http://www.gcreddy.com/2014/07/software-development-life-cycle.html

SDLC : Software Development Life Cycle…set of phases that will continue in a cycle to complete the process.

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

SDLC model can be classified as two types :

1. Sequential models
   1. Waterfall model
   2. V-model
2. Incremental Models
   1. Prototype model
   2. Spiral model
   3. Agile model

A typical Software Development life cycle consists of the following stages:

1. Requirement Gathering /Planning – business manager / project manager or owner

2. Analysis – product owner / BA

3. Design – architect / ui designer

4. Implementation/Develop/coding – developers—frontEnd/backend/application developers and DBA team

5. Testing – test engineers and team

6. Maintenance --

People involved in projects : in diff company designations would be different.

Project cycle : DEV-🡪SIT🡪UAT🡪Deploy

Deployment : building bundled together with all the jars and moving into server (Application package into server machine: build Engineer/release Engineer/deployment Engineer)

Stake Folder : a person/group/Organisation who is effected or who those influence directly or indirectly

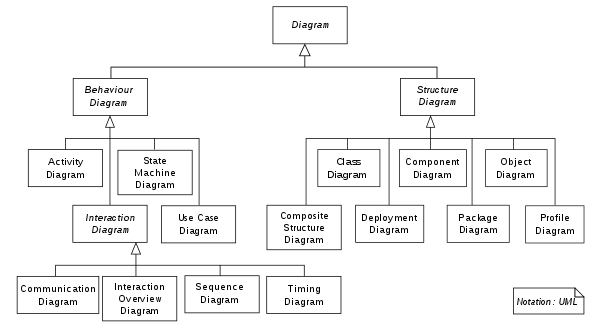
A person, group or organization that has interest or concern in an organization.  
Stakeholders can affect or be affected by the organization's actions, objectives and policies. Some examples of key stakeholders are creditors, directors, employees, government (and its agencies), owners (shareholders), suppliers, unions, and the community from which the business draws its resources.  
  
BRD : Business Requirements Documents : defines the requirements..highlevel documents contains scope, objective and minute details of the project.

SRS : System Requirement Document: describes requirements briefly : what software should go with the project.

Diagrams :

UML Diagrams: The **Unified Modeling Language** (**UML**) is a general-purpose, developmental, [modeling language](https://en.wikipedia.org/wiki/Modeling_language) in the field of [software engineering](https://en.wikipedia.org/wiki/Software_engineering), that is intended to provide a standard way to visualize the design of a system.[[1]](https://en.wikipedia.org/wiki/Unified_Modeling_Language#cite_note-:1-1)

, which are divided into two categories.[[4]](https://en.wikipedia.org/wiki/Unified_Modeling_Language" \l "cite_note-:0-4) Some types represent *structural* information, and the rest represent general types of *behavior*, including a few that represent different aspects of *interactions*. These diagrams can be categorized hierarchically as shown in the following class diagram:[[4]](https://en.wikipedia.org/wiki/Unified_Modeling_Language" \l "cite_note-:0-4)

[](https://en.wikipedia.org/wiki/File:UML_diagrams_overview.svg)

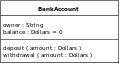
These diagrams may all contain comments or notes explaining usage, constraint, or intent.

**Structure diagrams**[[edit](https://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&action=edit&section=10" \o "Edit section: Structure diagrams)]

[Structure diagrams](https://en.wikipedia.org/wiki/Structure_diagram) emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the [software architecture](https://en.wikipedia.org/wiki/Software_architecture) of software systems. For example, the [component diagram](https://en.wikipedia.org/wiki/Component_diagram) describes how a software system is split up into components and shows the dependencies among these components.

* [](https://en.wikipedia.org/wiki/File:Policy_Admin_Component_Diagram.PNG)

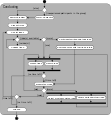
[Component diagram](https://en.wikipedia.org/wiki/Component_diagram)

* [](https://en.wikipedia.org/wiki/File:BankAccount1.svg)

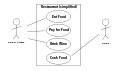
[Class diagram](https://en.wikipedia.org/wiki/Class_diagram)

**Behavior diagrams**[[edit](https://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&action=edit&section=11" \o "Edit section: Behavior diagrams)]

Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems. As an example, the [activity diagram](https://en.wikipedia.org/wiki/Activity_diagram) describes the business and operational step-by-step activities of the components in a system.

* [](https://en.wikipedia.org/wiki/File:Activity_conducting.svg)

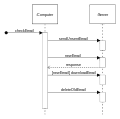
[Activity diagram](https://en.wikipedia.org/wiki/Activity_diagram)

* [](https://en.wikipedia.org/wiki/File:UML_Use_Case_diagram.svg)

[Use case diagram](https://en.wikipedia.org/wiki/Use_case_diagram)

**Interaction diagrams**[[edit](https://en.wikipedia.org/w/index.php?title=Unified_Modeling_Language&action=edit&section=12" \o "Edit section: Interaction diagrams)]

Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled. For example, the [sequence diagram](https://en.wikipedia.org/wiki/Sequence_diagram) shows how objects communicate with each other in terms of a sequence of messages.

* [](https://en.wikipedia.org/wiki/File:CheckEmail.svg)

[Sequence diagram](https://en.wikipedia.org/wiki/Sequence_diagram)

* [](https://en.wikipedia.org/wiki/File:UML_Communication_diagram.svg)

[Communication diagram](https://en.wikipedia.org/wiki/Communication_diagram)

Use case diagrams: Uniform defines about actors and their responsibilities or functionality according to actors.

Activity Diagrams: like flow charts. Every steps are broken down into flow depending upon the execution continues.

Sequence diagrams: depicted in form of a sequence diagram…step by step sequence.

Class diagrams: defining variables and methods/properties and action for given behavior. Are very technical.

Wireframes: pictorial representation of the requirements, position of the elements. For mobile testing we get the requirements documents are called wireframes.

**SDLC PHASES:**

**1.Requirement Gathering**: During this phase, the specific requirements of the software to be developed are gathered and documented. Requirements are gathered in different way for the software project and software product.

Business Analyst gathers the requirements for the software project and documented as the Business Required Document(BRD).

Business team gathers requirements from model customers and compares with the competitive model and prepares the Product Required Document(PRD or BRD).

This document is sent to Project manager for review and approval .and the final BRD document is the output and this is the input for next phase.

**2. Analysis and Planning:** During this phase Technical manager prepares the SRS document based on the BRD document and project manager prepares the project plan based on the analysis.

Technical manager analyzes the BRD based on following factors:

1. Are they complete requirements
2. Are they complete requirements
3. Are they achievable requirements
4. Are they testable requirements

After analyzing TM defines SRS document based which consist of two sub options:

FRS: Functional requirement specification: describes the functionalities of the project.

SRS: System Requirement Specification: describes the software and hardware to be used.

Planning: project manager create the project plan that explains how the requirements will be met and by which time. PM also analyzes the risks involved and contingency plan for them. Finally, he comes up the milestones ad the release dates for the project.

Output of this phase would be SRS and Project Plan.

1. **System Design**: During this phase , System architect or senior developer creates the software design document for the project based on the SRS. This divided into two categories:

HLD: High-level-design Document: global design that contains the overall architecture of the project.

LLD: Low-level-Design Document: detailed design that contains internal architecture related to each and every unit/module/component.

SSD will be used by the development team to produce the programs.

**4. Coding/Implementation**: during this phase based on the design document the developers will develop the software application according to the required standards ,In addition to coding it also involves the creation of the documentation of the developed software.

**5. Testing:** during this phase the testers test the developed software, based on the SRS document to check whether the developed application is met with requirements or not. If they found any defects they inform development team and the developers fixes the bug and sends back to the testing team , which is again checked and testers confirmed no defects found and closed the defects and it will be moved to the next phase.

The output of this phase is Tested software and test documents.

**6.Relese and Maintenance**: testes software is released to the customers. The released s/w to be deployed into the production environment, and is generally done by the release team. Customer starts using the s/w in their environments their they might phase problem which are not identified in testing phase, then it is called as maintenance phase. After that maintenance team would be handing the maintenance issues.

Maintenance can be sub divided into :

1. Correct changes: defect during production phase are required emergency fixes and these may not be identified in testing phase.
2. If customer wants some features to be added to the existing system then they are referred as Enhancements.
3. Migrating the software from the old technology to new technology.
4. Retirement: retiring the old system and developing the new system.

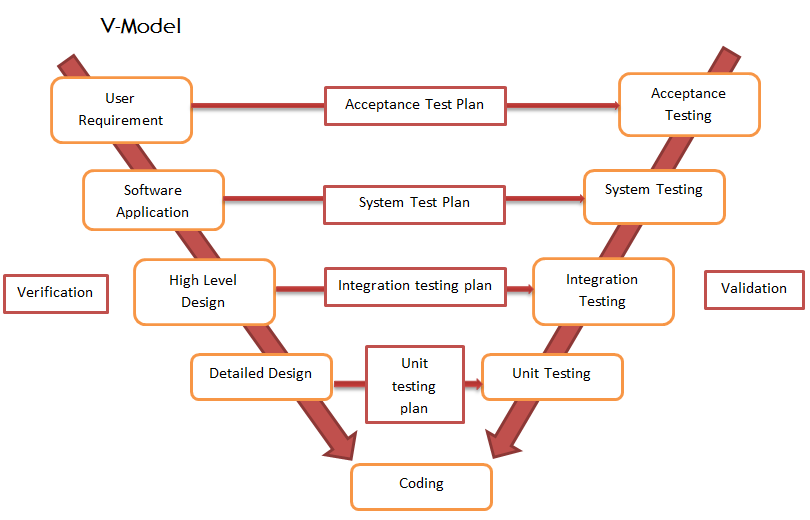
**Waterfall model:** in this model the process flow goes from top to down, step by step. Next phase of the project can only be started once the first phase is completed. The output of the first phase is the input for the next phase.

**Adv**: the model is simple and easy to understand

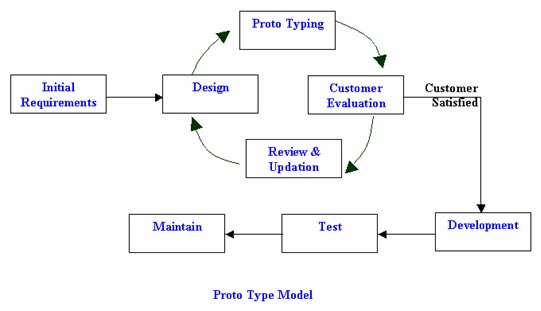
1. Requirements are well documented , clear and fixed.
2. Product definition is stable
3. Works well for the smaller projects
4. Phases are processed and completed at atime

**Dis Adv**:

V-Model:



Prototype Model: The Prototyping Model is a systems development method (SDM) in which a [prototype](http://searchcio-midmarket.techtarget.com/definition/prototype) (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed. This model works best in scenarios where not all of the project requirements are known in detail ahead of time. It is an iterative, trial-and-error process that takes place between the developers and the users.



Agile SDLC Model: is a combination of iterative and incremental models focusing on the process adaptability and customer satisfaction by rapid delivery of working software product(updating the status the project every time).

Agile method breaks the projects into small incremental builds. These builds are provided in iterations.( **Iteration** is the act of repeating a process, either to generate an unbounded sequence of outcomes, or with the aim of approaching a desired goal, target or result. Each repetition of the process is also called an "iteration", and the results of one iteration are used as the starting point for the next iteration.)

Each iteration lasts for one or three weeks of duration. Every iteration involves teams working simultaneously working on various area like planning, requirement analysis, design, coding, unit testing, acceptance testing.

At the end of the each iteration working project/product is displayed to the customer and important stake holders.

What is Agile: Agile is a principle or methodology. In agile the tasks are divided into small time frames to deliver specific feature for a release. Iterative approach is taken and working software build is released after each iteration. Each iteration is incremental in terms of features, the final build hold all the features required by the customer.



Different terms used in Agile:

**Scrum**: is an iterative and incremental agile software development framework for managing the product development. It defines the strategy where a development team works as a unit to reach a common goal.

Scrum Master: The Scrum Master is not a traditional [team lead](https://en.wikipedia.org/wiki/Team_leader) or [project manager](https://en.wikipedia.org/wiki/Project_manager), but acts as a buffer between the team and any distracting influences. The Scrum Master ensures that the Scrum framework is followed. The Scrum Master helps to ensure the team follows the agreed processes in the Scrum framework, often facilitates key sessions, and encourages the team to improve.

Scrum master assigns user stories to scrum team and monitor team and coordinates between the team and product owner.

The core responsibilities of a Scrum Master include (but are not limited to):

* Helping the Product Owner maintain the Product Backlog in a way that ensures the needed work is well understood so the team can continually make forward progress
* Helping the team to determine the *definition of done* for the product, with input from key stakeholders
* Coaching the team, within the Scrum principles, in order to deliver high-quality features for its product
* Promoting self-organization within the team
* Helping the Scrum Team to avoid or remove impediments to its progress, whether internal or external to the team
* Facilitating team events to ensure regular progress
* Educating key stakeholders in the product on Scrum principles
* Coaching the Development Team in self-organization and cross-functionality



Scrum Team : team comnsists of developers,testers,

**Sprint**: a sprint is a set period of time during which the specific works has to be completed and made ready for the review. Each sprint begins with a planning meeting. The duration of the sprint is determined by the scrum master. Normally each sprints lasts for 3 to 4 weeks or 30days. At the end of the Sprint, team submits the completed work to the project owner and project owner can take the decision to accept or reject the work during the planning meeting. Each iteration is divided into approx. 3 to 4 weeks duration which are called as Sprints.

**User stories are individual stories,divided to team members**.it tells to create a simplified description of a requirement

**Product** **Owner**:

* The Product Owner represents the product's stakeholders and the voice of the customer, and is accountable for ensuring that the team delivers value to the business.
* He is the one who gathers all the requirements from the end user/business/client and take care of all the backlogs.
* The Product Owner writes customer-centric items (typically user stories), prioritizes them based on importance and dependencies, and adds them to the Product Backlog.
* Scrum Teams should have one Product Owner. This role should not be combined with that of the Scrum Master.
* The Product Owner should focus on the business side of product development and spend the majority of their time liaising with stakeholders and should not dictate how the team reaches a technical solution.
* Communication is a core responsibility of the Product Owner.

As the face of the team to the stakeholders, the following are some of the communication tasks of the Product Owner to the stakeholders:[[19]](https://en.wikipedia.org/wiki/Scrum_(software_development)" \l "cite_note-19)

* demonstrates the solution to key stakeholders who were not present at a Sprint Review;
* defines and announces releases;
* communicates team status;
* organizes milestone reviews;
* educates stakeholders in the development process;
* negotiates priorities, scope, funding, and schedule;
* ensures that the Product Backlog is visible, transparent, and clear.

Example : BA

Backlogs: Whole set of requirements finalized by the product owner. The Product Backlog is a prioritized list of everything that might be included in a product. The Product Owner creates, maintains, and regularly re-orders the Product Backlog. The Product Owner uses the Product Backlog to adapt to emerging requirements, customer feedback, and market changes.

Sprint Backlogs: sprint requirements chosen from the product backlog which are required for that sprint.

User Stories: User Stories are Product Backlog Items that are descriptions of functionality. The User Story always takes the form:

"As a \_\_\_\_\_\_ I want to \_\_\_\_\_\_\_\_\_\_\_ so that I can \_\_\_\_\_\_."

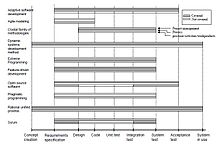
A user story is a very high-level definition of a requirement, containing just enough information so that the developers can produce a reasonable estimate of the effort to implement it

#### Agile principles[[edit](https://en.wikipedia.org/w/index.php?title=Agile_software_development&action=edit&section=3" \o "Edit section: Agile principles)]

The Agile Manifesto is based on twelve principles:[[13]](https://en.wikipedia.org/wiki/Agile_software_development" \l "cite_note-ManifestoPrinciples-13)

1. Customer satisfaction by early and continuous delivery of valuable software
2. Welcome changing requirements, even in late development
3. Working software is delivered frequently (weeks rather than months)
4. Close, daily cooperation between business people and developers
5. Projects are built around motivated individuals, who should be trusted
6. Face-to-face conversation is the best form of communication (co-location)
7. Working software is the principal measure of progress
8. Sustainable development, able to maintain a constant pace
9. Continuous attention to technical excellence and good design
10. Simplicity—the art of maximizing the amount of work not done—is essential
11. Best architectures, requirements, and designs emerge from self-organizing teams
12. Regularly, the team reflects on how to become more effective, and adjusts accordingly

## Agile methods[[edit](https://en.wikipedia.org/w/index.php?title=Agile_software_development&action=edit&section=14" \o "Edit section: Agile methods)]

[](https://en.wikipedia.org/wiki/File:SoftwareDevelopmentLifeCycle.jpg)

Software development life-cycle support[[36]](https://en.wikipedia.org/wiki/Agile_software_development" \l "cite_note-Abrahamsson2002-36)

Agile methods support a broad range of the [software development life cycle](https://en.wikipedia.org/wiki/Software_development_life_cycle).[[36]](https://en.wikipedia.org/wiki/Agile_software_development#cite_note-Abrahamsson2002-36) Some focus on the practices (e.g., XP, pragmatic programming, agile modeling), while some focus on managing the flow of work (e.g., Scrum, Kanban). Some support activities for requirements specification and development (e.g., FDD), while some seek to cover the full development life cycle (e.g., DSDM, [RUP](https://en.wikipedia.org/wiki/Rational_Unified_Process)).

Popular agile software development frameworks include (but are not limited to):

* [Adaptive software development](https://en.wikipedia.org/wiki/Adaptive_software_development) (ASD)
* [Agile modeling](https://en.wikipedia.org/wiki/Agile_modeling)
* [Agile Unified Process](https://en.wikipedia.org/wiki/Agile_Unified_Process) (AUP)
* [Crystal Clear methods](https://en.wikipedia.org/wiki/Crystal_Clear_(software_development))
* [Disciplined agile delivery](https://en.wikipedia.org/wiki/Disciplined_agile_delivery)
* [Dynamic systems development method](https://en.wikipedia.org/wiki/Dynamic_systems_development_method) (DSDM)
* [Extreme programming](https://en.wikipedia.org/wiki/Extreme_programming) (XP):  is a [software development methodology](https://en.wikipedia.org/wiki/Software_development_methodology) which is intended to improve software quality and responsiveness to changing customer requirements. As a type of [agile software development](https://en.wikipedia.org/wiki/Agile_software_development),[[1]](https://en.wikipedia.org/wiki/Extreme_programming#cite_note-Informatics85-1)[[2]](https://en.wikipedia.org/wiki/Extreme_programming#cite_note-UPenn49-2)[[3]](https://en.wikipedia.org/wiki/Extreme_programming#cite_note-USFCA601-3) it advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted.
* [Feature-driven development](https://en.wikipedia.org/wiki/Feature-driven_development) (FDD):  is an [iterative and incremental](https://en.wikipedia.org/wiki/Iterative_and_incremental_development) [software development process](https://en.wikipedia.org/wiki/Software_development_process). It is one of a number of [lightweight](https://en.wikipedia.org/wiki/Lightweight_methodology) or [Agile methods](https://en.wikipedia.org/wiki/Agile_software_development) for developing [software](https://en.wikipedia.org/wiki/Software).
* [Lean software development](https://en.wikipedia.org/wiki/Lean_software_development)
* [Kanban](https://en.wikipedia.org/wiki/Kanban_(development)): In [software development](https://en.wikipedia.org/wiki/Software_development), Kanban provides a visual [process-management](https://en.wikipedia.org/wiki/Process_management) system which aids decision-making about what, when and how much to produce. Although the method (inspired by the [Toyota Production System](https://en.wikipedia.org/wiki/Toyota_Production_System)[[1]](https://en.wikipedia.org/wiki/Kanban_(development)#cite_note-1) and [lean manufacturing](https://en.wikipedia.org/wiki/Lean_manufacturing))[[2]](https://en.wikipedia.org/wiki/Kanban_(development)#cite_note-2) originated in software development and IT, it may be applied to any professional service whose work outcome is intangible rather than physical.
* [Scrum](https://en.wikipedia.org/wiki/Scrum_(software_development)):  is an iterative and incremental [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) framework for managing product development.[[1]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-scrumalliance-1)[[2]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-gunther-2) It defines "a flexible, [holistic](https://en.wikipedia.org/wiki/Holism) product development strategy where a development team works as a unit to reach a common goal",[[3]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-TakeuchiNonaka-3) challenges assumptions of the "traditional, sequential approach"[[3]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-TakeuchiNonaka-3) to product development, and enables teams to self-organize by encouraging physical [co-location](https://en.wikipedia.org/wiki/Colocation_(business)) or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.
* [Scrumban](https://en.wikipedia.org/wiki/Scrumban):  is an [Agile](https://en.wikipedia.org/wiki/Agile_software_development) management methodology describing hybrids of [Scrum](https://en.wikipedia.org/wiki/Scrum_(software_development)) and [Kanban](https://en.wikipedia.org/wiki/Kanban_(development)" \o "Kanban (development)) and was originally designed as a way to transition from [Scrum](https://en.wikipedia.org/wiki/Scrum_(software_development)) to [Kanban](https://en.wikipedia.org/wiki/Kanban_(development)" \o "Kanban (development)). Today, Scrumban is a management framework that emerges when teams employ Scrum as their chosen way of working and use the Kanban Method as a lens through which to view, understand and continuously improve how they work.
* [Rapid application development](https://en.wikipedia.org/wiki/Rapid_application_development) (RAD)

Scrum and Xp are the important methodoliges implemented as a framework.

### Agile practices[[edit](https://en.wikipedia.org/w/index.php?title=Agile_software_development&action=edit&section=15" \o "Edit section: Agile practices)]

Agile development is supported by a number of concrete practices, covering areas like requirements, design, modelling, coding, testing, planning, risk management, process, quality, etc. Some notable agile practices include:

* [Acceptance test-driven development](https://en.wikipedia.org/wiki/Acceptance_test-driven_development) (ATDD)
* [Agile modeling](https://en.wikipedia.org/wiki/Agile_modeling)
* [Backlogs](https://en.wikipedia.org/wiki/Scrum_(development)#Product_Backlog) (Product and Sprint)
* [Behavior-driven development](https://en.wikipedia.org/wiki/Behavior-driven_development) (BDD)
* Business analyst designer method (BADM)[[37]](https://en.wikipedia.org/wiki/Agile_software_development#cite_note-37)
* [Cross-functional team](https://en.wikipedia.org/wiki/Cross-functional_team)
* [Continuous integration](https://en.wikipedia.org/wiki/Continuous_integration) (CI)
* [Domain-driven design](https://en.wikipedia.org/wiki/Domain-driven_design) (DDD)
* Information radiators (scrum board, task board, visual management board, [burndown chart](https://en.wikipedia.org/wiki/Burndown_chart" \o "Burndown chart))
* [Iterative and incremental development](https://en.wikipedia.org/wiki/Iterative_and_incremental_development) (IID)
* [Pair programming](https://en.wikipedia.org/wiki/Pair_programming)
* [Planning poker](https://en.wikipedia.org/wiki/Planning_poker)
* [Refactoring](https://en.wikipedia.org/wiki/Refactoring)
* [Scrum events](https://en.wikipedia.org/wiki/Scrum_(development)) (sprint planning, daily scrum, sprint review and retrospective)
* [Test-driven development](https://en.wikipedia.org/wiki/Test-driven_development) (TDD)
* [Agile testing](https://en.wikipedia.org/wiki/Agile_testing)
* [Timeboxing](https://en.wikipedia.org/wiki/Timeboxing)
* [User story](https://en.wikipedia.org/wiki/User_story)
* [Story-driven modeling](https://en.wikipedia.org/wiki/Story-driven_modeling)
* [Retrospective](https://en.wikipedia.org/wiki/Retrospective)
* [Velocity tracking](https://en.wikipedia.org/wiki/Velocity_(software_development))
* User Story Mapping

The Agile Alliance has provided a comprehensive online guide to applying agile these and other practices.[[38]](https://en.wikipedia.org/wiki/Agile_software_development#cite_note-Agile_Practices_Guide-38)

Scrum: is an iterative and incremental [agile software development](https://en.wikipedia.org/wiki/Agile_software_development) framework for managing product development.[[1]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-scrumalliance-1)[[2]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-gunther-2) It defines "a flexible, [holistic](https://en.wikipedia.org/wiki/Holism) product development strategy where a development team works as a unit to reach a common goal",[[3]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-TakeuchiNonaka-3) challenges assumptions of the "traditional, sequential approach"[[3]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-TakeuchiNonaka-3) to product development, and enables teams to self-organize by encouraging physical [co-location](https://en.wikipedia.org/wiki/Colocation_(business)) or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.

<https://en.wikipedia.org/wiki/Scrum_(software_development)#Product_Backlog>

http://searchsoftwarequality.techtarget.com/definition/Scrum-sprint

<https://www.tutorialspoint.com/sdlc/images/sdlc_agile_model.jpg>

----------------------------------------------------------------------------------------------------

Quality : Quality does not means same for everyone.

State Holders : Client, Producers and customers…

Something which expands beyong its limit is called Scalability.

Defect : any unexpected behavior

Defects are also referred as : Issue, bug, problem, fault and error.

Ex.Missing requirements , extra requirements if functions are not working…environment problems, documention errors etc.

Diff between FAULT and FAILURE ?

Fault is when a requirement or feature is working incorrectly, where as Failure is when whole AUT goes down, it usually accompanied by restoration activity.(restarting database, restarting the system…recovery operations)

Fault is any defect.

AUT : any kind of Application Under Test

SDLC : V-Model : Extremely useful for testers.. another development life cycle

Validation is the actual testing.

Left handed : verification testing and right handed is the validation testing

Verification …When we are checking if we built the right product.

Validation ….. did we build the product right.

Verification is static testing and validation is Dynamic testing.

Unit Testing : Done mostly by developer, done in development environment, manually or automatedly…

Unit code of code is developed and ready to test-verification is already done…and the output ready to be integrated based on the design

100% Test coverage :

Integaration Testing : input for integration testing is the “units of code are tested and ready to be integrated.”

Rules for Integration testing : two strategies : Bottom-Up and Top-Down

These are the directions to follow the integration process..

Integration testing is white box testing as we can see the code and also called as the GREY box testing.

Ex.from module A to B we see the coding , and module B to C we see the code ,which is white box testing. Once the A, B, C modules are integrated ..we test it with some input and get check for the correct output, in this scenario we don’t need to see any code which is again Black box testing. But at the end if we get the incorrect output then we need to see the code which again is White box testing.Hence it is the shades of mixing white box and black box which is sometimes called as Grey Box testing.

STUBS : two module Calling function and called funtions

A is 100% developed and B is 0% developed ..in order to test integration we create a dummy module as a proxy which has limited functionality and which gives temperory solution. the dummy one is called STUB.

DRIVERS : IF A is 0% developed and B is 100% developed ,then we create a dummy for called function which is called as DRIVER.

Calling function is STUB and called function is DRIVER. The output of the Integration testing is the Entire system put together to test.

SYSTEM TESTING : is a Black Box Testing….done by QA team in testing environment. functional and non functional testing can be done in system testing.

System testing is done in stages..

Questions :

What is software,project, domain,technology and testing,defect

SDLC :Initiate-define-design-build-test-deplyment

Main topics : youtube video

Quality :

What is Testing : **Testing** is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. In simple words, **testing** is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

Waterfall SDLC :

Funtional non functional requirements need

Test Strategy

Agile test strategy, Agile SDLC(iterative approach to sdlc)

Test Scope

Test types

Components testing based on internal architecture of the system.

Static Testing : testing the specifications, not actual system.by checking documents we can avoid defect multiplications. testing quality work products,meaning checking documents not final product ,

Developers use code reviews without code executions, just reviewing code, line by line they can find some errors and they can fix them.

1233780375 / item no. 190322030691 / no. 510 797 3333 / usps tracking number : 92748999920645553013952941 / 9th sep 1:46pm / mailbox