Manuel Difference

Parameter	SDLC	STLC
Origin	Development Life Cycle	Testing Life Cycle
Objective	The main object of SDLC life cycle is to complete successful development of the software including testing and other phases.	The only objective of the STLC phase is testing.
Requirement Gathering	In SDLC the business analyst gathers the requirements and create Development Plan	In STLC, the QA team analyse requirement documents like functional and non-functional documents and create System Test Plan
High & Low-Level Design	In SDLC, the development team creates the high and low-level design plans	In STLC, the test analyst creates the Integration Test Plan
Coding	The real code is developed, and actual work takes place as per the design documents.	The testing team prepares the test environment and executes them
Maintenance	SDLC phase also includes post-deployment supports and updates.	Testers, execute regression suits, usually automation scripts to check maintenance code deployed.

Verification	Validation
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does *not* include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.
It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and	The goal of validation is an actual product.

Verification	Validation
software architecture and specification.	
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.
It consists of checking of documents/files and is performed by human.	It consists of execution of program and is performed by computer.

Smoke and sanity

- Smoke Testing has a goal to verify “stability” whereas Sanity Testing has a goal to verify “rationality”.
- Smoke Testing is done by both developers and testers whereas Sanity Testing is done by testers.
- Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
- Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
- Smoke testing is documented or scripted whereas Sanity testing isn’t.
- Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component.

Smoke Testing	**Sanity Testing**
This testing is performed by the developers or testers	Sanity testing in software testing is usually performed by testers
Smoke testing is usually documented or scripted	Sanity testing is usually not documented and is unscripted
Smoke testing is a subset of Acceptance testing	Sanity testing is a subset of Regression Testing
Smoke testing exercises the entire system from end to end	Sanity testing exercises only the particular component of the entire system
Smoke testing is like General Health Check Up	Sanity Testing is like specialized health check up

Regression Testing	**Re-testing**
The purpose is that new code changes should not have any side effects to existing functionalities	Re-testing is done on the basis of the Defect fixes
Defect verification is not the part of Regression Testing	Defect verification is the part of re-testing
You can do automation for regression testing,	You cannot automate the test cases for Retesting
Regression testing is known as a generic testing	Re-testing is a planned testing
Regression testing is done for passed test cases	Retesting is done only for failed test cases
Regression testing checks for unexpected side-effects	Re-testing makes sure that the original fault has been corrected

Parameters	**Functional**	**Non-functional testing**
Execution	It is performed before non-functional testing.	It is performed after the functional testing.
Focus area	It is based on customer's requirements.	It focusses on customer's expectation.
Requirement	It is easy to define functional requirements.	It is difficult to define the requirements for non-functional testing.
Usage	Helps to validate the behavior of the application.	Helps to validate the performance of the application.
Objective	Carried out to validate software actions.	It is done to validate the performance of the software.
Requirements	Functional testing is carried out using the	This kind of testing is carried out by

	functional specification.	performance specifications
Manual testing	Functional testing is easy to execute by manual testing.	It's very hard to perform non-functional testing manually.
Functionality	It describes what the product does.	It describes how the product works.
Example Test Case	Check login functionality.	The dashboard should load in 2 seconds.
Testing Types	Examples of Functional Testing Types • Unit testing • Smoke testing • User Acceptance • Integration Testing • Regression testing • Localization • Globalization • Interoperability	Examples of Non-functional Testing Types • Performance Testing • Volume Testing • Scalability • Usability Testing • Load Testing • Stress Testing • Compliance Testing • Portability Testing • Disaster Recover Testing

-
- **Functional and Non functional**
- Functional testing verifies each function/feature of the software whereas Non Functional testing verifies non-functional aspects like performance, usability, reliability, etc.
- Functional testing can be done manually whereas Non Functional testing is hard to perform manually.
- Functional testing is based on customer's requirements whereas Non Functional testing is based on customer's expectations.
- Functional testing has a goal to validate software actions whereas Non Functional testing has a goal to validate the performance of the software.
- A Functional Testing example is to check the login functionality whereas a Non Functional testing example is to check the dashboard should load in 2 seconds.
- Functional describes what the product does whereas Non Functional describes how the product works.
- Functional testing is performed before the non-functional testing.

Agile Testing	Waterfall Testing

Agile Testing	Waterfall Testing
In agile testing, testing is performed alongside the development.	In waterfall testing, testing is carried out only after the completion of development.
In agile testing, development team and testing team work together.	In waterfall testing, development team and testing team work separately.
In agile testing, testers are involved in the requirements.	In waterfall testing, testers may or may not be involve in the requirements..
In agile testing, acceptance testing is carried out after every iteration.	In waterfall testing, acceptance testing is carried out only in the end.
In agile testing, regression testing is carried out after every iteration.	In waterfall testing, regression testing is carried out only in the end.
In agile testing, there is no time delays between coding and testing.	In waterfall testing, there is normal time delays between coding and testing.
In agile testing, different testing levels overlap.	In waterfall testing, testing levels can't overlap.

<u>Load Testing</u>**:**
Load Testing is a type of performance testing which determines the performance of a system, software product or software application under real life based load conditions.
<u>Stress Testing</u>**:**
Stress testing is a type of software testing that verifies the stability and reliability of the system. This test particularly determines the system on its robustness and error handling under extremely heavy load conditions.
Difference between Load Testing and Stress Testing:

Load Testing	Stress Testing
Load Testing is performed to test the performance of the system or software application under extreme load.	Stress Testing is performed to test the robustness of the system or software application under extreme load.
In load testing load limit is the threshold of a break.	In stress testing load limit is above the threshold of a break.
In load testing, the performance of the software is tested under multiple	In stress testing, the performance is tested under varying data amounts.

Load Testing	Stress Testing
number of users.	
Huge number of users.	Too much users and too much data.
Load testing is performed to find out the upper limit of the system or application.	Stress testing is performed to find the behavior of the system under pressure.
The factor tested during load testing is *performance*.	The factor tested during stress testing is *robustness* and *stability*.
Load testing determines the operating capacity of a system or application.	Stress testing ensures the system security.

Scalability Testing is a **type of non-functional testing** in which the performance of a software application, system, network or process is tested in terms of its capability to scale up or scale down the number of user request load or other such performance attributes.

Defect Density

Defect density is the number of defects found in the software product per size of the code. Defect Density' metrics is different from the 'Count of Defects' metrics as the latter does not provide management information.
Defect density metric not only indicates the quality of the product being developed, but it can also be used as a basis for estimating a number of defects in the next iteration or sprint. It can be defined as the number of defects per 1,000 lines of code or function points.

Defect Density = Total Number of defects/Total lines of code

Defect density has its own pros and cons. A QA manager needs to thoroughly understand these metrics before using it as a benchmark. It is recommended to use a tool to calculate the defect density else it might become labour intensive.
It is useful in comparing similar projects. However, this metric can be misleading if the complexity of the code is not considered, as different parts of the code have a different degree of complexity.
An overall reduction in the defect density indicates a better quality of the product being developed, i.e. there are fewer bugs in the product under test.

Bug is included by default, **defect** must be a custom issue type. When I've used those, a **Bug** was a standard issue type, and represented **Bug** Reports. A **Defect** was a subtask type and was used when the manual QA team found a **bug in a** story or feature. We can only guess why it's implemented in your system.

"A mistake in coding is called Error, error found by tester is called **Defect**, **defect** accepted by development team then it is called **Bug**, build does not meet the requirements then it Is Failure." In other words **Defect** is the **difference between** expected and actual result **in the** context of testing.

Defect Leakage is the metric which is used to identify the efficiency of the QA testing i.e., how many defects are missed/slipped during the QA testing.
Defect Leakage = (No. of Defects found in UAT / No. of Defects found in QA testing.)

Bug release is **when software or an application is handed over to the testing team knowing that the defect is present in a release**. ... Bug leakage is something, when the bug is discovered by the end users or customer, and not detected by the testing team while testing the software.

A burn down chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burn down charts are used to predict your team's likelihood of completing their work in the time available. ... Burn down charts are useful because they provide insight into how the team works.
A burn up chart is a visual diagram commonly used on agile projects to help measure progress. Agile burn up charts allow project managers and teams to quickly see how their workload is progressing and whether project completion is on schedule.

Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testing.
Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the type of User Acceptance Testing.
The difference between Alpha and Beta Testing is as follow:

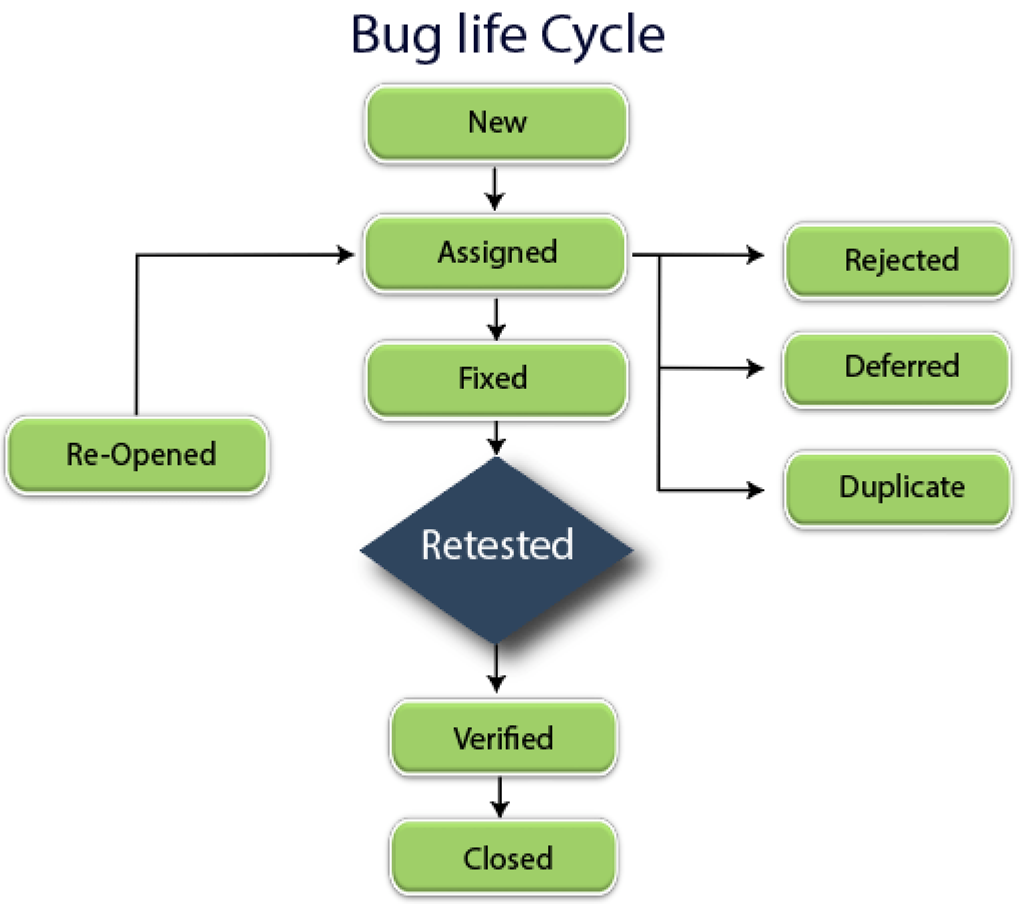
Alpha Testing	Beta Testing
Alpha testing involves both the white box and black box testing.	Beta testing commonly uses black box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.

Alpha Testing	Beta Testing
Alpha testing is performed at developer’s site.	Beta testing is performed at end-user of the product.
Reliability and security testing are not checked in alpha testing.	Reliability, security and robustness are checked during beta testing.
Alpha testing ensures the quality of the product before forwarding to beta testing.	Beta testing also concentrates on the quality of the product but collects users input on the product and ensures that the product is ready for real time users.
Alpha testing requires a testing environment or a lab.	Beta testing doesn’t require a testing environment or lab.
Alpha testing may require long execution cycle.	Beta testing requires only a few weeks of execution.
Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from beta testing will be implemented in future versions of the product.

Bug Life Cycle:

- **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
- **Assigned:** Once the bug is posted by the we tester, it assigns the bug to the developer team
- **Open**: The developer starts analysing and works on the defect fix
- **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."
- **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test."
- **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."
- **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
- **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
- **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."

- **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
- **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to "duplicate."
- **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".



What is RDBMS

Relational Database Management System (RDBMS) is an advanced version of a DBMS system.RDBMS is a software system which is used to store only data which need to be stored in the form of tables. In this kind of system, data is managed and stored in rows and columns which is known as tuples and attributes.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

- DBMS stores data as a file whereas in RDBMS, data is stored in the form of tables.
- DBMS supports single users, while RDBMS supports multiple users.
- DBMS does not support client-server architecture but RDBMS supports client-server architecture.
- DBMS has low software and hardware requirements whereas RDBMS has higher hardware and software requirements.

- In DBMS, data redundancy is common while in RDBMS, keys and indexes do not allow data redundancy.

The components of a test case include:

- **Test name.** A title that describes the functionality or feature that the test is verifying.
- **Test ID.** Typically a numeric or alphanumeric identifier that QA engineers and testers use to group test cases into test suites.
- **Objective.** Also called the description, this important component describes what the test intends to verify in one to two sentences.
- **References.** Links to user stories, design specifications or requirements that the test is expected to verify.
- **Prerequisites.** Any conditions that are necessary for the tester or QA engineer to perform the test.
- **Test setup.** This component identifies what the test case needs to run correctly, such as app version, operation system, date and time requirements and security specifications.
- **Test steps.** Detailed descriptions of the sequential actions that must be taken to complete the test.
- **Expected results.** An outline of how the system should respond to each test step.

Build

It is software, which is used to convert the code into application format. And it consists of some set of features and bug fixes that are handed over to the test engineer for testing purposes until it becomes stable.

Test design techniques

- Decision table testing
- All-pairs testing
- Equivalence partitioning
- Boundary value analysis
- Cause–effect graph
- Error guessing
- State transition testing
- Use case testing
- User story testing
- Domain analysis

- Syntax testing
- Combining technique

1. **Acceptance Criteria**
Acceptance criteria are a **set of conditions** that software must meet in order to be **accepted by a customer** or stakeholder.
2**. Acceptance Test**
An acceptance test ensures that a software feature is **working correctly** and meets the acceptance criteria. It's usually run **after** the software has been developed.
3. **Agile manifesto**
The Agile Manifesto is a **document** that sums up the **12 agile principles** that guide the agile framework.
Every Agile methodology strictly follows the principles and practices outlined in the Agile Manifesto.
5. Agile Mindset
An Agile Mindset is a **set of attitudes** that an Agile or Scrum team should have towards their work.
These attitudes are inspired by Agile values and principles, such as:

- Respect
- Collaboration
- Continuous improvement
- Focus on delivering value

7. Agile Release Train
An Agile Release Train is a **combination** of multiple **agile software development teams** used to tackle large enterprise-scale projects.
8. Agile Software Development
Agile Software Development is a project management technique that allows developers to **create a working software model in just a few weeks.**
10. Backlog
It's a list of new product features, updates, bug fixes, etc. that are required by the user. v
At the start of every iteration, the product owner decides which backlog items the team needs to work on. After every iteration, the backlog is regularly updated with user suggestions and new features.
Product backlog, sprint backlog
11. Backlog Refinement
It's a **Scrum meeting** where the Scrum team **organizes the backlog** to make sure it's ready for the next sprint or iteration. *In other words, it's like spring cleaning… but for* Scrum *teams!*

It is normally held at the end of the sprint.

12. Bottleneck

Bottlenecks are issues that can completely **slow down** the development process.

15. Continuous Integration (CI)

Continuous integration is an **agile practice** where developers **constantly add their code** to the main system.

16. Daily Scrum

It's a **daily meeting** usually hosted by the Scrum master. Every morning, the Scrum team gets together for **15 minutes** to **discuss their day** ahead.

Each member briefly talks about the following topics:

- What they plan to do today
- What they did yesterday
- Issues they have encountered

17. DSDM (Dynamic systems development method)

It's a business-oriented agile framework that focuses on the entire project from start to finish. The core belief of DSDM is that the work done on the agile project should align with the strategy of the company.

18. Epics

An epic is a **big idea** or feature that can be **broken down into smaller user stories.** Much like how large 'epics' like Lord of the Rings are split into 3 books.

19. Gantt Chart

It's a **horizontal bar chart** that visualizes the **sequence of tasks** within the project timeline. Each task has a **start date** and **end date** so that your team doesn't exceed deadlines.

20. Impediment

It's an **obstacle** that **reduces** an Agile team's **productivity** or prevents them from completing an Agile project altogether.

21. Iteration

It's a **period of time** in which an Agile team needs to develop working software. An iteration generally lasts for around 2-4 weeks for Kanban and Scrum teams.

24. Lean

It's a **set of principles and practices** that optimizes the development process. It was inspired by the lean manufacturing approach introduced by Toyota in the 50s.

Learn more about Lean project management and Lean principles in our dedicated articles.

25. Product backlog

A product backlog is Scrum terminology that refers to a **list** of new features, updates, bug fixes, etc. that are required by the user. The product owner is in charge of **prioritizing items** in the product backlog. They decide every product backlog item the team needs to work on at the beginning of each iteration.

26. Product manager

A product manager **assists the agile team** with the development process from start to finish.

Their main responsibilities include:

- dealing with problems in the development process
- ensuring the team meets project deadlines
- collaborating with other departments of the company like sales, marketing and, customer service

27. Product owner
They are the key members **of an Agile or Scrum team.**
They **decide the vision** and features of the final software, but the features are not chosen on a whim!

28. Refactoring
Refactoring is an **extreme programming** practice.
Here agile software development teams 'clean up' the code by:

- Removing redundant pieces of code
- Edit out unnecessary functions

29. Release plan
A release plan showcases all the features to be included in the next release, along with an estimated release date.
30. **Scaled Agile Framework**
It's an agile methodology that allows **large companies** to implement Lean and Agile practices throughout the organization. SAFe **unites all software development teams** within a company to work towards developing
31. Scrum
Scrum is an **Agile methodology** in which a team works in **short bursts of work** ranging from 2-4 weeks, called sprints. At the end of the sprint, they **deliver the product** to the customers, and in turn, the customers give the developers their **feedback.**
What goes on in a Scrum sprint?
33. Scrum board
It's a virtual or physical board that displays tasks that need to be done in a sprint.
34. Scrum master
The Scrum master is the **leader** of the Scrum. They organize meetings, remove impediments, and work with the product owner to ensure that the product backlog is up to date.
35. Scrum meeting
A Scrum meeting is an **essential part** of the Scrum framework. Without them, the sprint would have **no structure** or project plan in place!
There are five types of Scrum meetings that occur during the sprint.
Each Scrum meeting enables the whole team to do important sprint tasks like:

- Create a plan for the sprint in a **sprint planning** meeting
- Solve problems together in the **Daily Scrum**
- Organize the backlog in the **backlog refinement** meeting
- Demonstrate a working software to the customer in a **sprint review**

- Analyze the sprint performance in a **sprint retrospective**

36. Scrum of Scrums
It’s a special Scrum meeting for large Scrum teams.
Here, large Agile teams (more than 12 team members) are divided into smaller Scrum teams (around 5-10 members). Each small Scrum team designates one member as an ‘ambassador’.
Every day, all the ambassadors meet in the Scrum of Scrums to update their progress and resolve their issues.
37. Scrum team
It’s a cross-functional team of 5-10 individuals with different skill sets ranging from graphic design, UX, coding, etc. They work together to develop a product under the Scrum framework.
Each Scrum team usually contains 3 key roles:

- Product owner
- Scrum master
- Developers

38. Sprint
It’s another term for an iteration.
However, sprint is a Scrum term and is usually a phrase used by Scrum teams.
Another difference between the two is that sprints maintain a uniform length (2-4 weeks) during the agile development process. Whereas, iterations can have varying lengths, depending on the nature of the work.
39. Sprint backlog
It’s a list of features, bug fixes, user requirements, and tasks that the Scrum team needs to work on during the sprint.
During the sprint planning meeting, the product owner decides the backlog items need to be added to the sprint backlog.
40. Sprint goal
The sprint goal is the desired result the Scrum team wants to achieve during the sprint. In most cases, the end result is a working model of the software that can be shown to the stakeholders.
41. Sprint planning
Sprint planning is a Scrum meeting where the Scrum team decides the work they need to do during the sprint.
This includes picking up items (like user stories) from the sprint backlog and breaking them down into smaller, more manageable tasks.
42. Sprint retrospective
A sprint retrospective is a Scrum meeting where the Scrum team analyses their performance, at the end of the sprint.
The team uses agile metrics, charts, and reports to see where they excel and where they need to improve.

Instead of manually tracking agile metrics, agile project management tools like Click Up offer accurate graphs and charts to help your team power through a sprint retrospective.

43. Sprint review

A sprint review is a **Scrum meeting** where the Scrum team **demonstrates a working software** model to the stakeholders.

Along with the product demo, the Scrum team prepares a presentation that outlines the new features added, bug fixes, and other changes.

Sprint review meeting

45. Story points

It's a **measure of effort** that your team would need to **complete project tasks** (user story).

How are they calculated?

46. Task board

It's a generic agile term that can refer to either a Kanban board or Scrum board.

47. User story

It's a **brief description** of a specific **product feature** or a function that customers would find helpful.

49. Velocity

It's a **unit of measurement** that determines the amount of work your team can handle during an iteration. It's measured by calculating the average number of tasks/user stories completed in a sprint. Agile teams use a **velocity chart** to do this.

50. XP (Extreme programming)

Extreme Programming is a **software development methodology** in which a team has to complete a working software in **1-2 weeks**, rather than the usual 2-4 week iteration.

Here all the conventional practices of the Scrum methodology are cranked up to an 11, so teams can quickly respond to the **customer's requirements.**

AGILE ARTIFACTS

Product backlog

The product backlog is a list of new features, enhancements, bug fixes, tasks, or work requirements needed to build a product.

Sprint backlog

The sprint backlog is a set of product backlog tasks that have been promoted to be developed during the next product increment. Sprint backlogs are created by the development teams to plan deliverables for future increments

Product increment

A product increment is the customer deliverables that were produced by completing product backlog tasks during a sprint. It also includes the increments of all previous sprints.

Extended artifacts

In addition to the previously discussed official scrum artifacts, there exist some extended or Meta artifacts. While not official per official scrum guidelines these extended artifacts add additional value and insight to a scrum cycle.

Burn down chart

The definition of "done"

This definition can be another type of artifact, which should be documented and shared.

Artifact transparency

Scrum artifacts are powerful aids that help teams operate more efficiently. Therefore, it's important all teams have access and visibility into the artifacts.

What is meant by Agile Manifesto?

The Agile Manifesto is **a document that sets out the key values and principles behind the agile philosophy and serves to help development teams work more efficiently and sustainably**. Known officially as 'The Manifesto for Agile Software Development', the manifesto detailing **4 Values** and **12 Principles**.

The key values and principles of agile

1.Individuals and interactions over processes and tools.

2. Working software over comprehensive documentation.

3. Customer collaboration over contract negotiation.

4. Responding to change over following a plan.

What is agile release notes?

A release note refers **to the technical documentation produced and distributed alongside the launch of a new software product or a product update** (e.g., recent changes, feature enhancements, or bug fixes). ... The primary target audience is the product user, but a release note can also be used internally.

Who prepares the release notes?

Release notes are generally prepared by **the Development Team**