Module 1 – Syllabus

Introduction to Software Engineering: What is Software Engineering, Characteristics of Software Life cycle of a software system: software design, development, testing, deployment, Maintenance. Project planning phase: project objectives, scope of the software system, empirical estimation models, COCOMO, staffing and personnel planning. Software Engineering models: Predictive software engineering models, model approaches, prerequisites, predictive and adaptive waterfall, waterfall with feedback (Sashimi), incremental waterfall, V model; Prototyping and prototyping models.
Software requirements specification, Eliciting Software requirements, Requirement specifications, Software requirements engineering concepts, Requirements modelling, Requirements documentation. Use cases and User stories

**Software** is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called **software product.**

**Engineering** on the other hand, is all about developing products, using well-defined, scientific principles and methods


**Software engineering** is an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures. The outcome of software engineering is an efficient and reliable software product.

IEEE defines software engineering as:

(1) The application of a systematic,disciplined,quantifiable approach to the development,operation and maintenance of software; that is, the application of engineering to software.

(2) The study of approaches as in the above statement.

## SDLC

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares. The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

SDLC is the acronym of Software Development Life Cycle.

It is also called as Software Development Process.

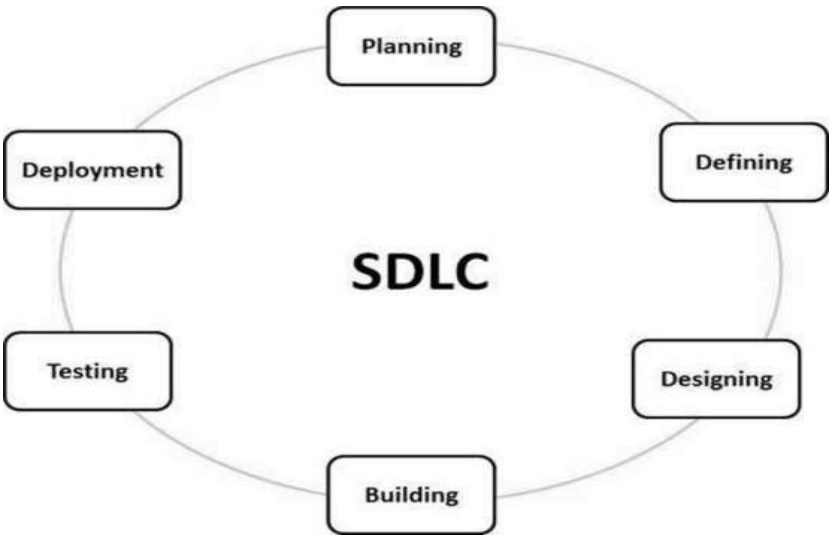
SDLC is a framework defining tasks performed at each step in the software development process.

ISO/IEC 12207 is an international standard for software life-cycle processes. It aims to be the standard that defines all the tasks required for developing and maintaining software.

## What is SDLC?

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.

The following figure is a graphical representation of the various stages of a typical SDLC.



Planning and Requirement Analysis

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

Defining Requirements

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

Designing the Product Architecture

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

Building or Developing the Product

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Testing the Product

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

Deployment in the Market and Maintenance

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT- User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

The Importance of SDLC

It is necessary give structure to the several phases involved in software development efforts and SDLC serves that purpose. The cycle does not conclude until all the requirements have been fulfilled, and will continue until all the potential needs are adjusted within the system. The biggest advantage of the software development life cycle is that it provides control of the development process to some extent, and ensures the system complies with all the requirements that have been specified. However, there are some disadvantages to using SDLC. It does not work so well where there there are levels of uncertainty or unnecessary overheads. It directs the development efforts with an emphasis on planning, but the system does not encourage creative input or innovation throughout the lifecycle. For those reasons, organizations tend to adopt Agile, and other such methodologies, which are incremental rather than sequential.

Project Planning Phase

The **Project Planning Phase** is the second phase in the *project life cycle*. It involves creating of a set of plans to help guide your team through the execution and closure phases of the project.

The plans created during this phase will help you to manage time, cost, quality, change, risk and issues. They will also help you manage staff and external suppliers, to ensure that you deliver the project on time and within budget.

There are 10 Project Planning steps you need to take to complete the Project *Planning Phase* efficiently.

1. Create a Project Plan
2. Create a Resource Plan
3. Create a Financial Plan
4. Create a Quality Plan
5. Create a Risk Plan
6. Create a  Acceptance Plan
7. Create a Communications Plan
8. Create a Procurement Plan
9. Contract the Suppliers

10. Perform a Phase Review

Project planning is an organized and integrated management process, which focuses on activities required for successful completion of the project. It prevents obstacles that arise in the project such as changes in projects or organization's objectives, non-availability of resources, and so on. Project planning also helps in better utilization of resources and optimal usage of the allotted time for a project.

The other objectives of project planning are listed below.

It defines the roles and responsibilities of the project management team members.
It ensures that the project management team works according to the business objectives.
It checks feasibility of the schedule and user requirements.
It determines project constraints.

Project Objective Definition

A project objective describes the desired results of a project, which often includes a tangible item. An objective is specific and measurable, and must meet time, budget, and quality constraints.

Objectives can be used in project planning for business, government, nonprofit organizations, and even for personal use (for example, in resumes to describe the exact position a job-seeker wants). A project may have one objective, many parallel objectives, or several objectives that must be achieved sequentially. To produce the most benefit, objectives must be defined early in the project life cycle, in phase two, the planning phase.

**Benefits of the Well-Written Objective**

A well written objective is crucial because it can affect every step of the project life cycle. When you create a specific objective, you give your team a greater chance of achieving the objective because they know precisely what they're working towards. Clear project objectives also support the current emphasis on total quality management: every member of the team can consider themselves responsible for quality, because the whole team can see the desired outcome from the beginning of the project.

Introduction: Software Planning

Software project management begins with a set of activities that are collectively called project planning

The manager and the software team must estimate the work that is to be done, the resources required and the time that will be taken to complete the project

Estimates should always be made with the future needs in mind and also taking into account the various degree of uncertainty

Process and project metrics provides the historical perspective and a powerful input for the generation of quantitative estimates

As estimation lays a foundation for all other project planning activities, project planning paves the way for successful software engineering.

Project Planning Objectives

The objective of software project planning is to provide a framework that enables the project manager to make some reasonable estimates of resources, cost and schedule

These estimates are made at the beginning of a software project and should be updated regularly as the project progresses towards completion

The planning objective is achieved through a process of information discovery that leads to the formulation of reasonable estimates

Software Scope

First activity in the planning of the software project is the determination of the scope of the software

Scope of the software describes the data and control to be processed, function, performance, constraints, interfaces and reliability

Obtaining information necessary for scope

Feasibility

Resources

The second task involved in software planning is the estimation of the resources required to accommodate the software development effort

Project Development Resources

Each resource is specified with 4 characteristics:

Description of the resource.

Statement of availability.

Time when the resource will be required.

Duration of time that resource will be applied

Software Project Estimation

Software is the most expensive element of virtually all computer-based systems

Cost over run can be disastrous for the software developer

To achieve a reliable cost and effort estimates, a number of options arise :

Delay estimation until late in the project.

Base estimates on similar projects that have already been completed.

Use relatively simple decomposition techniques to generate project cost and effort estimates.

Use one or more empirical models for software cost and effort estimation.

The first option, however attractive, is not practical

The second option can work reasonably well, if the current project is quite similar to the past efforts and other project influences are equivalent

Decomposing techniques take a divide and conquer approach to software project estimation

Empirical estimation models can be used to complement decomposition techniques and offer a potentially valuable estimation approach in their own right

Decomposition Techniques

Software project estimation is a form of problem solving

in most cases, the problem to be solved is too complex to be considered in one piece

The decomposition approach can be seen from two different points of view

Decomposition of the problem.

Decomposition of the process.

Software Sizing

Since the estimate of the project is only as good as the estimate of the size of the work to be accomplished, sizing represents the software project planner's first major challenge

If a direct approach is taken, size can be measured in LOC (lines of code).

If an indirect approach is chosen, size is represented as FP (function points).

Putnam and Myers's four different approaches to the sizing problem

Problem based Estimation

The project planner begins with a bounded statement of software scope

From this statement it attempts to decompose the software into various problem functions that can each be estimated individually

LOC or FP (the estimation variable) is then estimated for each function

When LOC is used as the estimation variable, decomposition is absolutely essential and is often taken to considerable levels of detail

For FP estimates, rather than focusing on function, each of the information domain characteristics as well as the 14 complexity adjustment values (discussed in the previous chapter) are estimated

The resultant estimates can then be used to derive a FP value that can be tied to past data and used to generate an estimate

Process-based Estimation

The most common technique for estimating a project is to base the estimate on the process that will be used

The process is decomposed into a relatively small set of tasks and the effort required to accomplish each task is estimated

It begins with a delineation of the software functions obtained from the project scope

A series of software activities must be performed for each function.

Once the problem functions and process activities are melded, the project planner estimates the effort (e.g., person-months) that will be required to accomplish each software for each software function

Cost and effort for each function and software process activity are computed as the last step

This estimation is performed independently of LOC or FP estimation

Empirical Estimation Models

The COCOMO Model

Software Equation

Make / Buy Decision

Outsourcing

Automated Estimation Tools

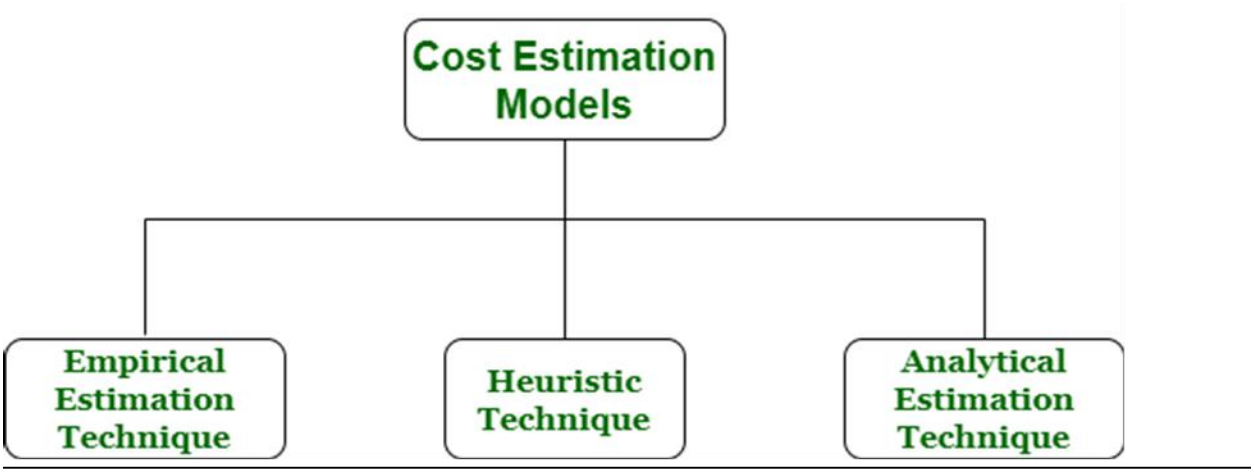
## Cost Estimation Models in Software Engineering

**Cost estimation** simply means a technique that is used to find out the cost estimates. The cost estimate is the financial spend that is done on the efforts to develop and test software in Software Engineering. Cost estimation models are some mathematical algorithms or parametric equations that are used to estimate the cost of a product or a project.
Various techniques or models are available for cost estimation, also known as Cost Estimation Models as shown below :

Empirical Estimation Technique –

Empirical estimation is a technique or model in which empirically derived formulas are used for predicting the data that are a required and essential part of the software project planning step. These techniques are usually based on the data that is collected previously from a project and also based on some guesses, prior experience with the development of similar types of projects, and assumptions. It uses the size of the software to estimate the effort.

In this technique, an educated guess of project parameters is made. Hence, these models are based on common sense. However, as there are many activities involved in empirical estimation techniques, this technique is formalized. For example Delphi technique and Expert Judgement technique.




**Staffing Management Plan**

Regardless of what you do in an organization, a staff is required in order to execute work tasks and activities. If you are a project manager, you need to have an adequate staff for executing your project activities.

Just having the required number of staff members for your project will not help you to successfully execute the project activities. These staff members selected for your project should have necessary skills to execute the project responsibilities as well. In addition, they should have the necessary motivation and availability as well.

Therefore, staffing_of your project should be done methodologically with a proper and accurate plan.

Understanding the Purpose

Before you start staffing your project, you need to understand the purpose of your project. First of all, you need to understand the business goals for the project and other related objectives. Without you being clear about the end results, you may not be able to staff the best resources for your project.

Spend some time brainstorming about your project purpose and then try to understand the related staffing requirements. Understand the different skills required for project execution, in order to understand what kind of staff you want.

Be Precise

Be precise when you prepare your staffing management plan. Make your staffing plan in black and white. Do not include things just to make the people happy. Always include the truth in your plan in a bold way. Whenever required, emphasize the roles and responsibilities of the staff and organizational policies as well.

The workforce should be disciplined in order to execute a project successfully. Therefore, you need to include discipline requirements to the staffing plan as well.

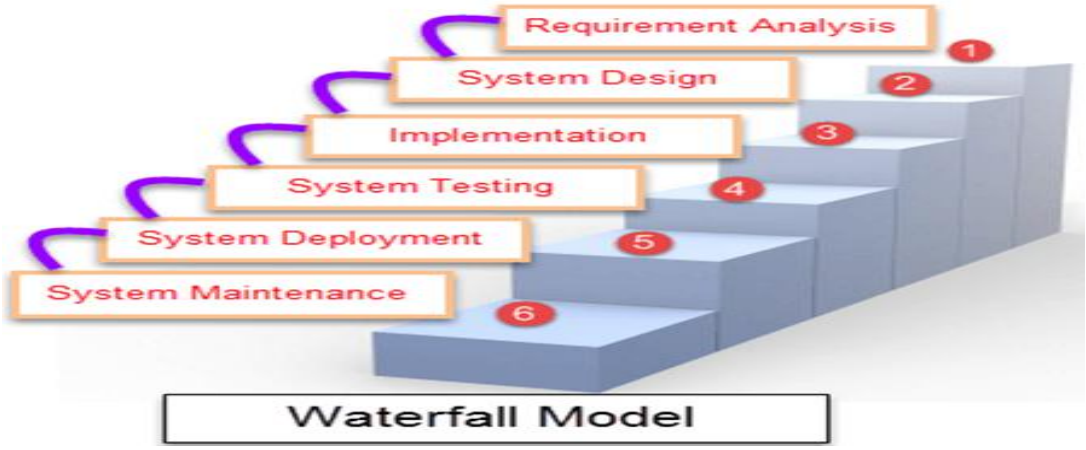
What is Software Engineering models?

A software process model is a simplified representation of a software process. Each model represents a process from a specific perspective. These generic models are abstractions of the process that can be used to explain different approaches to the software development. They can be adapted and extended to create more specific processes.

Predictive models are useful primarily because they give a lot of structure to a project. ... The chapter describes some predictive software development models such as waterfall model, waterfall with feedback, sashimi model, incremental waterfall model, V-model and software development life cycle

What is The Waterfall Model?

WATERFALL MODEL is a sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases. Each phase is designed for performing specific activity during the SDLC phase. It was introduced in 1970 by Winston Royce.




When to use SDLC Waterfall Model?

Waterfall model can be used when

Requirements are not changing frequently

Application is not complicated and big

Project is short

Requirement is clear

Environment is stable

Technology and tools used are not dynamic and is stable

Resources are available and trained

Waterfall Model advantage and Disadvantages

| Advantages | Dis-Advantages |
|---|---|
| Before the next phase of development, each phase must be completed | Error can be fixed only during the phase |
| Suited for smaller projects where requirements are well defined | It is not desirable for complex project where requirement changes frequently |
| They should perform quality assurance test (Verification and Validation) before completing each stage | Testing period comes quite late in the developmental process |
| Elaborate documentation is done at every phase of the software's development cycle | Documentation occupies a lot of time of developers and testers |
| Project is completely dependent on project team with minimum client intervention | Clients valuable feedback cannot be included with ongoing development phase |
| Any changes in software is made during the process of the development | Small changes or errors that arise in the completed software may cause a lot of problems |

Some situations where the use of Waterfall model is most appropriate are

Requirements are very well documented, clear and fixed.

Product definition is stable.

Technology is understood and is not dynamic.

There are no ambiguous requirements.

Ample resources with required expertise are available to support the product.

The project is short.

## Requirements Elicitation

Requirements elicitation is perhaps the most difficult, most error-prone and most communication intensive software development. It can be successful only through an effective customer-developer partnership. It is needed to know what the users really need.

There are a number of requirements elicitation methods. Few of them are listed below –

Interviews

Brainstorming Sessions

Facilitated Application Specification Technique (FAST)

Quality Function Deployment (QFD)

Use Case Approach

The success of an elicitation technique used depends on the maturity of the analyst, developers, users, and the customer involved.

**1. Interviews:**

Objective of conducting an interview is to understand the customer's expectations from the software.
It is impossible to interview every stakeholder hence representatives from groups are selected based on their expertise and credibility.
Interviews maybe be open ended or structured.

1. In open ended interviews there is no pre-set agenda. Context free questions may be asked to understand the problem.
2. In structured interview, agenda of fairly open questions is prepared. Sometimes a proper questionnaire is designed for the interview.

**2. Brainstorming Sessions:**

It is a group technique
It is intended to generate lots of new ideas hence providing a platform to share views
A highly trained facilitator is required to handle group bias and group conflicts.

Every idea is documented so that everyone can see it.
Finally a document is prepared which consists of the list of requirements and their priority if possible.

**3. Facilitated Application Specification Technique:**

It's objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.
A team oriented approach is developed for requirements gathering.
Each attendee is asked to make a list of objects that are-

1. Part of the environment that surrounds the system
2. Produced by the system
3. Used by the system

Each participant prepares his/her list, different lists are then combined, redundant entries are eliminated, team is divided into smaller sub-teams to develop mini-specifications and finally a draft of specifications is written down using all the inputs from the meeting.

**4. Quality Function Deployment:**

In this technique customer satisfaction is of prime concern, hence it emphasizes on the requirements which are valuable to the customer.
3 types of requirements are identified –

**Normal requirements –** In this the objective and goals of the proposed software are discussed with the customer. Example – normal requirements for a result management system may be entry of marks, calculation of results, etc
**Expected requirements –** These requirements are so obvious that the customer need not explicitly state them. Example – protection from unauthorized access.
**Exciting requirements –** It includes features that are beyond customer's expectations and prove to be very satisfying when present. Example – when unauthorized access is detected, it should backup and shutdown all processes.

The major steps involved in this procedure are –

1. Identify all the stakeholders, eg. Users, developers, customers etc
2. List out all requirements from customer.
3. A value indicating degree of importance is assigned to each requirement.
4. In the end the final list of requirements is categorized as –
   It is possible to achieve
   It should be deferred and the reason for it
   It is impossible to achieve and should be dropped off

**5. Use Case Approach:**

This technique combines text and pictures to provide a better understanding of the requirements.
The use cases describe the 'what', of a system and not 'how'. Hence they only give a functional view of the system.
The components of the use case design includes three major things – Actor, Use cases, use case diagram.

1. **Actor –** It is the external agent that lies outside the system but interacts with it in some way. An actor maybe a person, machine etc. It is represented as a stick figure. Actors can be primary actors or secondary actors.

Primary actors – It requires assistance from the system to achieve a goal.
Secondary actor – It is an actor from which the system needs assistance.

2. **Use cases –** They describe the sequence of interactions between actors and the system. They capture who(actors) do what(interaction) with the system. A complete set of use cases specifies all possible ways to use the system.
3. **Use case diagram –** A use case diagram graphically represents what happens when an actor interacts with a system. It captures the functional aspect of the system.
   A stick figure is used to represent an actor.
   An oval is used to represent a use case.
   A line is used to represent a relationship between an actor and a use case.