## Manifesto for Agile Software Development

## *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.*

## 

## 

## Principles of Agile Testing

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**

**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, the testing team is the only team who is responsible for a testing process in the [Software Development Life Cycle](https://www.javatpoint.com/software-development-life-cycle). But on the other hand, in agile testing, the [business analysts (BA)](https://www.javatpoint.com/business-analyst) 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 reduce 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.

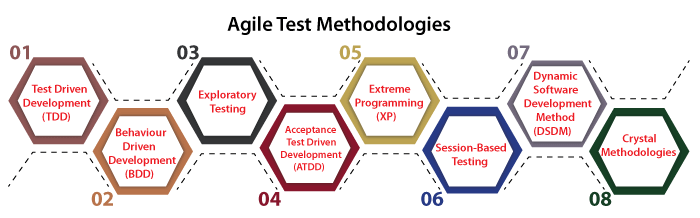
If we deliver the software quickly with the best of the attributes, and customer satisfaction is the primary concern at some stage in the agile testing process.

## 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 the 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.

### 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 understand 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](https://www.javatpoint.com/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

For more information related to the exploratory testing, refers to the following link: <https://www.javatpoint.com/exploratory-testing>.

### 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 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 correlated 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 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.

## What is agile?

[Agile](https://www.atlassian.com/agile) is a project management philosophy that employs a set of principles and values to help software teams respond to change. Agile teams value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. These values were set down in the [Agile Manifesto](https://www.atlassian.com/agile/manifesto) along with [12 principles](https://agilemanifesto.org/principles.html) behind the manifesto.

A good way to understand agile is to contrast it with another project management philosophy, waterfall. In waterfall delivery, the scope of a product is set, time and resources are flexible. Waterfall organizations will add more programmers and pad schedules to deliver the product that they decided to ship.

In agile, the scope of the product is flexible while resources and time are fixed. Agile teams commit to deliver software on time with the team they have today. What they deliver is a flexible combination of what they’ve learned that the customer wants and what they can create in the allotted time.

### Benefits of using agile

Agile teams have a strong “why” behind what they do and clarity about how they do it. The agile principles help teams break huge, ambitious goals into manageable chunks of work that they can consistently deliver. Agile software developers are empowered by countless stories of small, agile teams outperforming large competitors that use waterfall delivery. Agile teams also benefit from the “agile industrial complex.” There is a wealth of resources and tools for those needing to learn agile and a whole army of consultants eager to help with implementation.

### Disadvantages of using agile

Following agile principles can lead you to places you never thought you’d go. Agile helps teams change direction based on market and customer feedback. Chasing these ideals, you might find that your team has built something completely different than what you set out to do. This can be an unnerving feeling and you might even feel a lack of direction as you pursue new avenues and follow customer feedback in new directions. Because of these divergent outcomes, not all teams and companies can work in an agile way. But the teams that choose to overcome these hurdles often find they can ship a better product to their customers in the end.

## What is scrum?

[Scrum](https://www.atlassian.com/agile/scrum) is an agile framework that helps teams structure their work into short development cycles called [sprints](https://www.atlassian.com/agile/scrum/sprints). Scrum teams commit to shipping work at the end of each sprint and adopt practices and a team structure that helps them achieve this cadence. Scrum takes the agile principles one step further, creating structure that helps teams live the agile principles in their day-to-day work. Scrum is a well-documented agile framework that many teams can adopt without much disruption.

### Benefits of using the scrum methodology

Scrum teams ship software on time. Rather than updating the business on your progress, you can show them! When you ship software, customers start using it. More customer usage data helps inform your direction and drives growth. Scrum teams also tend to be healthier, with less burnout and churn than others. This is because scrum practices, like sprint planning and sprint retrospectives, are focused on setting up teammates for success.

### Disadvantages of using the scrum methodology

Scrum is an “all-in” approach. Success stems from the addition of new roles, like a [scrum master](https://www.atlassian.com/agile/scrum/scrum-master), and the refactoring of everybody's schedules around a set meeting cadence. Many teams don’t have the resources to hire new teammates and the time for new meetings. When teams fail to go “all-in” they often fail to unlock the benefits of scrum. Additionally, not all teams can deliver work at such a high cadence. When quality suffers as a result, many teams make their sprints longer and longer. Eventually, you’re back to doing waterfall!

## Other methodologies: Kanban and Waterfall

### What is kanban?

[Kanban](https://www.atlassian.com/agile/kanban) is an agile framework that helps teams deliver work on a continuous basis. Kanban teams organize their work on a kanban board with cards, columns, WIP limits, and specific commitment and delivery points. Kanban is best for knowledge work, where the product or service is largely invisible. Kanban helps teams visualize their and make strides day after day.

### What is waterfall?

Waterfall delivery is focused on the development of products or solutions based on specifications from the client or business. Teams study the requirements and build the solution over weeks, months, or even years. Waterfall is the preferred method in regulated industries where tolerances are very narrow.

Imagine you are making a surgery robot that needs to perform a task flawlessly for a government-mandated 100 hours of operation. That constraint inspires your work and that specification becomes the focus of your development. Your team experiments and tests until your robot meets the specifications. When specifications are specific and stringent, waterfall development focuses your team on meeting the requirements above all else.

## What is the best methodology for your team?

If you’re excited to begin an [agile transformation](https://www.atlassian.com/agile/advantage) you may need to choose a methodology. Agile methodologies include the team structure, practices, and tools you’d need for your org to live the agile principles. You can also step out on your own. With the agile manifesto and some creativity, you can design your own approach that works for your business and your team.

### Agile vs Scrum

Agile doesn’t have any set rules, while scrum has quite a few! If you are seeking a framework that can guide you on your way to more agility, choosing scrum is a strong start. Scrum will help energize your team to deliver work quickly and to pivot when needed. Additionally, there are [templates you can adopt today](https://www.atlassian.com/software/jira/templates/scrum) to supercharge your scrum adoption. If you are seeking the ultimate flexibility, you can instead inspire your team to go agile. An agile transformation is the thrilling process of breaking down what you do now and building up an agile way of working.

### Agile vs waterfall

It’s uncommon that you’d find yourself needing to choose between agile and waterfall. It’s more common that you’re needing to pivot from one to the other. In moments like these, the customer is key. Is the customer more solution focused or problem focused? If a customer knows what they want and wants to pay someone to build it, you can lean towards waterfall. If the customer is experiencing a problem and you want to be the one to solve it, agile all the way.

## **Agile Model Vs Waterfall Model**

Agile and Waterfall models are two different methods for software development process. Though they are different in their approach, both methods are useful at times, depending on the requirement and the type of the project.

| **Agile Model** | **Waterfall Model** |
| --- | --- |
| Agile methodology in software testing definition: Agile methodologies propose incremental and iterative approach to software design | Waterfall Model: Development of the software flows sequentially from start point to end point. |
| The **Agile process** in software testing is broken into individual models that designers work on | The design process is not broken into an individual models |
| The customer has early and frequent opportunities to look at the product and make decision and changes to the project | The customer can only see the product at the end of the project |
| Agile model in testing is considered unstructured compared to the waterfall model | Waterfall model are more secure because they are so plan oriented |
| Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time. | All sorts of projects can be estimated and completed. |
| Error can be fixed in the middle of the project. | Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning |
| Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less. | The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase. |
| Documentation attends less priority than [software development](https://www.guru99.com/software-development-tools.html) | Documentation is a top priority and can even use for training staff and upgrade the software with another team |
| Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released. | Only after the development phase, the testing phase is executed because separate parts are not fully functional. |
| In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers. | All features developed are delivered at once after the long implementation phase. |
| Testers and developers work together | Testers work separately from developers |
| At the end of every sprint, user acceptance is performed | User acceptance is **performed** at the end of the project. |
| It requires close communication with developers and together analyze requirements and planning | Developer is not involved in the requirement and planning process. Usually, time delays between tests and coding |

3 amigos - BA DEV QA OPS UI/UX

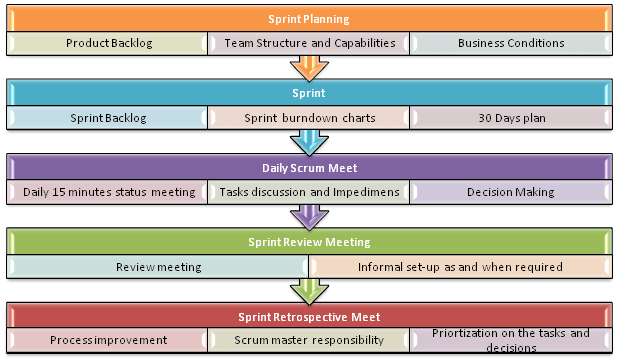
## **Product Backlog**

This is a repository where requirements are tracked with details on the number of requirements(user stories) to be completed for each release. It should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion

## 

## **Scrum Practices**

Practices are described in detailed:

Scrum Practices

## **Process flow of Scrum Methodologies:**

Process flow of [scrum testing](https://www.guru99.com/scrum-testing-beginner-guide.html) is as follows:

* Each iteration of a scrum is known as Sprint
* Product backlog is a list where all details are entered to get the end-product
* During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
* Team works on the defined sprint backlog
* Team checks for the daily work
* At the end of the sprint, team delivers product functionality

## **Lean Software Development**

Lean software development method is based on the principle “Just in time production”. It aims at increasing speed of software development and decreasing cost. Lean development can be summarized in seven steps.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

## **Kanban**

[Kanban](https://www.guru99.com/kanban-cards-boards-methodology.html) originally emerged from Japanese word that means, a card containing all the information needed to be done on the product at each stage along its path to completion. This framework or method is quite adopted in software testing method especially in Agile concepts.

## **Scrum Vs Kanban**

| **Scrum** | **Kanban** |
| --- | --- |
| In scrum technique, test must be broken down so that they can be completed within one sprint | No particular item size is prescribed |
| Prescribes a prioritized product backlog | Prioritization is optional |
| Scrum team commits to a particular amount of work for the iteration | Commitment is optional |
| Burndown chart is prescribed | No particular item size is prescribed |
| Between each sprint, a scrum board is reset | A Kanban board is persistent. It limits the number of items in workflow state |
| It cannot add items to ongoing iteration | It can add items whenever capacity is available |
| WIP limited indirectly | WIP limited directly |
| Timeboxed iterations prescribed | Timeboxed iterations optional |

**Also Check:-** [Kanban Vs. Scrum: What’s the Difference?](https://www.guru99.com/scrum-vs-kanban.html)

## **Agile metrics**

Metrics that can be collected for effective usage of Agile is:

* Drag Factor
  + Effort in hours which do not contribute to sprint goal
  + Drag factor can be improved by reducing number of shared resources, reducing the amount of non-contributing work
  + New estimates can be increased by percentage of drag factor -New estimate = (Old estimate+drag factor)
* Velocity
  + Amount of backlog(user stories) converted to shippable functionality of sprint
* No of Unit Tests added
* Time interval taken to complete daily build
* Bugs detected in an iteration or in previous iterations
* Production defect leakage

## What is scrum?

Scrum is an [agile project management](https://www.atlassian.com/agile/project-management) framework that helps teams structure and manage their work through a set of values, principles, and practices. Much like a rugby team (where it gets its name) training for the big game, scrum encourages teams to learn through experiences, self-organize while working on a problem, and reflect on their wins and losses to continuously improve.

While the scrum I’m talking about is most frequently used by software development teams, its principles and lessons can be applied to all kinds of teamwork. This is one of the reasons scrum is so popular. Often thought of as an agile project management framework, scrum describes a set of meetings, tools, and roles that work in concert to help teams structure and manage their work.

Scrum is an Agile project management framework that helps teams work together efficiently and in an organized way. It's divided into five phases:

* Initiation
* Planning and estimates
* Implementation
* Review and retrospective
* Release phase

The process flow of Scrum includes a set of repeatable steps, including: Sprint planning, Daily scrum, Sprint review, Sprint retrospective, Backlog refinement.

**Here are some details about each of these steps:**

* **Sprint planning**
* The team selects items from the product backlog to complete in one sprint and defines the sprint goal. They also estimate the effort required for each item and create a detailed plan to accomplish the work.
* **Sprint retrospective**
* The scrum team reflects on the last sprint and determines what went well and wrong. They also gather stakeholder and customer feedback to prioritize user stories and improve product performance. The sprint retrospective marks the end of the sprint, and a new sprint immediately begins.
* **Sprint backlog**
* A subset of the entire product backlog that the scrum team plans to implement in one iteration or sprint. The team provides all included estimates.
* **Burndown charts**
* A step in the Scrum workflow

## Exploring the origins of Kanban

Kanban is prominent among today's agile and DevOps software teams, but the Kanban methodology dates back more than 50 years. In the late 1940s, [Toyota began optimizing its engineering processes](https://mag.toyota.co.uk/kanban-toyota-production-system/) based on the same model supermarkets used to stock their shelves.

Supermarkets stock just enough inventory to meet consumer demand, a practice that optimizes the flow between the supermarket and the consumer. Because inventory levels match consumption patterns, the supermarket gains significant efficiency in inventory management by decreasing the excess inventory it must hold at any given time. Meanwhile, the supermarket can still ensure that essential products are always in stock.

When Toyota applied this same system to its factory floors, the goal was to better align its massive inventory levels with the actual consumption of materials. To communicate capacity levels in real-time on the factory floor (and to suppliers), workers would pass a card, or "kanban," between teams.

[Sprints](https://www.atlassian.com/agile/scrum/sprints)

[Sprint planning](https://www.atlassian.com/agile/scrum/sprint-planning)

[Ceremonies](https://www.atlassian.com/agile/scrum/ceremonies)

[Backlogs](https://www.atlassian.com/agile/scrum/backlogs)

[Sprint reviews](https://www.atlassian.com/agile/scrum/sprint-reviews)

[Standups](https://www.atlassian.com/agile/scrum/standups)

[Scrum master](https://www.atlassian.com/agile/scrum/scrum-master)

[Retrospectives](https://www.atlassian.com/agile/scrum/retrospectives)

[Distributed scrum](https://www.atlassian.com/agile/scrum/distributed-scrum)

[Roles](https://www.atlassian.com/agile/scrum/roles)

[Scrum of scrums](https://www.atlassian.com/agile/scrum/scrum-of-scrums)

[Agile scrum artifacts](https://www.atlassian.com/agile/scrum/artifacts)

[Scrum metrics](https://www.atlassian.com/agile/scrum/scrum-metrics)

In agile, an epic is a large body of work that can be broken down into smaller tasks called stories. A user story is a short request or requirement written from the perspective of an end user. A task is a specific piece of technical work for an agile team.

**Here's some more information about epics, user stories, and tasks:**

* Epics
* Epics are broad in scope and are often considered the "top tier" of work hierarchies. They can contain several user stories and describe major product requirements or areas of functionality. Epics are too big to fit into a sprint and can take up to a quarter of a calendar year to complete.
* User stories
* Also called "stories", user stories are short requirements or requests written from the perspective of an end user. User stories define a discrete product function or benefit that creates new value for customers. Ideally, you should be able to complete a user story within a single sprint.
* Tasks
* Tasks are specific pieces of technical work for an agile team that represent items of development work that must be done to complete a user story. Tasks are broken-down sections of a story that address "how" the story will be finished. Examples of tasks include creating a new database table, creating a user interface, writing an API, and performing functional testing.