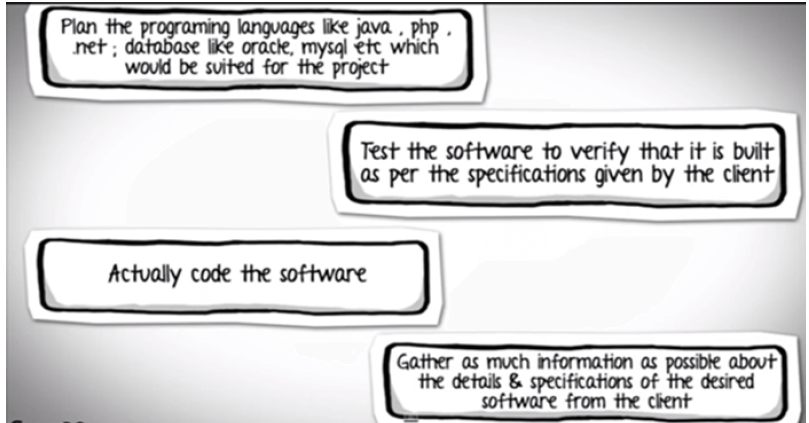
1. **What is SDLC?**

**Let us try to understand this heavy term by a simple example:**

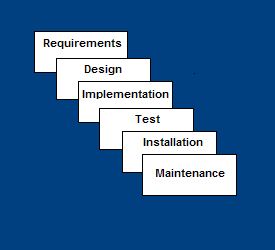
Suppose you are assigned a task, to develop an application. Now, irrespective of whether SDLC is new to you or not, try to make a guess about the sequence of steps you will follow, to achieve the task by looking at the given image:

Let’s hang on for a couple of minutes before we reveal the correct sequence:



Let’s hang on for a couple of minutes before we reveal the correct sequence:

2. The correct sequence would be:



So the correct order- shown is nothing but the phases which form SDLC i.e.

Activities performed in Software Development Cycle Activities.

Requirement Gathering stage: Gather as much information as possible about the details & specifications of the desired software from the client. This is nothing but the Requirements gathering stage.

Design Stage: Plan the programming language like Java, PHP.net; database like Oracle, MySQL, etc. which would be suited for the project, also some high-level functions & architecture.

Build Stage: After design stage, building the application i.e. actually coding the software takes place.

Test Stage: Next, you test the software to verify that it is built as per the specifications given by the client.

Deployment stage: Deploy the application in the respective environment

Maintenance stage: Once your system is ready to use, you may require to change the code later on as per customer request.

**This sequence of activities to develop the given software is nothing but SDLC**

**3. Waterfall model:**

**Since the inception of software industry, many models got developed with each model superior over the former in a way or the other. These models were designed keeping the software in mind and modified to suit the same.**

**The first and the foremost one is waterfall.**

**The waterfall model is a linear sequential design approach. In software development, it tends to be among the less iterative and flexible approaches, as progress flows in largely one direction ("downwards" like a waterfall) through the phases of conception, initiation, analysis, design, testing, deployment and maintenance.**

So the previous example we discussed was a waterfall model. Starting with:

1. System and software requirements are captured

2. Analysis of these requirements

3. Design

4. Coding: the development

5. Testing: the software and reporting of defects

6. Operations: the installation, migration, support, and maintenance of complete systems

Thus the waterfall model maintains that one should move to a phase only when it’s receding phase is reviewed and verified.

**4. This gives rise to many limitations due to lack flexibility**

As you may observe, that testing in the model starts only after implementation is done. But if you are working in the large project, where the systems are complex, it's easy to miss out the key details in the requirements phase itself. In such cases, an entirely wrong product will be delivered to the client and you might have to start afresh with the project.

Assessments of thousands of projects have shown that defects introduced during requirements & design make up close to half of the total number of defects.

* All requirements must be known upfront
* Deliverables created for each phase are considered frozen – lack of flexibility
* Little opportunity for customer to preview the system (until it may be too late)

To address this concern, the V model was developed where for every phase, in the Development life cycle there is a corresponding Testing phase

**5. V-Shaped SDLC Model**

* A variant of the Waterfall that emphasizes the verification and validation of the product.
* Testing of the product is planned in parallel with a corresponding phase of development
* The left side of the model is Software Development Life Cycle - SDLC
* The right side of the model is Software Test Life Cycle - STLC
* The entire figure looks like a V, hence the name V – model
* For every SDLC activity on the left, there is a corresponding Testing activity on the right

Left Hand side:

As said earlier, left hand side activities are the development activities. Normally we feel, what testing can we do in development phase, but this is the beauty of this model which demonstrates that testing can be done in all phase of development activities as well.

Requirement analysis:  In this phase the requirements are collected, analyzed and studied. Here how the system is implemented, is not important but, what the system is supposed to do, is important. Brain storming sessions/walkthrough, interviews are done to have the objectives clear.

* Verification activities: Requirements reviews.
* Validation activities: Creation of UAT ([User acceptance test](http://www.softwaretestinghelp.com/successful-user-acceptance-testing/)) test cases
* Artifacts produced: Requirements understanding document, UAT test cases.

System requirements / High level design:  In this phase a high level design of the software is build. The team studies and investigates on how the requirements could be implemented. The technical feasibility of the requirements is also studied. The team also comes up with the modules that would be created/ dependencies, Hardware / software needs

* Verification activities: Design reviews
* Validation activities: Creation of [System test plan](http://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/) and cases, Creation of traceability metrics
* Artifacts produced: System test cases, Feasibility reports, System test plan, Hardware software requirements, and modules to be created etc.

Architectural design: In this phase, based on the high level design, software architecture is created. The modules, their relationships and dependencies, architectural diagrams, database tables, technology details are all finalized in this phase.

* Verification activities: Design reviews
* Validation activities: Integration test plan and test cases.
* Artifacts produced: Design documents, Integration test plan and test cases, Database table designs etc.

Module design/ Low level Design: In this phase each and every module or the software components are designed individually. Methods, classes, interfaces, data types etc are all finalized in this phase.

* Verification activities: Design reviews
* Validation activities: Creation and review of unit test cases.
* Artifacts produced: Unit test cases,

Implementation / Code: In this phase, actual coding is done.

* Verification activities: Code review, test cases review
* Validation activities: Creation of functional test cases.
* Artifacts produced: test cases, review checklist.

Right Hand Side:

Right hand side demonstrates the testing activities or the Validation Phase. We will start from bottom.

Unit Testing: In this phase all the unit test case, created in the Low level design phase are executed.

\*Unit testing  is a white box testing technique, where a piece of code is written which invokes a method (or any other piece of code) to test whether the code snippet is giving the expected output or not. This testing is basically performed by the development team. In case of any anomaly, defects are logged and tracked.

Artifacts produced:  Unit test execution results

Integration Testing:  In this phase the integration test cases are executed which were created in the Architectural design phase. In case of any anomalies, defects are logged and tracked.

\*Integration Testing:  Integration testing is a technique where the unit tested modules are integrated and tested whether the integrated modules are rendering the expected results. In simpler words, It validates whether the components of the application work together as expected.

Artifacts produced: Integration test results.

Systems testing: In this phase all the system test cases, functional test cases and nonfunctional test cases are executed. In other words, the actual and full fledge testing of the application takes place here. Defects are logged and tracked for its closure. Progress reporting is also a major part in this phase. The traceability metrics are updated to check the coverage and risk mitigated.

Artifacts produced: Test results, Test logs, defect report, test summary report and updated traceability matrices.

User acceptance Testing:  Acceptance testing is basically related to the business requirements testing. Here testing is done to validate that the business requirements are met in the user environment. Compatibility testing and sometimes nonfunctional testing ([Load, stress and volume](http://www.softwaretestinghelp.com/what-is-performance-testing-load-testing-stress-testing/)) testing are also done in this phase.

Artifacts produced: UAT results, Updated Business coverage matrices.

6. Apart from V model, there are iterative / incremental development models, where development is carried in phases. Each phase comprises of its independent set of development and testing activities.

To simplify, an application is developed in multiple increments, sub-models of the application are developed in each of these increments. So the software or the pplication is developed in multiple increments

* Then slowly add increased functionality
* The incremental model prioritizes requirements of the system and then implements them in groups.
* Each subsequent release of the system adds function to the previous release, until all designed functionality has been implemented.
* Application is not developed in one go but multiple increments.
* Sub parts of the application are developed in each of these increments.

**7. Iterative vs incremental:**

[IterativeDevelopment](http://wiki.c2.com/?IterativeDevelopment) is often confused with [IncrementalDevelopment](http://wiki.c2.com/?IncrementalDevelopment). [IterativeDevelopment](http://wiki.c2.com/?IterativeDevelopment) is about planned rework. You create something, review it and then change it (hopefully improving it) based on the feedback.

A good analogy is when authors write books.

[IterativeDevelopment](http://wiki.c2.com/?IterativeDevelopment) means:

* I write loads of stuff that's a complete mess
* I go through it throwing out the irrelevant drivel, expanding on the important bits, and sorting out the structure
* I go through it again now I can start to see the shape of it, sorting it some more
* I go through it yet again, etc, until it's [GoodEnough](http://wiki.c2.com/?GoodEnough) to publish it

[IncrementalDevelopment](http://wiki.c2.com/?IncrementalDevelopment) means:

* I write part one
* I write part two
* I write part three, etc, until the book is finished

**8. AGILE:**

Agile is both interative and incremental development when done properly. Each iteration delivers a fully functional increment, just as a plant works at every stage of growth. If it does this every iteration is “potentially shippable" and in the ideal case is shipped to a set of end users who use it to get real work done and provide feedback.

* Each iteration typically lasts from about one to three weeks.
* break their work into actions that can be completed within timeboxed iterations, called Sprints (30 days or less) and track progress and re-plan in 15-minute [stand-up meetings](https://en.wikipedia.org/wiki/Stand-up_meeting), called Daily Scrums

Each sprint starts with a sprint planning event that aims to define a sprint backlog, identify the work for the sprint, and make an estimated forecast for the sprint goal. Each sprint ends with a sprint review and sprint retrospective,[[12]](https://en.wikipedia.org/wiki/Scrum_(software_development)#cite_note-autogenerated1-12) that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.

Each day during a sprint, the team holds a daily scrum (or [stand-up](https://en.wikipedia.org/wiki/Stand-up_meeting)) with specific guidelines:

* During the daily scrum, each team member typically answers three questions:
  + What did I complete yesterday that contributed to the team meeting our sprint goal?
  + What do I plan to complete today to contribute to the team meeting our sprint goal?
  + Do I see any impediment that could prevent me or the team from meeting our sprint goal?

At the end of a sprint, the team holds two events: the sprint review and the sprint retrospective.

At the sprint review, the team:

* Reviews the work that was completed and the planned work that was not completed
* Presents the completed work to the stakeholders (a.k.a. the demo)
* The team and the stakeholders collaborate on what to work on next

At the sprint retrospective, the team:

* Reflects on the past sprint
* Identifies and agrees on [continuous process improvement](https://en.wikipedia.org/wiki/Continual_improvement_process) actions