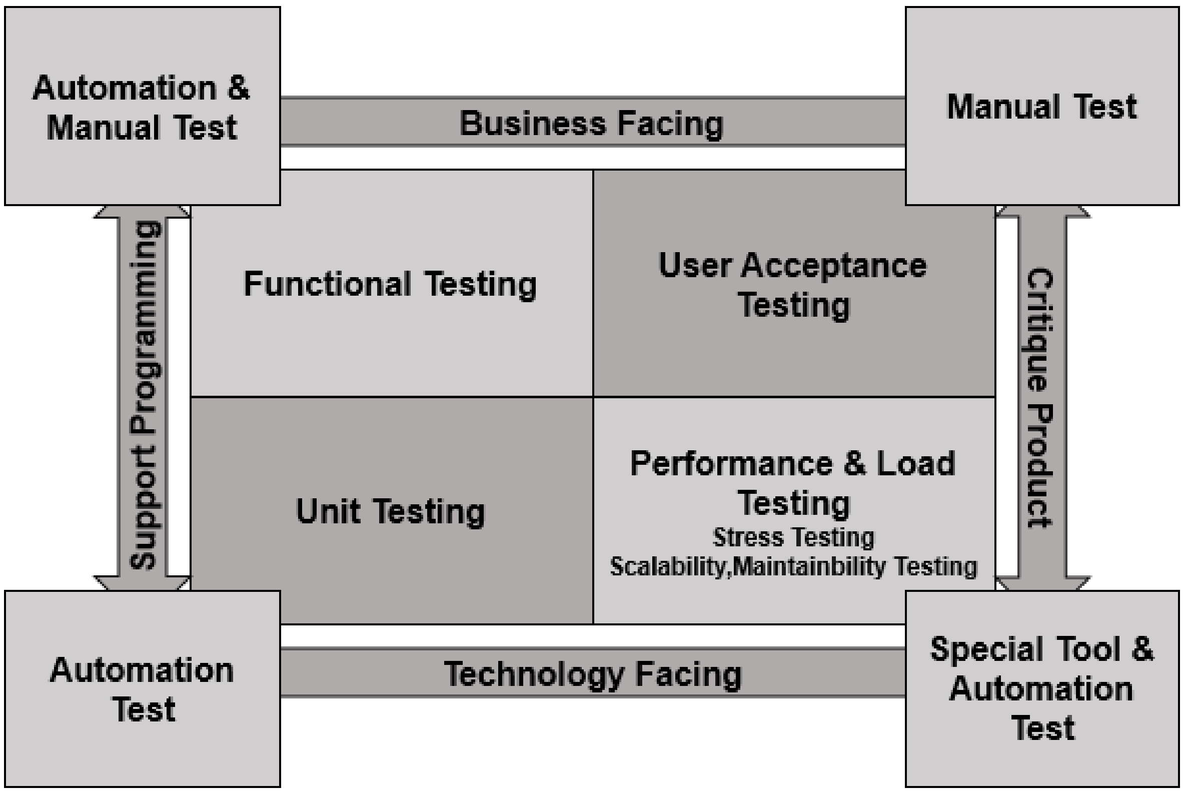
- **Quadrant Q1**: **Unit Level**, **Technology Facing**, and supports the developers. Unit tests belong to this Quadrant. These tests can be Automated tests.

- **Quadrant Q2**: **System level**, **business facing**, and conform product behavior. Functional tests belong to this quadrant. These tests are either manual or automated.

- **Quadrant Q3**: **System or User Acceptance Level, Business Facing** and focus on real time scenarios. User Acceptance Tests belong to this quadrant. These tests are manual.

- **Quadrant Q4**: **System or Operational Acceptance Level**, **Technology Facing** and Focus on Performance, Load, Stress, Maintainability, Scalability Tests. Special tools can be used for these tests along with automation testing.

Combining these, the Agile Testing Quadrants that reflect **What-Testing-When** can be visualized as follows –



7.Agile Testing in Scrum

Scrum advocates **Whole Team Approach**, in the sense that every team member has to take part in every project activity. Scrum team is self-organizing with accountability to the project deliverables. Decision-making is left to the team that results in appropriate actions being taken at the right time without any time delays. This approach also encourages proper use of the team talent instead of restricting to one activity. Testers also participate in all the project and development activities contributing their expertise in testing.

The whole team works together on Test Strategy, Test Planning, Test Specification, Test Execution, Test Evaluation, and Test Results Reporting.

Collaborative User Story Creation

Testers participate in User Story Creation. Testers contribute their ideas on possible behavior of the system. This helps in the customer and/or end user understanding the system in the real environment and thus getting clarity on what actually they want as the outcome. This results in faster freezing of requirements and also reduces the probability of changes in the requirements later on.

Testers also come up with the Acceptance Criteria for every scenario agreed by the customer.

Testers contribute to the creation of testable user stories.

Release Planning

Release Planning is done for the entire project. However, Scrum framework involves iterative decision making as more information is obtained in the due course of executing sprints. Hence, Release Planning session at the beginning of the project need not produce a detailed release plan for the entire project. It can be updated continually, as relevant information is available.

Every sprint-end need not have a release. A release can be after a group of sprints. The main criteria of a release is to deliver business value to the customer. The team decides on the sprint length with release planning as an input.

Release Planning is the basis of test approach and test plan for release. Testers estimate Test Effort and plan Testing for the release. When release plans change, testers must handle changes, obtain an adequate test basis considering larger context of release. Testers also provide testing effort that is required at the end of all the sprints.

Sprint Planning

Sprint planning is done at the beginning of each sprint. The sprint backlog is created with the user stories picked up from the product backlog for implementation in that particular sprint.

The testers should-

- Determine the testability of the user stories selected for the sprint
- Create acceptance tests
- Define test levels
- Identify test automation

Testers update the test plan with the estimates for testing effort and durations in the sprint. This ensures provision of time for required testing during the time-boxed sprints and also accountability of the testing effort.

Test Analysis

When a sprint begins, as the developers carry on story analysis for design and implementation, testers perform test analysis for the stories in the sprint backlog. Tester create the required test cases – both manual and automated tests.

Testing

All the members of the Scrum team should participate in testing.

- The developers execute the unit tests as they develop code for the user stories. Unit Tests are created in every sprint, before the code is written. The unit test cases are derived from low level design specifications.

- The testers perform functional and non-functional features of the user stories.

- The testers mentor the other members in the scrum team with their expertise in testing so that the entire team will have a collective accountability for the quality of the product.

- At the end of the sprint, customer and/or end users carry out User Acceptance Testing and provide feedback to the scrum team. This forms as an input to the next sprint.

- Test results are collected and maintained.

Automation Testing

Automation testing is given high importance in Scrum teams. The testers devote time in creating, executing, monitoring and maintaining of automated tests and results. As changes can occur any time in scrum projects, testers need to accommodate testing of changed features and also the regression testing involved. Automation testing facilitates managing of test effort associated with the changes. Automated tests at all levels facilitate achieving continuous integration. Automated tests run much faster than manual tests at no additional effort.

The manual testing focuses more on exploratory testing, product vulnerability, predicting defects.

Automation of Testing Activities

Automation of testing activities reduces the burden of repeated work and result in cost savings. Automate

- Test Data Generation
- Test Data Loading
- Build Deployment into Test Environment
- Test Environment Management
- Data Output Comparison

Regression Testing

In a sprint, testers test the code that is new / modified in that sprint. However, testers also need to ensure that the code developed and tested in the earlier sprints also is working along with the new code. Hence Regression testing is given importance in scrum. Automated regression tests are run in continuous integration.

Configuration Management

A Configuration management system that uses automated build and test frameworks is used in Scrum projects. This allows to run static analysis and unit tests repeatedly as new code is checked into the Configuration Management System. It also manages continuous integration of the new code with the system. Automated Regression Tests are run during Continuous Integration.

Manual Test Cases, Automated Tests, Test Data, Test Plans, Test Strategy and other Testing Artifacts need to be Version Controlled and require relevant Access Permissions to be ensured. This can be accomplished by maintaining the Testing Artifacts in the Configuration Management System.

Agile Testing Practices

Testers in a Scrum Team can follow the following Agile Practices-

- **Pairing:** Two Team Members sit together and work collaboratively. The two people can be two Testers or one Tester and one Developer.

- **Incremental Test Design**: Test Cases are developed as the Sprints progress incrementally and User Stories are added up.

Agile Metrics

During software development, collection and analysis of metrics help in improving the process and thereby achieving better productivity, quality deliverables and customer satisfaction. In Scrum based development, this is possible and the testers have to pay attention to the metrics that they need to.

Several metrics are suggested for Scrum development. The significant metrics are-

- **Ratio of Successful Sprints**: **(Number of successful Sprints / Total number of Sprints) * 100**. A Successful Sprint is one in which the Team could meet its commitment.

- **Velocity:** A team's Velocity is based on the amount of Story Points a team earned during a sprint. Story Points are the measure of the User Stories counted during estimation.

- **Focus Factor**: **(Velocity / Team's Work Capacity) / 100**. Focus Factor is the percentage of the team's effort that results in finished stories.

- **Estimation Accuracy: (Estimated effort / Actual effort) / 100.** Estimation Accuracy is the Team's ability in estimating the effort accurately.

- **Sprint Burndown:** Work (in Story Points or in hours) that is Remaining Vs. Work that needs to be Remaining ideally (as per the Estimation).
    - If it is more, then it means that the Team has taken up more Work than they can do.
    - If it is less, then it means the Team did not Estimate accurately

- **Defect Count**: Number of defects in a Sprint. Defect count is the amount of defects in the software as against the backlog.

- **Severity of Defects**: Defects can be categorized as minor, major and critical as per their severity. Testers can define the categorization.

Sprint Retrospectives

In Sprint Retrospectives, all the team members will participate. They share

- The things that went well
- Metrics
- The scope for improvements
- Action items to apply

8. Agile Testing – Methods

In Agile Testing, the commonly used Testing methods are from the traditional practices and are aligned to the principle – Test Early. The Test Cases are written before the code is written. The emphasis is on defect prevention, detection, and removal running the right test types at the right time and at right level.

In this chapter, you will get an understanding of the methods-

- Test Driven Development (TDD)
- Acceptance Test Driven Development (ATDD)
- Behavior Driven Development (BDD)

Test Driven Development

In the Test Driven Development (TDD) method, the code is developed based on the Test-first approach directed by Automated Test Cases. A test case is written first to fail, code is developed based on that to ensure that the test passes. Method is repeated, refactoring is done through the development of code.

TDD can be understood with the help of the following steps-

- **Step 1:** Write a Test case to reflect the expected behavior of the functionality of the code that needs to be written.
- **Step 2:** Run the test. The test fails as the code is still not developed.
- **Step 3:** Develop code based on the test case.
- **Step 4:** Run the test again. This time, the test has to pass as the functionality is coded. Repeat Step (3) and Step (4) till the test passes.
- **Step 5:** Refactor the code.
- **Step 6:** Run the test again to ensure it passes.

Repeat **Step 1 – Step 6** adding test cases to add functionality. The added tests and the earlier tests are run every time to ensure the code is running as expected. To make this process fast, tests are automated.

The tests can be at unit, integration or system level. Constant communication between testers and developers needs to be ensured.

Acceptance Test Driven Development

In the Acceptance Test Driven Development (ATDD) method, the code is developed based on the test-first approach directed by Acceptance Test Cases. The focus is on the acceptance criteria and the Acceptance Test Cases written by the testers during User Story Creation in collaboration with the customer, end users and relevant stakeholders.

- **Step 1:** Write Acceptance Test Cases along with user stories in collaboration with the customer and users.
- **Step 2:** Define the associated acceptance criteria.
- **Step 3:** Develop code based on the acceptance tests and acceptance criteria.
- **Step 4:** Run the acceptance tests to ensure that the code is running as expected.
- **Step 5:** Automate the acceptance tests. Repeat **Step 3 – Step 5** until all the user stories in the iteration are implemented.
- **Step 6:** Automate the regression tests.
- **Step 7:** Run the automated Regression Tests to ensure Continuous Regression.

Behavior Driven Development (BDD)

Behavior Driven Development (BDD) is similar to the Test Driven Development (TDD), and the focus is on testing the code to ensure the expected behavior of the system.

In BDD, language like English is used so that it makes sense to the users, testers and developers. It ensures

- Continuous communication among the users, testers and developers.
- Transparency on what is being developed and tested.

9. Agile Testing – Techniques

Testing Techniques from traditional testing can also be used in Agile testing. In addition to these, Agile specific testing techniques and terminologies are used in the Agile projects.

Test Basis

In agile projects, the product backlog replaces the requirements specification documents. The contents of product backlog are normally user stories. The non-functional requirements also are taken care in the user stories. Thus, the test basis in Agile projects is the user story.

To ensure quality testing, the following can also be considered additionally as test basis-

- Experience from previous iterations of the same project or past projects.
- Existing functions, architecture, design, code, and quality characteristics of the system.
- Defect data from the current and past projects.
- Customer feedback.
- User documentation.

Definition of Done

The Definition of Done (DoD) is the criteria that is used in Agile projects to ensure completion of an activity in the Sprint backlog. DoD can vary from one Scrum team to another, but it should be consistent within one team.

DoD is a checklist of necessary activities that ensure implementation of functions and features in a user story along with the non-functional requirements that are part of the user story. A user story reaches the Done stage after all the items in the DoD checklist are accomplished. A DoD is shared across team.

A typical DoD for a user story can contain-

- Detailed Testable Acceptance Criteria
- Criteria to ensure consistency of the User Story with the others in the Iteration
- Specific Criteria related to the Product
- Functional Behavior Aspects
- Non-functional characteristics
- Interfaces
- Test Data Requirements
- Test Coverage

- Refactoring
- Review and Approval Requirements

In addition to the DoD for User Stories, DoD is also required-

- at every Level of Testing
- for each Feature
- for each Iteration
- for Release

Test Information

A tester needs to have the following Test information-

- User Stories that need to be tested
- Associated Acceptance Criteria
- System Interfaces
- Environment where the System is expected to Work
- Tools availability
- Test Coverage
- DoD

In Agile projects, as testing is not a sequential activity and testers are supposed to work in a collaborative mode, it is the tester's responsibility to-

- Obtain necessary test information on an ongoing basis.
- Identify the information gaps that affect testing.
- Resolve the gaps collaboratively with other team members.
- Decide when a test level is reached.
- Ensure appropriate tests executed at relevant times.

Functional and Non-Functional Test Design

In Agile projects, the traditional testing techniques can be used, but the focus is on early testing. Test cases need to be in place before the implementation starts.

For Functional test design, the testers and developers can use the traditional Black Box test design techniques such as:

- Equivalence Partitioning
- Boundary Value Analysis
- Decision Tables
- State Transition
- Class Tree

For non-functional test design, as the non-functional requirements are also a part of each user story, the black box test design techniques only can be used to design the relevant test cases.

Exploratory Testing

In Agile projects, time is often the limitation factor for Test Analysis and Test Design. In such cases, Exploratory testing techniques can be combined with the traditional testing techniques.

Exploratory Testing (ET) is defined as simultaneous learning, test design and test execution. In Exploratory Testing, the tester actively controls the design of the tests as they are performed and uses the information gained while testing to design new and better tests.

Exploratory Testing comes handy to accommodate changes in Agile projects.

Risk-Based Testing

Risk-based testing is testing based on the risk of failure and mitigates the risks using test design techniques.

A Product quality risk can be defined as a potential problem with product quality. Product quality risks include:

- Functional risks
- Non-functional performance risks
- Non-functional usability risks

Risk analysis is to be done to evaluate the probability (likelihood) and impact of each risk. Then, the risks are prioritized:

- High Risks require Extensive Testing
- Low Risks require only Cursory Testing

Tests are designed using appropriate Test Techniques based on the Risk Level and Risk Characteristic of each Risk. Tests are then executed to mitigate the Risks.

Fit Tests

Fit Tests are automated Acceptance Tests. The Tools Fit and FitNesse can be used for automating acceptance tests.

FIT uses JUnit, but extends the testing functionality. HTML tables are used to display the Test cases. Fixture is a Java class behind the HTML table. The fixture takes the contents of the HTML table and runs the test cases on the project being tested.

10. Agile Testing – Workproducts

Test Plan is prepared at the time of Release Planning and is revised at every Sprint Planning. Test Plan acts as a guide to the testing process in order to have the complete test coverage.

Typical Contents of a Test Plan are-

- Test Strategy
- Test Environment
- Test Coverage
- Scope of Testing
- Test Effort and Schedule
- Testing Tools

In Agile Projects, all the Team Members are accountable for the quality of the product. Hence, everyone participates in test planning as well.

A testers' responsibility is to provide necessary direction and mentor the rest of the team with their testing expertise.

User Stories

User Stories are not testing work products in principle. However, in Agile Projects, the testers participate in the User Stories Creation. Testers write User Stories that bring value to the customer and cover different possible behaviors of the system.

Testers also ensure that all the User Stories are testable and ensure the Acceptance Criteria.

Manual and Automated Tests

During the first run of Testing, Manual Tests are used. They include-

- Unit Tests
- Integration Tests
- Functional Tests
- Non-Functional Tests
- Acceptance Tests

The Tests are then automated for subsequent runs.

In **Test Driven Development**, Unit Tests are written first to fail, Code is developed and tested to ensure the Tests pass.

In **Acceptance Test Driven Development**, Acceptance Tests are written first to fail, Code is developed and tested to ensure the Tests pass.

In other Development methods, the Testers collaborate with the rest of the Team to ensure Test Coverage.

In all the types of methods, Continuous integration takes place, which includes continuous integration testing.

The team can decide when and what tests are to be automated. Even if automation of the tests requires effort and time, the resulting automated tests significantly reduce the repetitive testing effort and time during the iterations of the Agile Project. This in turn facilitates the team to pay more attention to the other required activities, such as new User Stories, Changes, etc.

In **Scrum**, the iterations are time-boxed. Hence, if a User Story testing cannot be completed in a particular Sprint, the tester can report in the daily standup meeting that the user story cannot reach the Done Status within that Sprint and hence needs to be kept pending to the next Sprint.

Test Results

As most of the Testing in Agile Projects is automated, the Tools generate the necessary Test Results Logs. Testers review the Test Results Logs. The test results need to be maintained for each sprint / release.

A Test Summary can also be prepared that contains-

- Testing Scope (What was tested and what was not tested)
- Defect Analysis along with Root Cause Analysis if possible
- Regression Testing Status after Defect Fixes
- Issues and the corresponding Resolution
- Pending Issues, if any
- Any modifications required in Test Strategy
- Test Metrics

Test Metrics Reports

In Agile Projects, the Test Metrics include the following for each Sprint-

- Test Effort
- Test Estimation Accuracy
- Test Coverage
- Automated Test Coverage
- No. of Defects
- Defect Rate (No. of Defects per User Story Point)
- Defect Severity

- Time to Fix a Defect in the same Sprint (It costs 24x as much to fix a bug that escapes the current sprint)
- No. of Defects fixed in the same Sprint
- Completion of Acceptance Testing by Customer within the Sprint

Sprint Review and Retrospective Reports

Testers also contribute to the Sprint Review and Retrospective Reports. The typical contents are-

- Test Metrics
- Test Result Logs review results
- What went right and what can be improved from Testing Point of View
- Best Practices
- Lessons Learned
- Issues
- Customer Feedback

11. Agile Testing in Kanban

Agile Testing activities can be managed effectively using Kanban concepts. The following ensure testing to be completed in time within an iteration / sprint and thus focus on the delivery of quality product.

- User Stories that are testable and effectively sized result in development and testing within the specified time limits.
- WIP (Work-In-Progress) limit allows to focus on a limited number of user stories at a time.
- Kanban board that represents the workflow visually, helps to track the testing activities and bottlenecks, if any.
- Kanban team collaboration concept lets resolution of bottlenecks as they are identified, without wait time.
- Preparation of Test Cases upfront, maintaining the test suite as the development progresses and obtaining Customer Feedback helps in eliminating Defects within the iteration / sprint.
- Definition of Done (DoD) is said to be Done-Done in the sense that a Story reaches a completion state only after the testing is also complete.

Testing Activities in Product Development

In Product development, the releases can be tracked with feature Kanban board. Features for a particular release are assigned to the Feature Kanban board that tracks the feature development status visually.

The Features in a release are broken into stories and developed within the release using agile approach.

The following Agile Testing activities ensure quality delivery in every release and at the end of all releases as well-

- Testers participate in User Story Creation and thus ensure-
    - All the possible Behaviors of the System are captured by means of User Stories and the Non-functional Requirements that are part of the User Stories.
    - User Stories are Testable.
    - Size of the User Stories allow Development and Testing to be complete (Done-Done) within the Iteration.

- Visual Task Kanban Board
    - Depicts the status and progress of the Tasks
    - Bottlenecks are identified immediately as they occur
    - Facilitates to measure the cycle time which can then be optimized
- Team Collaboration helps in
    - Accountability of the entire Team for Quality product
    - Resolution of bottlenecks as and when they occur, saving on wait time
    - Contribution of every expertise in all the activities
- Continuous Integration that focuses on Continuous Integration Testing
- Automation of Tests to save on Testing Effort and Time
- Defect Prevention with Test Cases written earlier to Development and mentoring the Developers on what is anticipated by different behaviors of the System
    - WIP Limit to focus on a limited number of User Stories at a Time
- Continuous Testing as the Development progresses, to ensure Defect Fixes within the Iteration
    - Ensure Test Coverage
    - Keep the Open Defects Count Low

Story Exploration

Story Exploration is the communication within an Agile team to explore Story understanding when the product owner passes a story for acceptance for development.

The product owner comes up with the story based on the functionality expected by the system. The developers do more exploring on each story before they mark it ready for acceptance. Testers also participate in the communication from testing perspective to make it as testable as possible.

Finalization of the Story is based on constant and continuous communication among the Product Owner, Developers and Testers.

Estimation

Estimation happens in Release Planning and each Iteration Planning.

In Release Planning, the testers provide-

- Information on what testing activities are required
- Effort Estimation for the same

In Iteration planning, the testers contribute to deciding on what and how many stories can be included in an iteration. The decision depends on the Test Effort and Test Schedule Estimation. The Story Estimation reflects the test estimation as well.

In Kanban, Done-Done is accomplished only when a story is developed and tested and marked as complete without defects.

Hence, Test Estimation plays a major Role in story estimation.

Story Planning

Story Planning begins after a Story has been estimated and assigned to current Iteration.

Story Planning includes the following test tasks-

- Prepare Test Data
- Extend Acceptance Tests
- Execute Manual Tests
- Conduct Exploratory Testing sessions
- Automate Continuous Integration Tests

In addition to these Testing Tasks, other tasks also may be required, such as-

- Performance Testing
- Regression Testing
- Updates of related Continuous Integration Tests

Story Progression

Story Progression uncovers additional tests that are required resulted by continuous communication between the developers and testers. In situations where the developers need more clarity on implementation, testers perform exploratory testing.

Continuous Testing is performed during Story Progression and includes Continuous Integration Testing. The entire team participates in the testing activities.

Story Acceptance

Story Acceptance occurs when the story reaches the Done-Done state. i.e., the story is developed and tested and signaled as complete.

Story testing is said to be completed when all the tests relevant to the story pass or level of test automation is met.

12. Agile Testing – Tools

In Agile Projects, Testers are responsible for the following daily tasks-

- Support the developers in coding, with clarifications on the expected behavior of the system.

- Help developers in creating effective and efficient unit tests.

- Develop automation scripts.

- Integrate automation testing tools / scripts with continuous integration for regression testing.

For an effective and fast implementation of these tasks, a Continuous Integration (CI) system that supports CI of Code and test components is used in most of the Agile projects.

The testers and the developers in agile projects can benefit from various tools to manage testing sessions and to create and submit Defect reports. In addition to specialized tools for agile testing, agile teams can also benefit from test automation and test management tools.

Note: Record-and-Playback, Test-Last, Heavyweight, and Test Automation Solutions are not Agile as-

- The test-last workflow encouraged by such tools does not work for Agile teams.

- The unmaintainable scripts created with such tools become an impediment to change

- Such specialized tools create a need for Test automation specialists and thus foster silos

The Tools that are widely used are-

Tool	Purpose
Hudson	CI Framework
Selenium	Functional Testing – Integrated with Hudson
CruiseControl	CI Framework
Junit	Java Unit Test
Nunit	.Net Unit Test
Cobertura / JavaCodeCoverage / JFeature / JCover /	Java Test Coverage

Tool	Purpose
Jester	Java - Mutation Testing/ Automated Error Seeding
Gretel	Java Test Coverage Monitoring Tool
TestCocoon	C/C++ or C# - reduces the amount of Tests by finding redundant Tests and finds Dead Code
JAZZ	Java - Branch, Node, and Defuse Coverage and implements a GUI, Test Planners, Dynamic Instrumentation, and a Test Analyzer
Ant	Java – Automation Build
Nant	.Net - Automation Build
Bonfire	Agile Testing add-on for JIRA

Agile Test Automation Tools

Effective Agile test automation tools support-

- Early test automation using a test-first approach.
- Writing test automation code using real languages, domain specific languages.
- Focusing on the expected behavior of the system.
- Separating the essence of the Test from the implementation details, thus making it Technology independent.
- Fostering Collaboration.

Automated Unit Tests (using Junit or NUnit) support test-first approach for coding. These are white-box tests and ensure that the design is sound, and that there are no defects. Such tests are built by developers with support from testers, and can be independent of the functionality that is required. This results in delivering a product that may not meet customer requirements and hence with no business value.

This concern is addressed by automating Acceptance Tests that are written with collaboration of customer, other stakeholders, testers and developers. The automated Acceptance Tests are written by the customers or product owners/business analysts reflecting the expected behavior of the product. The developers' involvement ensures the production of code as per the requirements. However, if the testing is focused only on acceptance, the resulting code may remain non-extensible.

Thus, Automated Unit Tests and Automated Acceptance Tests are complimentary and both are needed in Agile Development.

Agile Tools and Frameworks that support Automated Acceptance Testing are-

- Fit
- Fitnesse
- Concordion
- Ruby
- Cucumber

Fit

Ward Cunningham developed the tool Fit that can be used for Acceptance Test Automation. Fit allows-

- Customers or Product Owners to give examples of product behavior using Microsoft Word and Microsoft Excel
- Programmers to easily turn those examples into automated tests.

Fit 1.1 supports both Java and .NET.

FitNesse

FitNesse is a wiki, which is a style of web server that allows any visitor to make any edits, including changing existing pages and creating new pages. A simple markup language lets you easily create headings, make text bold, underline, and italic, create bulleted lists, and do other kinds of simple formatting.

In FitNesse, Acceptance Test Automation is as follows-

- Express tests as tables of input data and expected output data.
- Use FitNesse to put the test table on the page that you can edit.
  - Alternatively, put the test table in Microsoft Excel, copy to clipboard and then use the **Spreadsheet to FitNesse** command to have FitNesse format your table properly
- Run the test
- You get the test results by color coding of the cells in the test table
  - green cells represent that the expected values are obtained
  - red cells represent that a different value than what you expected is obtained
  - yellow cells represent that an exception was thrown

Cucumber

Cucumber is a tool based on Behavior Driven Development (BDD) framework. The key features are-

- Is used to write acceptance tests for web applications.
- Allows automation of functional validation in easily readable and understandable format like plain English.
- Was implemented in Ruby and then extended to Java framework. Both support Junit.
- Supports other languages like Perl, PHP, Python, .Net etc.
- Can be used along with Selenium, Watir, Capybara, etc.