
Advanced Software Engineering **(CS6401)**

Autumn Semester (2022-2023)

Dr. Judhistir Mahapatro
Department of Computer Science and
Engineering
National Institute of Technology Rourkela

Software Life Cycle Models

(Lecture-3)

Topics covered in Previous Lectures:

- ▶ Nature of software
- ▶ Nature of software projects
- ▶ What is Software Engineering?
- ▶ Programs vs. Software Products
- ▶ Software Process
- ▶ Introduction to Life Cycle Models

Software Life Cycle Models

- Let us review the main steps
 - Problem Definition (define the problem)
 - Feasibility study (establish **cost effectiveness** of the solution we proposed to)
 - Analysis (try to clearly define **the responsibility or functions** that the software must undertake)
 - System Design
 - Detailed Design
 - Implementation or Coding
 - Maintenance

Software Life Cycle Models

- A separate planning step for large applications may be introduced after feasibility (or part of feasibility)
- Study the outputs of the steps

Problem Definition

- To answer: What is the Problem?
- Where and by whom is the problem felt?
 - Where in the organization particularly the problem has been felt
- Meet users and management and obtain their agreement that there is a problem

Problem Definition

- If problem exists, and it need to be resolved
 - It becomes a project
 - Commitment of funds implied

The objective is to clearly define goal for the project and establish as a project

Problem Definition

- Prepares a brief statement of problem (small document not extensive but important, it is the first deliverable)
 - Avoids misunderstandings
 - Get concurrence from user/management
 - Usually short: 1 or 2 pages
- Estimate cost and schedule for the next feasibility step

Problem Definition

- Estimate **roughly overall project cost** to give users a sense of project scope. The estimates become more **refined** in later steps.
 - misconception by the user that the **project will be done with the cost** in his mind
 - roughly **estimated cost is acceptable** by the user
 - it is preliminary and based on the **experience of analyst or computer experts**

Problem Definition

- This step is short; lasts a day or two
 - do not involve cost and we just **invite an expert from the development industry** to make him understand the problem and give a scope
- Proper understanding and characterization of problem essential
 - To discover cause of the problem
 - To plan directed investigation
 - Else, **success** is **unlikely** (end up solving a wrong problem and user may not accept)

Problem Definition

- Possible initial characterization of problems
 - Existing system has **poor response time**, i.e., user is unable to process a transaction or take too large time.
 - Unable to **handle workload** (so many transactions or users making transactions at same time and long queue of users)
 - Problem of cost: existing system uneconomical
 - Problem of accuracy and reliability
 - Requisite information is not produced by system
 - Problem of security

Ex: Railways Reservation System

Problem Definition Document

- Short document called problem statement document
 - **Project Title**
 - **Problem Statement:** Concise statement of problem, possibly in a few lines
 - **Project Objectives:** state objective of the project defined for the problem
 - **Preliminary Ideas discussion with the user or past experience:** possible solutions, if any, occurring to user and/or analyst could be stated here
 - **Project Scope:** give overall cost estimate as rough figure
 - **Feasibility Study:** indicate time and cost for the next step

Problem Definition Document

Note:

- Do not confuse between problems and solutions e.g., ‘develop computerized payroll’ cannot be a problem
- No commitment is implied to preliminary ideas (may explore many ideas during analysis)

Feasibility Study

- To get better understanding of problems and reasons by studying existing system, if available
 - Are there feasible solutions?
 - Is the problem worth solving?
- Consider different alternatives
- Estimate costs and benefits for each alternative
- Essentially covers other steps of methodology (analysis, design, etc.) in a capsule form

Feasibility Study

- Make a formal report and present it to management and users; review here confirms the following:
 - Will alternatives be acceptable
 - Are we solving the right problem
 - Does any solution promise a significant return (because investing on the project)
- Users/management select an alternative
- Many projects 'die' here (alternatives are not acceptable or too high cost)

Types of Feasibility

- **Economical**
 - will returns justify the investment in the project?
- **Technical**
 - is technology available to implement the alternative?
- **Operational**
 - will it be operationally feasible of the solution as per rules, regulations, laws, organization culture, union agreements, etc.?

Costs

- One-time(initial) costs include equipment, training, software development, consultation, site preparation
- Recurring costs include salaries, supplies, maintenance, rentals, depreciation (finally there will be replacement cost)
- Fixed and variable costs; vary with volume of workload

Benefits

- Benefits could be tangible (i.e., quantifiable) or intangible (which can not be measured)
- Saving (tangible benefits) could include
 - saving in salaries
 - saving in material or inventory costs
 - more production
 - Reduction in operational costs, etc.

Benefits

- Intangible benefits may include
 - Improved customer service
 - Improved resource utilization
 - Better control over activities (such as production, inventory, finances, etc.)
 - Reduction in errors
 - Ability to handle more workload

Estimating Costs

- How to estimate costs so early in the project?
 - Decompose the system and estimate costs of components; this is easier and more accurate than directly estimating cost for the whole system
 - computer cost or networking cost which is separate from the software development cost
 - Use historical data whenever available

Estimating Costs (cont..)

- Use organizations standards for computing overhead
- costs (managerial/secretarial support, space, electricity, etc)
- Personal (for development and operations) costs are function of time, hence estimate time first

Financial Analysis

- Consider time-value of money; while investment is today, benefits are in future
- Compute present value P for future benefit F by
$$P = F/(1+L)^n$$
where L is the prevailing interest rate and n is year of benefit
- Take into account life of system: most systems have life of 5-7 years of course this is also less because of technology advancements

Financial Analysis

- Cost is 'investment' in the project, benefits represent 'return'
- Compute payback period in which we recover initial investment through accumulated benefits
- Payback period is expected to be less than system life.

Feasibility study report

- **Introduction**

- A brief statement of the problem, the environment in which the system is to be implemented, and constraints that affect the project (cost, effort, available resources)

- **Management Summary and Recommendations**

- Important findings and recommendations

Feasibility study report

- **Alternatives:**
 - A presentation of alternative system specifications; criteria that were used in selecting the final approach
- **System Description**
 - An abbreviated version of information contained in the System-Specification or reference to the specifications

Feasibility study report

- Cost-Benefit Analysis
- Evaluation of Technical Risk
- Legal Ramification (if any)
 - Once the estimation is accepted by the user/management then the clear signal for the go-ahead with the project

Requirement Analysis

- **Objective:** determine what the system must do to solve the problem (without describing how)
- Done by Analyst (also called Requirements Analyst)
- Produce Software Requirement Specifications (SRS) document
- Incorrect, incomplete, inconsistent, ambiguous SRS often cause for project failures and disputes

Requirement Analysis

- A very challenging task
 - Users may not know exactly what is needed or how computers can bring further value to what is being done today
 - Users change their mind over time
 - They may have conflicting demands
 - They can't differentiate between what is possible and cost-effective against that is impractical (wish-list)
 - Analyst has no or limited domain knowledge
 - Often client is different from the users

SRS

- SRS is basis for subsequent design and implementation
- First and most important baseline
 - Defines contract with users
 - Basis for validation and acceptance
- Cost increases rapidly after this step; defects not captured here become 2 to 25 times more costly to remove later (defects which entered in the design and implementation which will be more costlier to remove it)

SRS

- It identifies all functional (inputs, outputs, processing) and performance requirements, and also other important constraints (legal, social, operational)

SRS

- Should be adequately detailed so that
 - Users can visualize what they will get
 - Design and implementation can be carried out
- Covers what and what at business level; e.g.,
 - What calculate take-home pay
 - How: procedure (allowances, deductions, taxes etc.)

Analysis Process

- Interviewing clients and users essential to understand their needs from the system
- Often existing documents and current mode of operations can be studied (they may have documents explaining procedures)

Analysis Process

- Long process: needs to be organized systematically
 - Interviewing, correlating, Identifying gaps, and iterating again for more details
 - Focus on what gets done or needs to be done
 - Focus on business entities, their interactions, business events, ...
- Identify users and important business entities
- Get functional (domain) knowledge

Analysis Process

- Interview users or get details through questionnaires
- Examine existing system:
 - Study existing forms, outputs, records kept (files, ledgers, computerized systems)
- Often goes outside in:
 - what outputs needed, which inputs provide data, what processing done, what records kept, how records updated (i.e., go inwards from system boundaries)

Interviews

- Identify users, their roles and plan interviews in proper order to collect details progressively and systematically
- Conducting interviews is an art
 - Workout scope, durations, purpose
 - Keep records and verify/confirm details it with the user
 - Needs to sometimes 'prompt' users in visualizing requirements
- Need good communication skills, domain knowledge, patience, etc

Organizing Findings

- Massive amount of information is collected from interviews, study of existing systems
- Need to be organized, recorded, classified and conceptualized (at multiple level of details)
- Tools/repositories available (describe inputs, outputs, files, computations, usages, functions):
 - Help in checking consistency and completeness of the work done so far

Organizing Findings

- Create **models** or **projections** from different perspectives (in order to systematically organize the idea found out from the process)
 - Way to handle complexity (divide-and-conquer)
 - Hide unnecessary details (focus on certain specific aspects and hide unnecessary data)
- Reduces errors, ensures consistency/completeness
- Data-flow diagrams (for processing) - what data is being used in which step, entity-relationship models (for data domain) and object models commonly used

System Design (address 'how part')

- **Objective:** To formulate alternatives about how the problem should be solved
 - Input is SRS from previous step
 - Consider several technical alternatives based on type of technology, automation boundaries, type of solutions (batch/on-line), including make or buy (Before business alternatives now here technical alternatives)
 - Propose a range of alternatives: low-cost, medium cost and comprehensive high cost solutions
-



Alternatives

- For each alternatives, prepare high-level system design (in terms of architecture, DB design, ...); prepare implementation schedule, carry out cost-benefit analysis



Alternatives

- Prepare for technical (ensure different **technological alternatives considered are meaningful**) and management review (ensure that we are within the proposed cost and able to meet the schedule)
 - Costs rise sharply hereafter
 - Costs can be quantified better at this stage
 - Technical review uncovers errors, checks consistency, completeness, alternatives
- Phase ends with a clear choice which can be further taken into design and implementation phase.



Design Goals

- Processing component: main alternatives
 - Hierarchical modular structure in functional approach (conventional methodology)
 - Object-oriented model and implementation
- Different design methodologies for functional and Object Oriented



Design Goals

- Data component
 - Normalized data base design using ER model (conceptual design)
 - De-normalization for performance (modify the conceptual normalized data into the data base design to improve the performance)
 - Physical design: Indexes (choosing right storage technique through which data can be accessed efficiently)
 - Design of software consisting of designing the processing component and data component. They may be designed separately or they may merge into single design dimension when we use object oriented
-

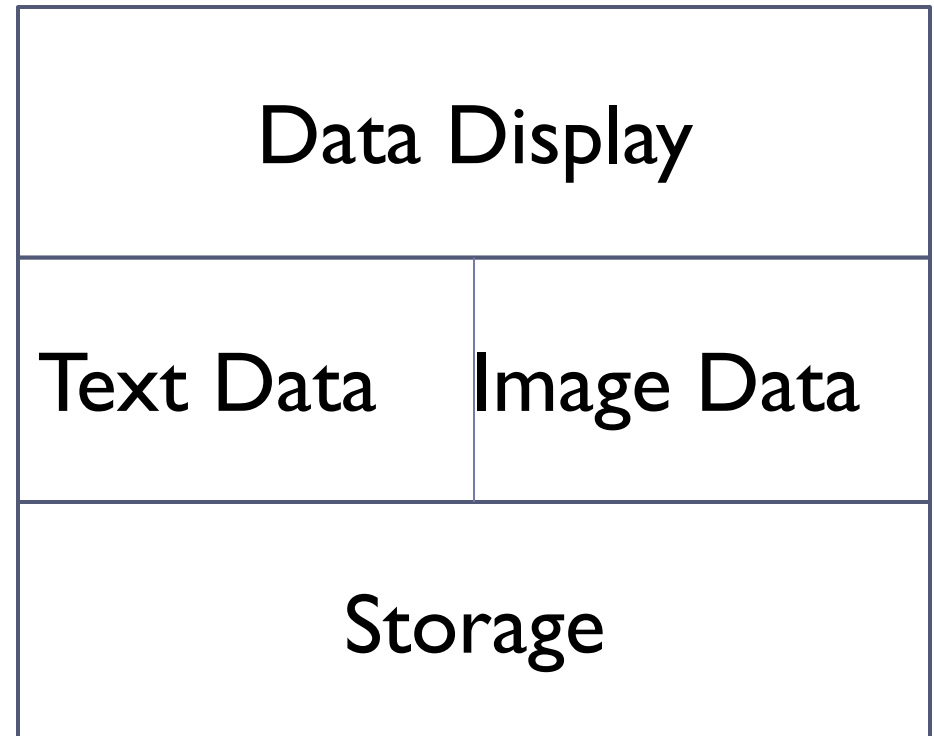
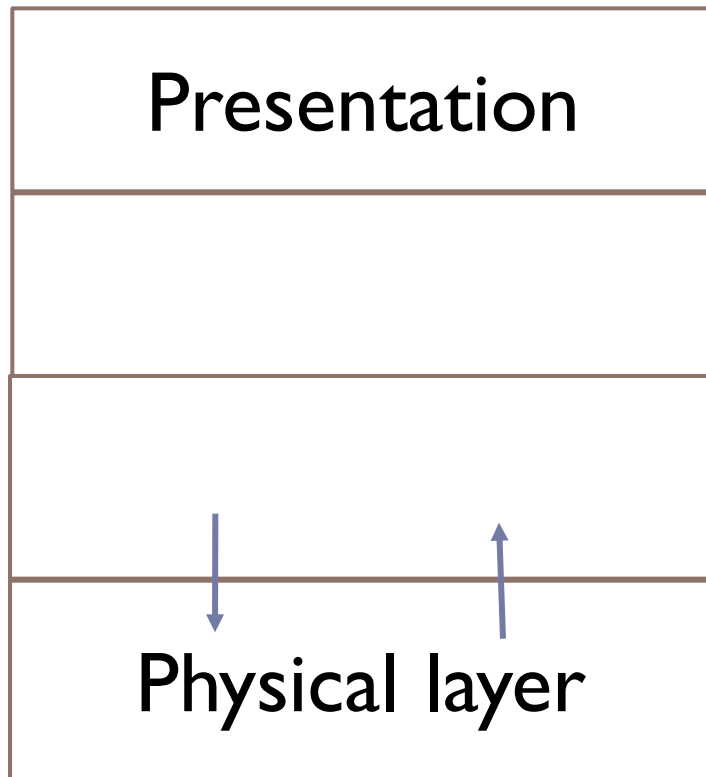


System Architecture

- Decompose a complex system:
 - Partitions (vertical)
 - Layers (horizontal)
- Each layer and partition is given specific responsibility
- Define subsystems/modules as building blocks
 - Each module has a specific function to perform (piece of code)
 - Modules together may make a subsystem
 - Multiple such subsystems make up a overall system
 - subsystem may be representing a partition or layer
- Modules make calls on each other
 - Pass data, obtain results



System Architecture



System Architecture

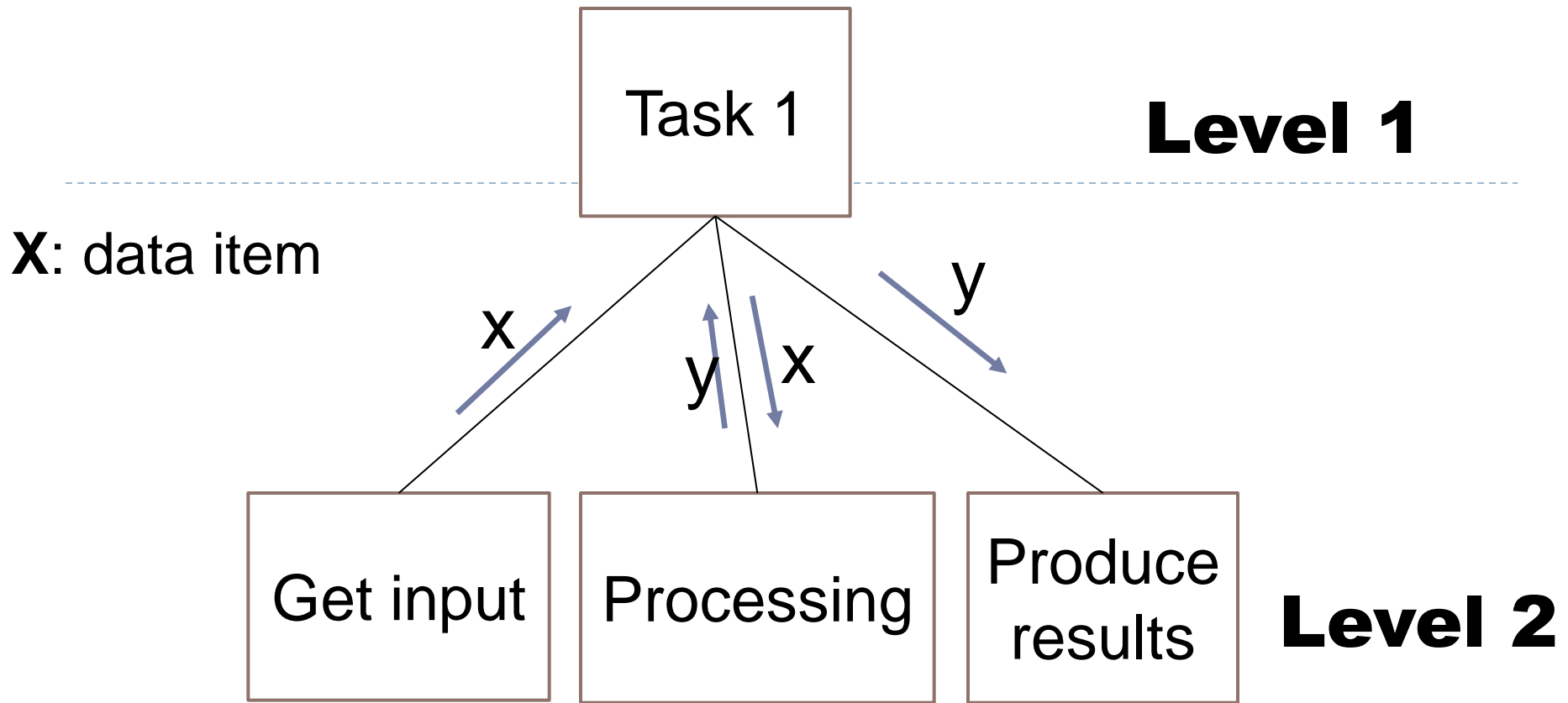
- Maximize module independence and minimize module interdependence (from maintenance point of view and complex system)
 - Cohesion and coupling characteristics
 - Essential for maintenance (a module can be replaced with an equivalent module without disturbing overall functioning)



Structure Chart Notation

- Software Architecture can be defined in terms of modules and their interactions can be captured through a Diagram Notation tool called **Structure Chart**
- Used in functional methodology to depict modules and their calling relationships
- Techniques are available to go from DFD to structure charts
- Hierarchical structure: module at level i calls modules at level $i+1$; control flow not shown





- **Lines** representing the calls that the module Task1 makes to Get input/Processing/Produce results in order to perform its own job
- Modules on level 2 can be decomposed further

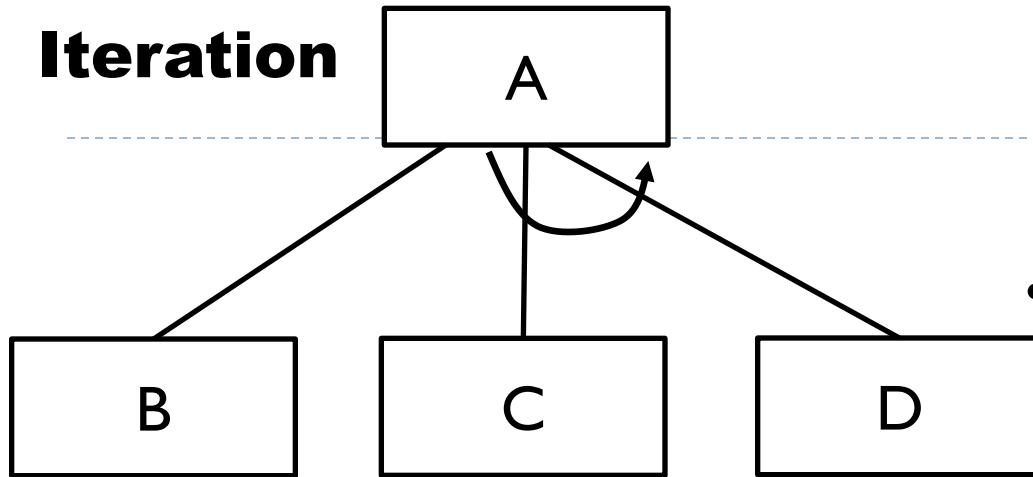


Structure Chart ...

- Modules at higher levels generally do coordination and control; modules at lower levels do i/o and computations
- Structure chart may show important data passing between modules, and also show main iterations and decision-making without much details

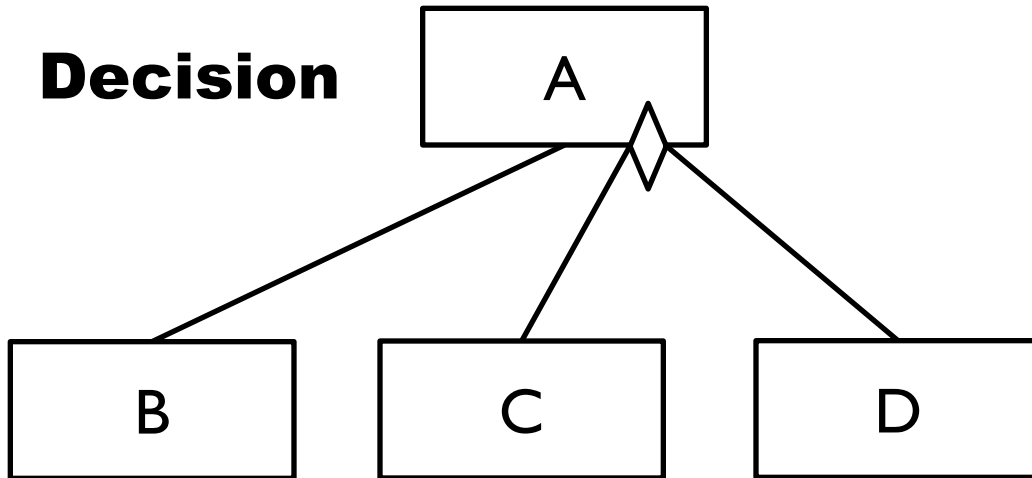


Iteration



- Module A calls C and D module repeatedly, however, the details of how often the iteration would be done are not given on the structure chart
- It only shows a such a iteration or decision exists in the software

Decision



OO Approach

- Design consists of classes
 - Have structure (properties)
 - Have behavior (methods/operations)
 - Inheritance major feature in OO for re-use
- Data and executable functions associated with the class (these are called structure and behavior for the class)
- This is a paradigm that combines data and processing together on a single dimension and identifies classes which have structural properties and behavioral methods defined for them



OO Approach

- Class diagrams show static structure of the system
- Interaction diagrams are used to capture dynamic behavior of classes and objects
- Large systems decomposed into packages



Design Document Format

- Introduction
- Problem Specification: include here the data-flow diagrams, entry-relationship diagrams or class diagrams
- Software structure: give the high-level software structure chart identifying major modules and major data elements in their interfaces



Design Document Format

- Data Definitions: for major data structure, files and database
- Module Specifications: Indicate inputs, outputs, purpose and subordinates modules for each software modules
- Requirements Tracing: Indicates which modules meet which requirements



Design document format...

- This document will be reviewed by the technical people and ensured that this document is comprehensive
- Ensures that it covers all the functions identified in the SRS document



Detailed Design

- Specific implementation alternative already selected in previous step giving
 - Overall software structure
 - Modules to be coded
 - Database/file design
- In this step, each component is defined further for implementation



Detailed Design...

- Deliverables include
 - Program Specifications (e.g. pseudo-code)
 - File design (organization, access method...)
 - Hardware specifications (as applicable)
 - Test plans
 - Implementation schedule
- Ends in technical review



Implementation Phase

- Programs are coded, debugged and documented
- Initial creation of data files and their verification (manual existing data is converted to the data files and database)
- Individual modules as well as whole system is tested
- Operating procedures are designed
- User does acceptance of the system
- System is installed and switch-over affected



Operations and Maintenance

- Systems must continue to serve user needs correctly and continuously
- Maintenance activities consist of
 - Removing errors
 - Extending present functions
 - Adding new functions
 - Porting to new platforms (occasionally it may be)



Summary

- Each phase has a well defined task and a deliverable
- Feasibility establishes alternatives and carries out cost-benefit analysis
- Requirements analysis is very challenging and SRS forms the first baseline
- Design step consists of architecture, database and interface design



Summary

- Adherence to a software life cycle model:
 - helps to do various development activities in a systematic and disciplined manner.
 - also makes it easier to manage a software development effort.

Reference

- ▶ R. S. Pressman, *Software Engineering A Practitioner's Approach*, McGraw Hill Publications , 2006
- ▶ R. Mall, *Fundamentals of Software Engineering*, Prentice Hall of India , 2014
- ▶ I. Sommerville, *Software Engineering*, Pearson Education, Asia , 2006
- ▶ P. Jalote, *An Integrated Approach to Software Engineering*, Narosa , 2006