**2.1. Introduction to Software Development Life Cycle**

Software Development Life Cycle (or SDLC) is the process which is followed to develop a software product. It is a structured way of building software applications. Most organizations have a process in place for developing software; this process may, at times, be customized based on the organizations requirement and framework followed by organization.Life Cycle Model is one specific embodiment of a software process

**SDLC Phases**

•    The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process.

•    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.

•    The following picture depicts the phases that are part of a development cycle.

**Requirements Gathering**

A Software Requirement Specification or SRS is a document which records expected behavior of the system or software which needs to be developed.

**Design**  
•    Software design is the blueprint of the system, which once completed can be provided to developers for code development.   
•    Based on the components in design, they are translated into software modules/functions/libraries, etc and these pieces together form a software system.

**Coding**  
•    During this phase, the blueprint of the software is turned to reality by developing the source code of the entire application.   
•    Time taken to complete the development depends on the size of the application and number of programmers involved.

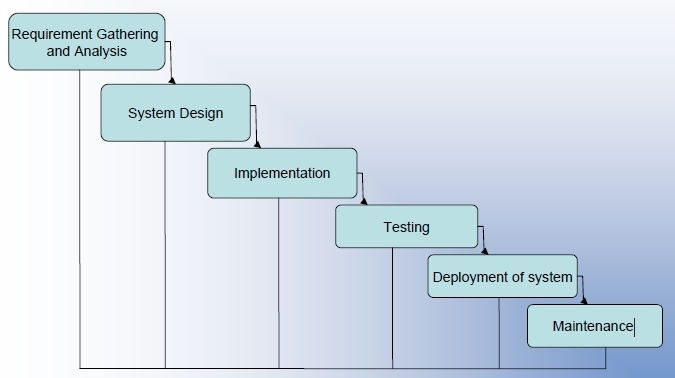
**2.2. SDLC Models**

There are various software development life cycle models defined and designed which are followed during software development process.

These models are also referred as "Software Development Process Models".

Each process model follows a Series of steps unique to its type, in order to ensure success in process of software development.

**Waterfall Model**



In the Waterfall model, the different phases of the software development life cycle are explicitly recognized as mentioned in picture  below: .

In this model, the project takes a straight line path.Since different phases are explicitly recognized,the finalisation of the Contract is seen with reference to delivery and payment schedules of different phases.

The waterfall model is the oldest and most widely used in software development life cycle. It is also known as sequential model. Each phase must be completed before the next phase begins.At the end of each review takes place to determine if the project is on the right track or not.

Since requirement validation is not done in an explicit manner,it may result in design and development of large quantities of unusable code. It may also result in extensive rework later,as document-driven standards, force elaborate specifications of poorly understood user interfaces and decision- support functions.

**Advantages**

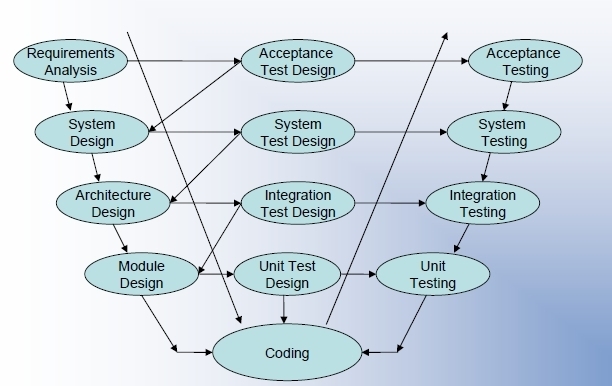
Below are some of the advantages of the Waterfall Model

•    Simple and easy to use.   
•    Easy to manage due to the rigidity of the model – each phase has specific deliverable and a review process.   
•    Phases are processed and completed one at a time.   
•    Works well for smaller projects where requirements are very well understood.

**Disadvantages**

•    Adjusting scope during the life cycle can kill a project   
•    No working software is produced until late during the life cycle.   
•    High amounts of risk and uncertainty.   
•    Poor model for complex and object-oriented projects.   
•    Poor model where requirements are at a moderate to high risk of changing.

**V-Model**



V- model is a sequential path of execution of processes.Each phase must be completed before the next phase begins.Testing is emphasized in this model more so than the waterfall model though. The testing procedures are developed early in the life cycle before any coding is done, during each of the phases preceding implementation. Testing of the product is planned in parallel with a corresponding phase of development.

**Advantages**

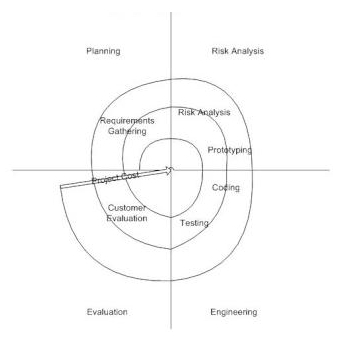
•    Simple and easy to use  
•    Each phase has specific deliverable  
•    Higher chance of success over the waterfall model due to the development of  test plans early on during the life cycle  
•    Works well for where requirements are easily understood

**Disadvantages**

•    Very inflexible, like the waterfall model   
•    Little flexibility and adjusting scope is difficult and expensive   
•    Software is developed during the implementation phase, so no early prototypes of the software are produced   
•    Model does not provide a clear path for problems found during testing phases.

**The Spiral Model**

A well-managed development project should take a straight line approach,right from feasibility report to analysis, design, testing, and acceptance to operation. But in actuality, the project needs to proceed iteratively by taking a spiral path. The project starts with the analysis phase, then proceeds towards design, moves back to analysis and then design, followed by coding (version 1.0) and back to design and so forth.



The basic idea behind this model is that you start on a small scale,in the middle of the core functionality, explore the risks (such as poorly understood requirements and architecture, potential performance problems, problems in underlying technology etc.) and then make a plan to handle the risks. This should be followed by a commitment to approach, for the next iteration. Each iteration moves your project to a larger scale. One layer of the project is rolled first, to check what was actually wanted, and then, work on the next layer is started.

Each iteration involves the following steps:

- To determine objectives, alternatives and constraints

- To identify and resolve risks

- To evaluate alternatives

- To develop the deliverables for that iteration, and to verify their correctness

- To plan the next iteration

- To commit to an approach for the next iteration

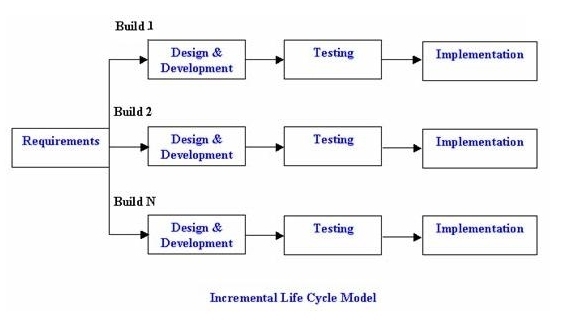
**Advantages**  
•    Good for large projects and also for medium to high risk projects  
•    High amount of risk analysis hence avoidance of risk is enhanced  
•    String approval and documentation control  
•    Additional functionality can be added at a later date  
•    Software is produced early in the software life cycle.

**Disadvantages**

•    Can be a costly model to use.   
•    Risk analysis requires highly specific expertise.   
•    Project’s success is highly dependent on the risk analysis phase.   
•    Doesn’t work well for smaller projects.

**Incremental model**

The whole requirement is divided into various builds. Multiple development cycles make the life cycle a “multi-waterfall” cycle. Cycles are divided up into smaller, more easily managed modules. Each module passes through the requirements, design,implementation and testing phases. A working version of software is produced during the first module.Each subsequent release of the module adds function to the previous release. The process continues till the complete system is achieved



**Advantages**

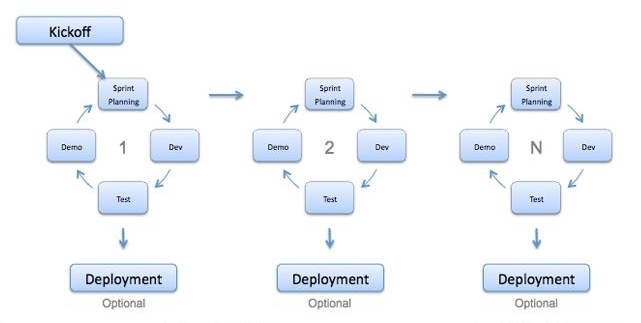
•    Generates working software quickly and early during the software life cycle.   
•    More flexible – less costly to change scope and requirements.   
•    Easier to test and debug during a smaller iteration.   
•    Customer can respond to each built.   
•    Lowers initial delivery cost.   
•    Easier to manage risk because risky pieces are identified and handled during each iteration.

**Disadvantages**

•    Needs good planning and design.   
•    Needs a clear and complete definition of the whole system before it can be broken down and built incrementally.   
•    Expensive than waterfall model

**Agile Model**

Agile model is also a type of Incremental model which is used for time critical applications. Software is developed in incremental, rapid cycles. This results in small incremental releases with each release building on previous functionality. Each release is thoroughly tested to ensure software quality is maintained.Extreme Programming (XP), Scrum are some of the most well known agile development model



External link: <https://help.rallydev.com/sites/default/files/multimedia/videos/agile_primer.swf>

**Advantages**

•    People and interactions are emphasized rather than process and tools. Customers, developers and testers constantly interact with each other.   
•    Working software is delivered frequently (weeks rather than months).   
•    Face-to-face conversation is the best form of communication.   
•    Close, daily cooperation between business people and developers.   
•    Continuous attention to technical excellence and good design.   
•    Regular adaptation to changing circumstances.   
•    Even late changes in requirements are welcome.

**Disadvantages**

•    In case of some software deliverables, especially the large ones, it is difficult to assess the effort required at the beginning of the software development life cycle.   
•    There is lack of emphasis on necessary designing and documentation.   
•    The project can easily get taken off track if the customer representative is not clear what final outcome that they want.   
•    Only senior programmers are capable of taking the kind of decisions required during the development process. Hence it has no place for new programmers, unless combined with experienced resources.

Different software development models have their own advantages and disadvantages. The right model or combination of models has to be chosen based on the project and all other related factors.

•    Risks and contingencies  
•    Approvals

**Contents of Test Plan**

**Scope clause:** Defines what will be covered in a project

Test items

* Items of software, hardware, and combinations of these which will be tested.
* Programmers or Developers point of view
* Contents depend on level of the Test Plan

Features to be tested

* Which parts of the software specification are to be tested?
* User or customer point of view
* Contents depend on level of the Test Plan
* Help to focus on software risks

Features not to be tested

* Which parts of the software specification are to be excluded?
* Reason for not testing
* Normally lists low risk features

**Resource clause:** Gives overall view of the resources to deliver the tasks.

Environmental Needs

* What is needed in the way of testing software, hardware etc.
* Are there any special requirements for this test plan, such as special hardware such as simulators, static generators etc.
* How will test data be provided. Are there special collection requirements or specific ranges of data that must be provided?
* How much testing will be done on each component of a multi-part feature?
* Specific versions of other supporting software.
* Restricted use of the system during testing

Responsibilities

* Lists who has responsibility for delivering various part of the plan like setting risks.
* Selecting features to be tested and not tested.
* Setting overall strategy for this level of plan.
* Ensuring all required elements are in place for testing.
* Providing for resolution of scheduling conflicts, especially, if testing is done on the production system.
* Who provides the required training?
* Who makes the critical go/no go decisions for items not covered in the test plans?

Staffing and training needs

* Number of people and skills needed to execute the plan
* Training needs might include
* How to use tool
* Testing Methodology
* Interfacing systems
* Management systems such as defect tracking, configuration management, basic business knowledge (related to the system under test), etc.

**Time clause:**Specifies the tasks to be undertaken to meet the quality objectives and when they occur

Testing tasks

* Tasks that are necessary for performing test execution and the dependencies, the elapsed time they will take, and resource required.

Examples include:

* Test Planning
* Test Analysis
* Test Design
* Test Implementation
* Test Execution
* Test Evaluation

Schedule

* This specifies when the tasks will take place.
* It should be based on realistic and validated estimates.
* Build around milestones
* All relevant milestones should be identified
* Reference to project schedule.
* It is always best to tie all test dates directly to their related development activity dates. This prevents the test team from being perceived as the cause of a delay. For example, if system testing is to begin after delivery of the final build, then system testing begins the day after delivery. If the delivery is late, system testing starts from the day of delivery, not on a specific date. This is called dependent or relative dating.
* Include deadlines
* Estimation and resources are considered
* How slippage in the schedule will be handled should also be addressed here.
  + If the estimates for the development of the application are inaccurate, the entire project plan will slip. The first area of a project plan to get cut when it comes to crunch time at the end of a project is the testing. If the users know in advance that a slippage in the development will cause a slippage in the test and the overall delivery of the system, they just may be little more tolerant, if they know it’s in their interest to get a better tested application.
  + By spelling out the effects here you have a chance to discuss them in advance of their actual occurrence. You may even get the users to agree to few defects in advance, if the schedule slips.

**Quality clause**: It specifies to what standard the testing will be completed against the scope.

* Introduction section
* This gives a high level view of the testing standard. Specifies overall test objectives.
  + A basic description of the project or release including key features,history, etc.,
  + An introduction to the plan that describes the scope of the plan.
  + Any references to other plans, documents or items that contain information relevant to this project/process. If required a references section can be created to contain all reference documents.
* Other items
* Resource and budget constraints, scope of the testing effort, how testing relates to other evaluation activities (Analysis & Reviews),and possible process to be used for change control and communication and coordination of key activities.

As this is the “Executive Summary” the information kept here should be brief and to the point.

* Item pass/fail criteria
  + This defines how the item being tested has passed and completion criteria for this plan. This is a critical aspect of any test plan and should be appropriate to the level of the plan. Each test item needs to have an expected result relative to the test level.
* At the Unit test level this criteria could be as
* All test cases completed.
* A specified percentage of cases completed with a percentage containing some number of minor defects.
* Code coverage tool indicates all code covered.
* At the Master test plan level this criteria could be as
* All lower level plans completed.
* A specified number of plans completed without errors and a percentage with minor defects.

This could be an individual test case level criterion or a unit level plan or itcan be generalfunctional requirements for higher level plans.

* Describes severity
* The number and severity of defects located
* Approach
* This is the overall test strategy for this test plan. It should be appropriate to the levelof the plan (master, acceptance, etc.) and should be in agreement with all higher andlower levels of plans. Overall rules and processes should be identified.This gives details of how the testing process will be followed.
* Test deliverables
  + This specifies the test documents and other deliverables to be produced.
* Test Plans
* Test Design Specs
* Test Cases
* Custom Tools
* Defect Reports
* Test Summary Reports
* Simulators (if any)

One thing that is not a test deliverable is the software itself that is listed under test itemsand is delivered by development.

**Risk clause:**Defines in advance what will happen if something goes wrong with the plan.

* Suspension criteria and resumption requirements
* This is a particular risk clause to define when testing will stop and restart.
* Conditions for temporary suspension
* If the number or type of defects reaches a point where the follow ontesting has no value, it makes no sense to continue the test.
* Metrics can be used to derive condition
* Specifies what constitutes stoppage for a test or series of tests andwhat is the acceptable level of defects that will allow the testing toproceed past the defects.
* Risks and contingencies
* This defines all risk events, their likelihood, impact with an emphasis on the testing process and counter measures to over come them. Examples
* Lack of personnel resources when testing is to begin.
* Lack of availability of required hardware, software, data or tools.
* Late delivery of the software, hardware or tools.
* Delays in training on the application and/or tools.
* Changes to the original requirements or designs.
* Specify what will be done for various events, for example requirements definition will be complete by January 1, 19XX, and, if the requirements change after that date, the following actions will be taken.
* The test schedule and development schedule will move out an appropriate number of days. This rarely occurs, as most projects tend to have fixed delivery dates.
* The number of test performed will be reduced.
* The number of acceptable defects will be increased.
* These two items could lower the overall quality of the delivered product.
  + Resources will be added to the test team.
  + The test team will work overtime.
* This could affect team morale.
* The scope of the plan may be changed.

This section helps us to make

* Intelligent and informed decisions
* Consider risk priorities to move features from “Features to Be Tested” to “Feature not to be Tested”
* Others
* Test plan identifier
* This is a unique name or code by which the plan can be identified in the project's documentation including version.
* Approvals
* This clause contains the signatures of the various stakeholders in the plan, to show they agree in advance with what it says.
* Various stakeholders signature
* Different Approvers for different levels of Test Plan
* Approvers should involve in the creation and/or review of the plan

**Test Strategy VS Test Plan**

* In any project, testing is an integral aspect of the process. Testing is a way to make sure that the project is up to standards. In order to ensure quality and the proper testing methods, the company usually draws up papers documenting the process of the project. These documents also serve as proof that the project has complied with any and all required testing methods. Test strategy and test plan are two such documents that the company draws up
* A test strategy outlines the testing approach that the company aims to take. The purpose of a test strategy is to inform project managers, testers, and developers about some of the key issues of the testing process. A test plan, on the other hand, is a detailed document which lists the systematic approach to testing the system. Its aim is to map out the detailedunderstanding of the workflow
* The main difference between the two is their scope. The test strategy is a mere outline of the entire project testing. It covers the general views and objectives that the project should achieve, in addition to the approaches that should be used to achieve them. It is usually a project-wide document that is shared with the entire project team and/or the whole company
* The test plan is more detailed. It incorporates a lot of the specifics of that which has been outlined in the test strategy, such as who does the actual testing and how the steps are to be conducted. However, it will always follow the guidelines as stated in the test strategy. The test plan may deal with the entire project as a whole or a specific part of the project. In some smaller projects, the test strategy can also be found as a section of the test plan.
* Some companies include the “Test Approach” or “Strategy” inside the Test Plan, which is fine and it is usually the case for small projects. However, for larger projects, there is one Test Strategy document and different number of Test Plans for each phase or level of testing
* Furthermore, the test strategy is a static document, i.e. it is the same throughout the project from the beginning to the end. However, the test plan may often be revised in face of unforeseen circumstances faced by the project. Hence, it is versatile

**3. Test Design**

This phase involves creation, verification and rework of test cases & test scripts. Test data, is identified/created and is reviewed and then reworked as well.

Activities:

• Create test cases, automation scripts (if applicable)

• Review and baseline test cases and scripts

• Create test data (If Test Environment is available)

Deliverables:

• Test cases/scripts

• Test data

**4. Test Environment Setup**

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and can be done in parallel with Test Case Development Stage. Test team may not be involved in this activity if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

Activities:

• Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.

• Setup test Environment and test data

• Perform smoke test on the build

Deliverables:

• Environment ready with test data set up

• Smoke Test Results.

**5. Test Execution**

During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

Activities:

•    Execute tests as per plan.  
•    Document test results and log defects for failed cases   
•    Map defects to test cases in RTM   
•    Retest the defect fixes  
•    Track the defects to closure

Deliverables:

•     Completed RTM with execution status   
•    Execute tests as per plan.   
•     Test cases updated with results

**6. Test Cycle Closure**

Testing team will meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future.

Activities:

•    Evaluate cycle completion criteria based on Time,Test coverage,Cost,Software,Critical Business objectives and Quality  
•    Prepare test metrics based on the above parameters.  
•    Document the learning out of the project  
•    Prepare Test closure report   
•    Qualitative and quantitative reporting of quality of the work product to the customer.  
•    Test result analysis to find out the defect distribution by type and severity

Deliverables:

•    Test Closure report   
•    Test metrics

**2.4. Levels of Testing**

There are broadly four levels of testing done in any Testing Project

•    Unit Testing  
•    Integration Testing  
•    System Testing  
•    Acceptance Testing

**Unit Testing**

Unit Testing is testing unit of code (program or set of programs) using Unit Test Specifications,after coding is completed. It involves the basic testing of a piece of code, the size of which is often undefined in practice, although it is usually a function or a subroutine. It tests the functional aspects. It tests smallest testable part of an application. It is done by the developer of the functionality.

Example:

Testing of a cobol program in the reservation system that calculates the price for the ticket requested based on the inputs supplied to the program from the calling cobol program.

**Integration Testing**

The process of testing interfaces and data flows between the programs within a sub system,and between the sub-systems within a system is known as Integration Testing.

In Integration testing tester must look for bugs in the relationship and the interfaces between pairs of components and groups of components under test.

Example:

Check whether the calling program in previous example passes the right information relating to starting station and destination, date of journey, discounts to be applied andclass requested.

**System Testing**

System Testing is a test, executed by the developer or independent test team in a laboratory environment that should demonstrate that the developed system or subsystems meet the requirements set in the functional and quality specifications.

It is a process of proving that the system meets its stated design specifications (design documents) w.r.t criteria such as recoverability, maintainability and security

Example:

– Comprehensive black box testing of railway reservation system with transactionsinitiated and validations performed on databases and reports generated after thecompletion of the transactions

**Acceptance Testing**

It is a test, executed by the user(s) and system manager(s) in an environment simulating the operational environment to the greatest possible extent, that should demonstrate that the developed system meets the functional and quality requirements.

**Non Functional and Other types of Testing**

•    Performance testing  
•    Volume testing  
•    Load testing  
•    Limit testing  
•    Stress testing  
•    Disaster Testing  
•    Recovery testing  
•    Security testing  
•    Reliability testing  
•    Installation Testing  
•    Usability Testing  
•    Accessibility Testing  
•    Regression testing