Page No.
Date

Name : Kunal Bandale

Subject : Software Engineering

Ref. : Gate Smashers (Youtube)

# Software Engg (IMP Topics)

(1) Introduction
** (2) Software Development Life Cycle
(3) Requirement analysis (SRS)
** (4) Software Project Management (COCOMO)
(5) Software Design (Coupling, Cohension, UML, DFD, class Dia)
(6) Coding and Testing **
(7) ~~Quality~~ Maintenance
(8) Quality Mang. Reuse

## Soft. Engg Defn and Evolution

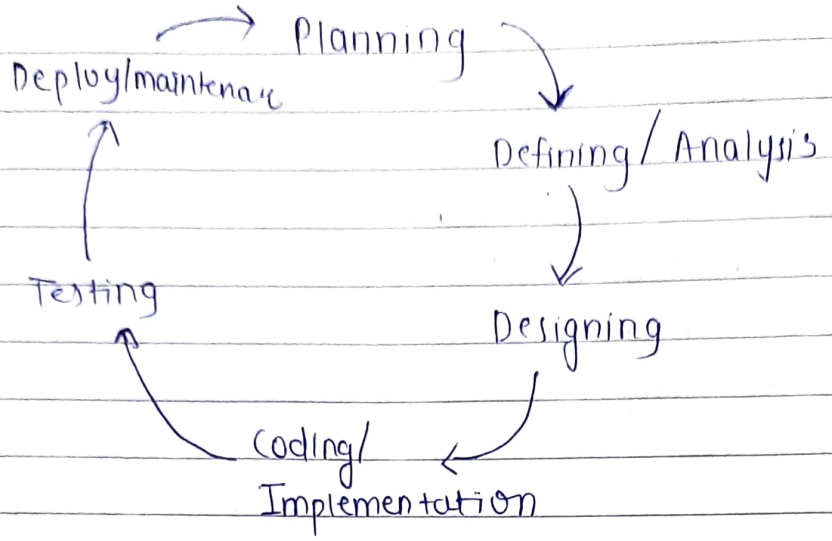
* It is systematic, displined, cost effective techniques for software development.

* Engg Approach to develop a software.

### Evolution

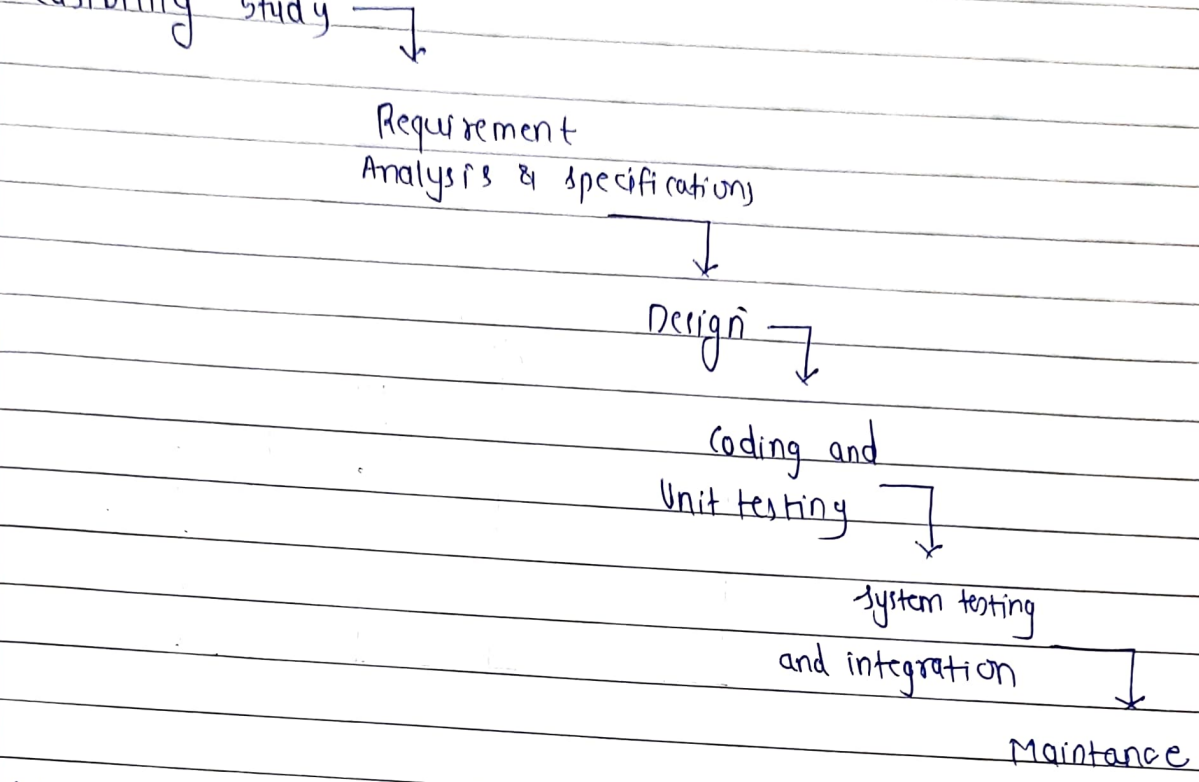
1945-65 → Origin
1965-85 → Crisis* 100 → 50 (2)
1990-2000 → Internet
2000-2010 → Light weight [Software like mobile Phone etc]
2010 → AI/ML/DL

# Software Development Life Cycle (SDLC)




(1) Planning - Communicate / Draft
(2) Defining - Estimation, Analysis, Cost, time, charges, Negotiation. (SRS) formal written document
(3) Designing - Modules, Coupling, DFDs,
(4) Coding / Implementation
(5) Testing
(6) Deploy

# Classical Waterfall Model (1970s)



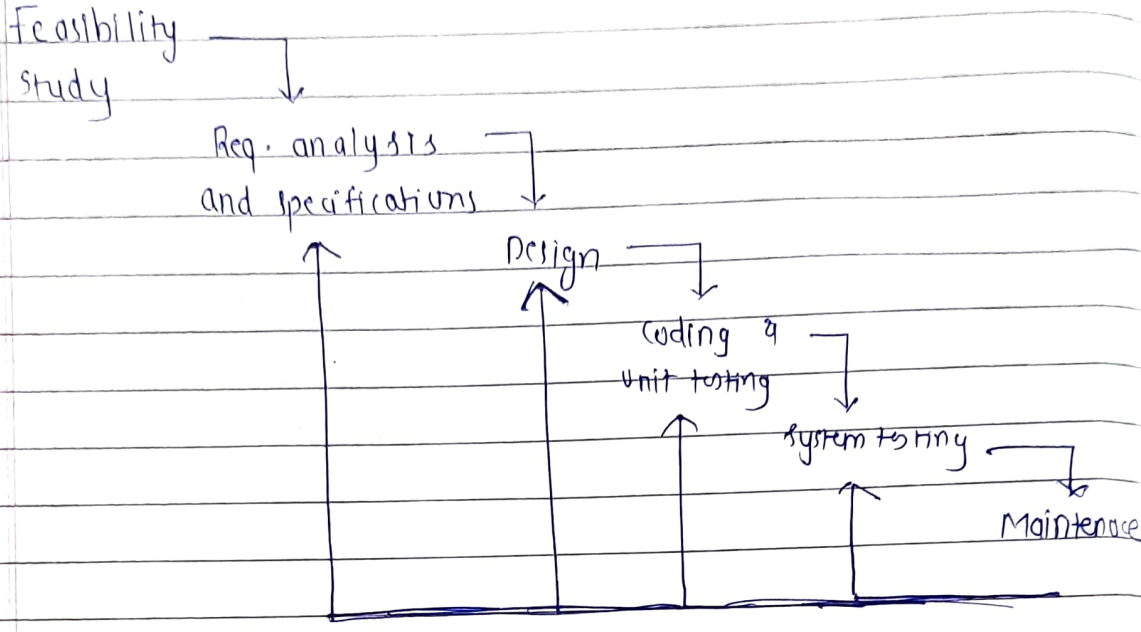

## Advantages

* Base Model
* Simple and Easy
* small projects

## Disadvantages

* NO feedback
* NO Experiment
* NO parallerism
* High Risk
* 60% Efforts Maintenance

# Iterative Waterfall Model

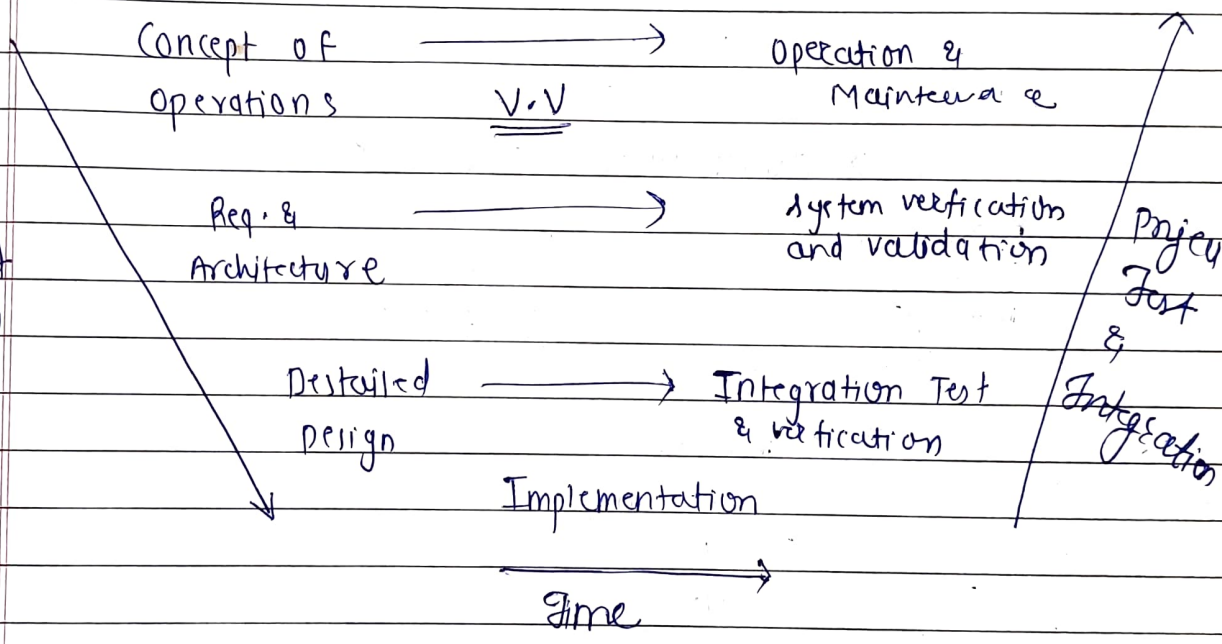



* **Advantages**
  - Base Model
  - Simple and Easy
  - Small projects
  - Feedbacks

* **DisAdvantages**
  - No phase overlapping
  - No intermediate Delivery
  - Rigid (No changes)
  - Less Customer interaction

# V-shaped Model

- Also known as verification and validation Model
- Extension of Waterfall Model.
- Testing is associated with every phase of lifecycle.
- Verification Phase (Req. analysis, System design, Architecture design, Module design)
- Validation phase (Unit testing, Integration System, Acceptance Testing)




- Advantages
  - Time saving
  - Good understanding of project in the beginning
  - Every component must be testable
  - Process can be tracked easily
  - Proactive defect tracking

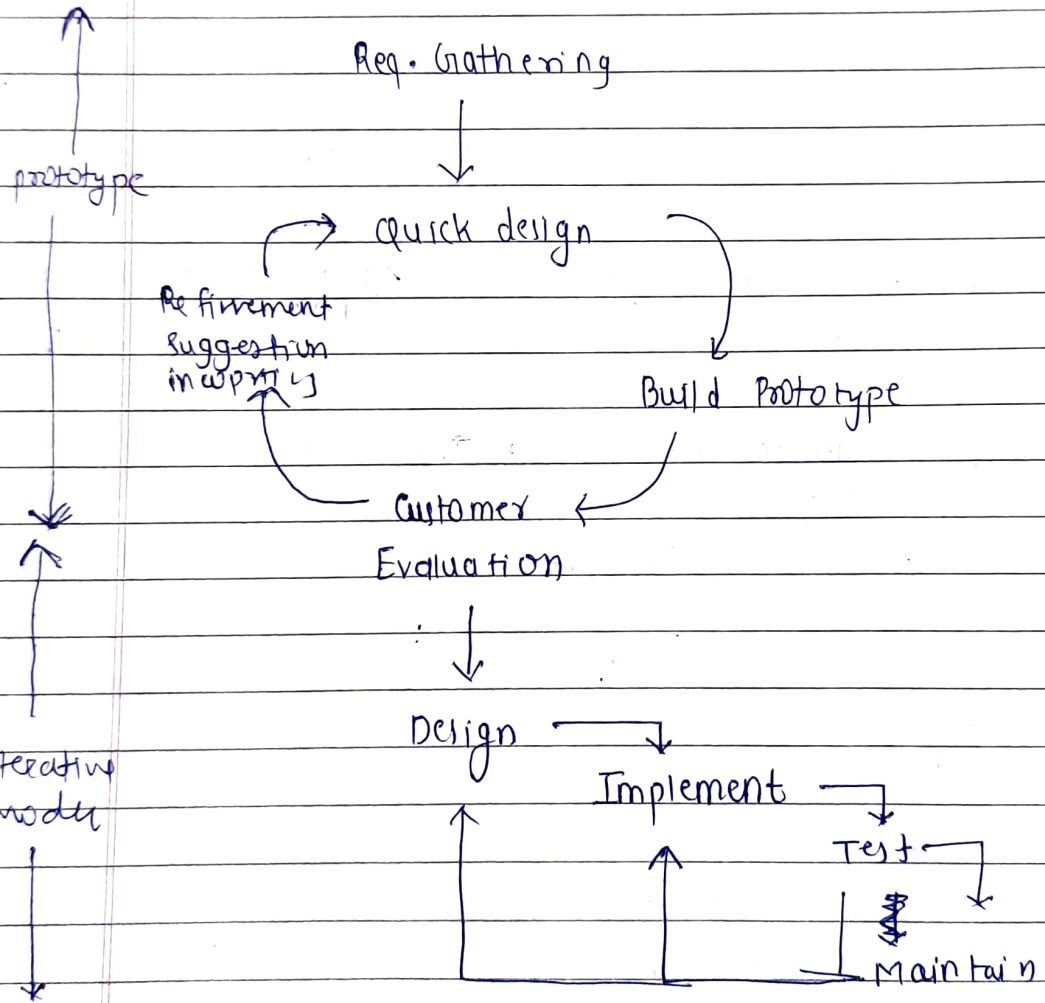
Page No. 06
Date

## Disadvantages

- No feedback so less scope of changes
- Risk analysis not done
- Not good for big or object-oriented projects

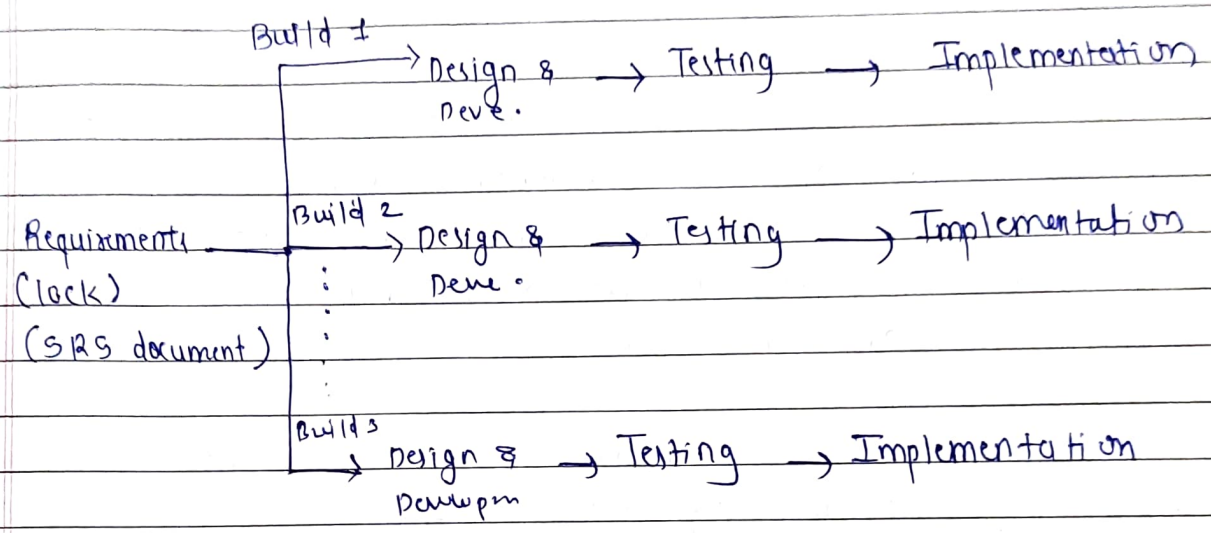
## Prototyping Model in Software Engg

↳ kind of dummy model / kind of structure



* Customer not clear with idea
* Throw away Model
* good for technical requirement risks
* Increase in cost of development

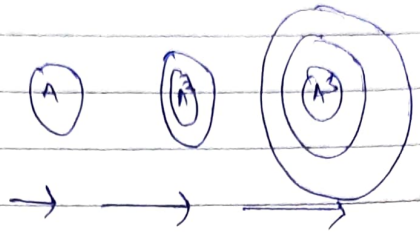
## Incremental Model




* Module by Module working
* Customer Interaction Maximum
* Large projects
* Early release product demand
* Flexible to changes

# Evolutionary Model

- Evolutionary model is a combination of iterative and incremental model of software development life cycle.




- Incremental model first implement a few basic features and deliver to the customer. Then build the next part and deliver it again and repeat this step until the desired system is fully realized. No long-term plans are made.

- Iterative model main advantage is its feedback process in every phase.

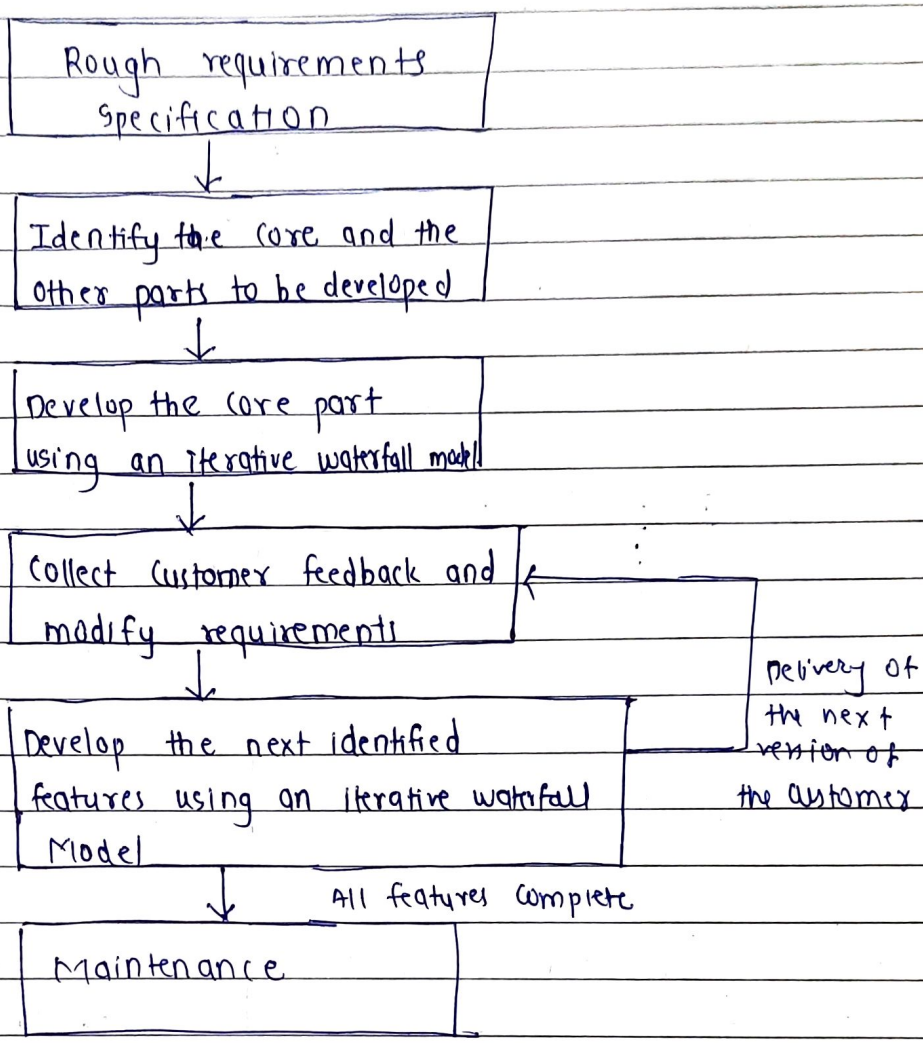
- Also known as "Design a little, Built a little, test a little, deploy a little mode".

## Advantages

* Customer requirements are clearly specified.
* Risk analysis is better.
* It supports changing environment
* Initial operating for large mission-critical projects

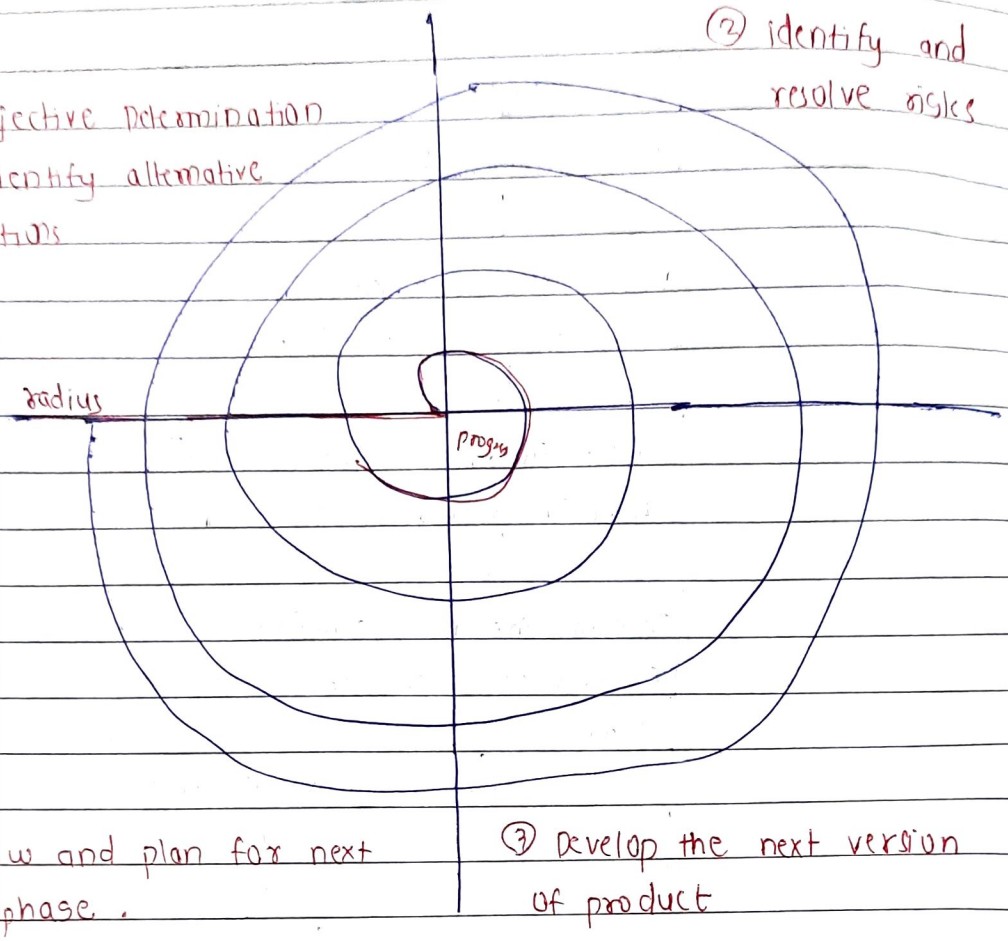
## DisAdvantages

* NO suitable for smaller projects.
* cost
* Highly skilled resources are required




## Evolutionary Model

# Spiral Model




* Risks Handling
* Radius of spiral = Cost
* Angular Dimension = Progress
* Meta Model

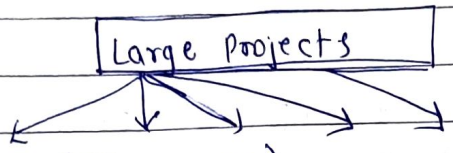
* **Advantages**
(1) Risk handling
(2) Large projects
(3) flexible
(4) Customer satisfaction

* **Disadvantages**
(1) Complex
(2) Expensive
(3) Too much risk analysis
(4) Time

# Agile in Software Engineering

* Used in Major companies

'Agile' (Move quickly)




Small (Iterations)
chunks
↓
Release
↓
feedback
↓
Enhance
↓
Rerelease

* Breaking Large project in Small chunks
* Release in the market
* Taking feedback from the customer
* Based on feedback product is Enhance and
* It is Released

---

* Advantages
(1) frequent delivery
(2) face to face communication with client
(3) changes
(4) Time

* Disadvantages
(1) Less documentation
(2) Maintenece Problem

Page No. 12
Date

# SCRUM Model

* One of the most popular agile methodology
* Scrum is a lightweight iterative and incremental framework.
* Scrum breaks down the development phases into stages or cycles called "sprints".
* The development time for each sprint is maximized and dedicated, thereby managing only one sprint at a time.
* Scrum Team has Scrum master and product owner with constant communications on the daily basis.
* Keywords:- backlog, Sprint, Daily scrum, Scrum master, Product owner.

* Backlog (form, place of req everything is designed)
* Daily Scrum (Daily 10-15 minutes meeting)
* Scrum master (Monitoring (Handling team)
* product owner (client)

* Advantages

(1) Freedom & Adaption
(2) High quality, low risk product
(3) Reduce the development time up to 40%.
(4) Scrum customer satisfaction is very important.
(5) Reviewing the current sprint before moving to new one

## Disadvantages:

* More efficient for small team size
* No changes in the sprint.

## Comparison of All SDLC Models

| classical waterfall Model | Iterative Model | Prototype Model | Incremental Model |
|---|---|---|---|
| Basic, Rigid, Inflexible, Not for real project | Basic, Problem is well understood | User Req. not clear, Costly, No early look for requirements, High user Involvement, Reusability | Module by Module Delivery, Easy to test and debug |
| **RAD Model** | **Spiral Model** | **Evolutionary** | **Agile Model** |
| Time and Cost Constraint, User at all levels, Reusability | Risk, Not for small projects, No early look on requirement, less Experiment can work | Large projects | Flexible, Advanced, Parallel, Process into sprint |

## Software Requirements

- It is the description of features and functionalities of the target system.
- It is the description of what the system should do
- Requirement Engineering (RE) refers to the process of defining documenting, and maintaining requirements in the engineering design process.
- It is a four step process, which includes —

(1) feasibility Study

(2) Requirement Gathering / Elicitation

★ (3) Software Requirement Specification

(4) Software Requirement Validation

## Tool Support for Requirements Engineering

(1) observation reports (user observation)
(2) Questionnaires (interviews, surveys and polls)
(3) Use cases
(4) Used stories
(5) Requirement workshops
(6) Mind Mapping
(7) Role - Playing
(8) Prototyping

## Functional Vs Non - Functional Requirements

Functional :-
Requirements, which are related to functional / working aspects software fall into this category

Non - Functional :-
Requirements are expected characteristics of target software (Security, Storage, Configuration, recovery, Accessibility, Disaster recovery)

# Software Requirements Specification (SRS)

- SRS is a description of a software system to be developed.
- It lays out functional and non-functional requirements of the software to be developed.
- It may include part of use cases that describe user interactions that the software must provide to the user for perfect interaction.

## SRS Structure

1. Introduction
   - 1.1 Purpose
   - 1.2 Intended Audience
   - 1.3 Scope
   - 1.4 Definitions
   - 1.5 References
2. Overall Description
   - 2.1 User interface
   - 2.2 System interfaces
   - 2.3 Constraints, assumptions and dependencies
   - 2.4 User characteristics

3. System Features and Requirements
   3.1. Functional Requirements
   3.2. Use cases
   3.3. External Interface Requirements
   3.4. Logical database requirements
   3.5. Non functional Requirements

4. Deliver for Approval

User Requirements
↳ End User

- Easy and simple to Operate
- Quick Response
- Effectively Handling Operational error
- Customer Support

User Requirement Specification (URD)

- The user requirement(s) document (URD) or user requirement specification (URS) is a document usually used in software engineering that specifies what the user expects the software to be able to do.
- It is a contractual agreement.

# Data Flow Diagrams (DFD) → Bubble chart

- A graphical tool useful for communicating with users, managers, and other personnel.
- Useful for analyzing existing as well as proposed systems.

## Why DFD?

provides an overview of :-
- What data system processes
- What transformations are performed
- What data are stored
- What result are produced
- Graphical nature makes it a good communication tool between -
  (1) User and analyst
  (2) Analyst and system designer

## ★★ DFD elements

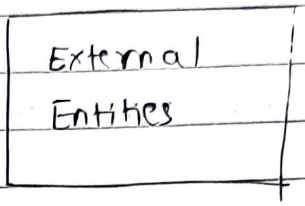
Source / Sinks (External entities) 
Data Flows 
Processes 
Data stores 
(DB)

## External Entities

Source :- Entity that supplies data to the system.

Sink :- Entity that receives data from the system.

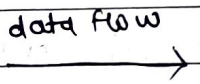
- A rectangle represents an external entity




- They either supply or receive data
- They do not process data

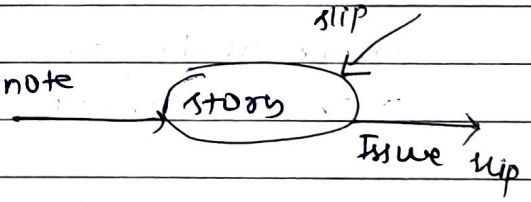
## Data Flows

- Marks movement of data through the system - a pipeline to carry data.



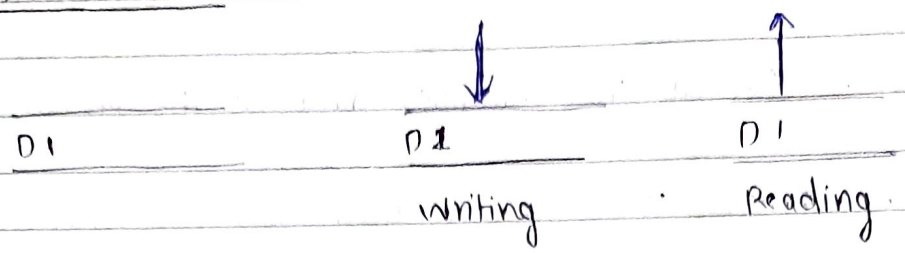

- Connects the processes, external entities and data stores.
- Generally unidirectional. If same data flows in both directions, double-headed arrow can be used.

## Processes




- A circle represents a process
- Straight line with incoming arrows are input data flows
- Straight lines with outgoing arrows are output data flows.
- Labels are assigned to Data flow.

## Data Stores




- A Data store is a repository of data
- Data can be written into the data store. This is depicted by an incoming arrow
- Data can be read from a datastore. This is depicted by an outgoing arrow
- External entity cannot read or write to the data store

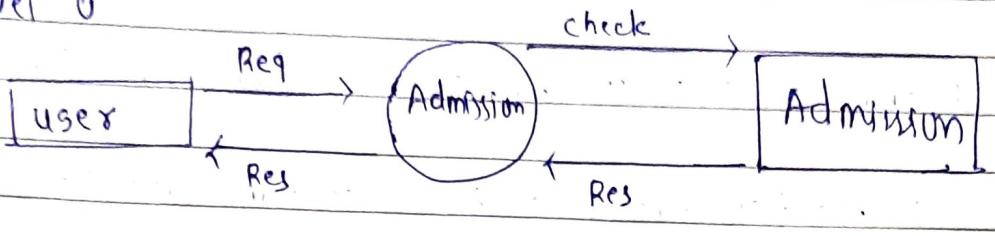
## ★★ Rules of Data Flow

- Data flow from
  (1) External entity to process
  (2) Process to external entity
  (3) Process to store and back
  (4) Process to process

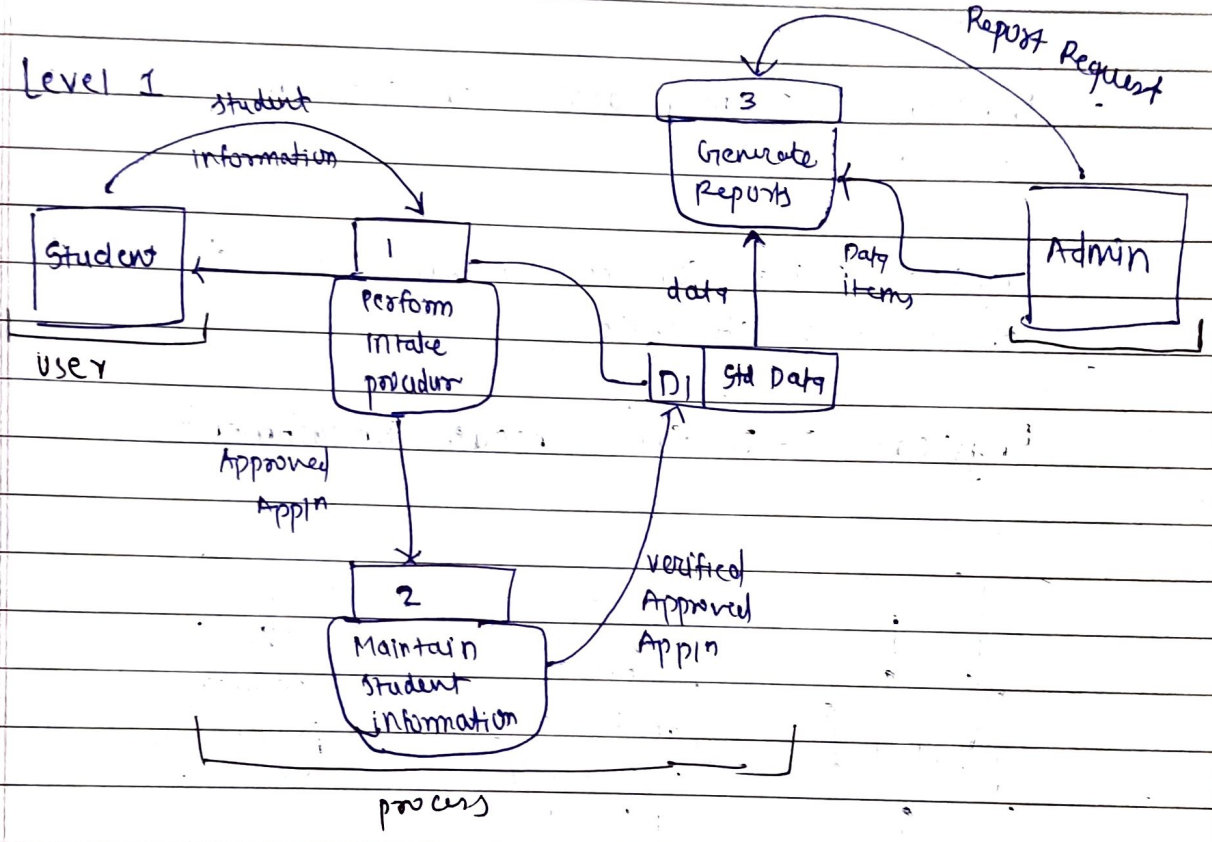
- Data cannot flow from
  (1) External entity to external entity
  (2) External entity to store
  (3) store to external entity
  (4) store to store

# Levels of DFD

## Level 0




## Level 1

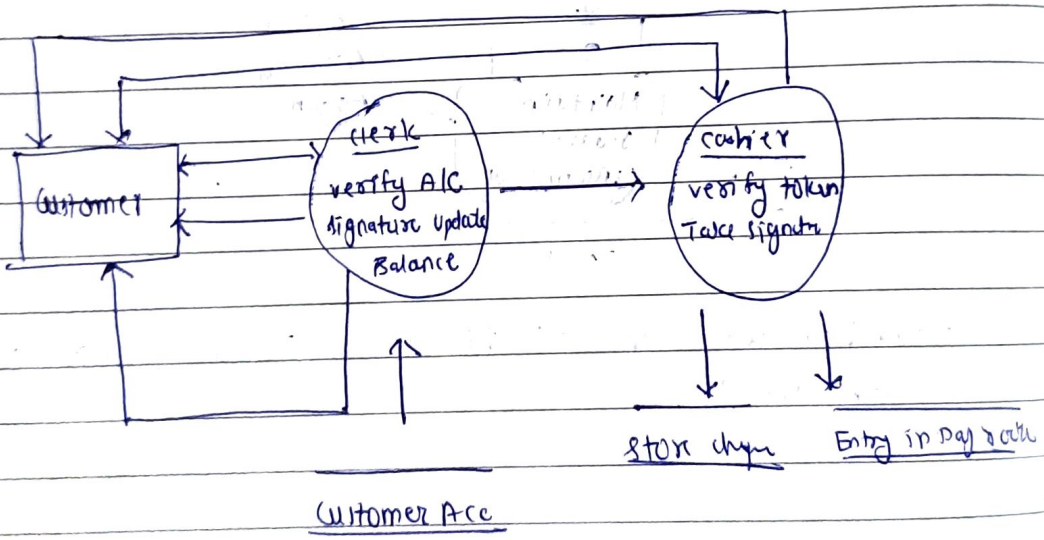



- further we can divide level 2 in parts for each level.

## Logical and Physical DFD

- DFDs considered so far are called logical DFDs
- A physical DFD is similar to a document flow diagram
- It specifies who does the operations specified by the logical DFD
- Physical DFD may depict physical movements of the goods.
- Physical DFDs can be drawn during fact gathering phase of a life cycle.

### Physical DFD for cheque Encashment



# Software Design Approaches

| Function Oriented | Object oriented |
|---|---|
| System is designed from a functional view point. | System is viewed as a collection of objects |
| Top-down | Bottom up approach |
| Divide & Conquer approach | |
| DFD is used | UML is used |
| | |

## SPM (Software Project Management)

- Software Project management is an art and science of planning and leading software projects.

- Main goal is to enable a group of developer to work effectively towards successful completion of project

- Project Manager is an administrative leader of the team.

- Various factors make this job very complex (e.g. changeability, complexity, uniqueness, possibilty of multiple solutions etc.)

## Job responsibilites of Project Manager

- Planning
- Organizing
- Staffing
- Directing
- Monitoring
- Controlling
- Innovating
- Representing

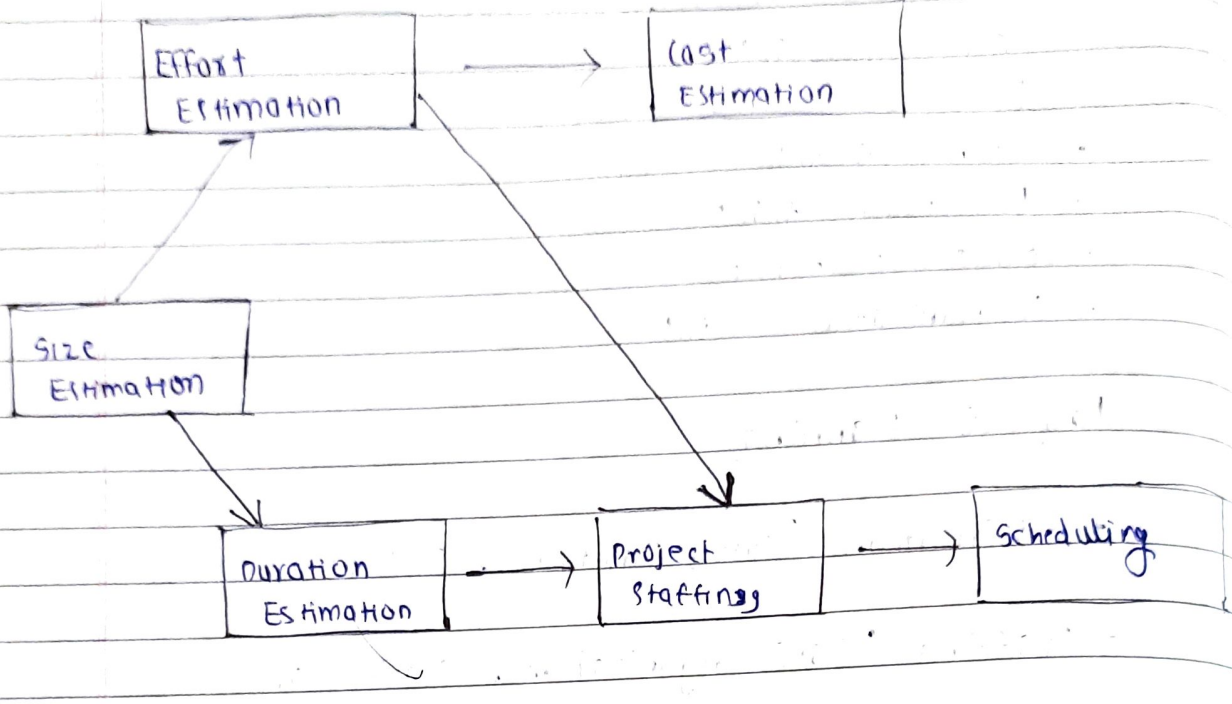
## Skills required for Project Manager

- Managerial skills
- Technical skills
- Problem solving skills
- Coping skills
- Conceptual skills
- Leadership skills
- Communication skills

## Project Planning

- Estimation (cost, Duration, Effort)
- Staffing (Staff organization, Staff plans)
- Scheduling manpower and other resources
- Miscellaneous Plans (Quality assurance plan, configuration, Installation plans)

# Precedence Ordering



# Risk Management

- A risk is "an uncertain event or condition that, if it occurs, has a positive or negative effect on a project's objectives.

- A risk management plan is a document that a project management prepares to foresee risks, estimate impacts, and define responses to risks.

## Reactive vs Proactive Risk Management Strategies

- Reactive risk management tries to reduce the damage of potential threats and speed an organization's recovery from them but assumes that those threats will happen manually.

- Proactive risk management identifies threats and aims to parent those events from ever happening in the first place.

## Types of Risks

- Business Risk :
Building a product that no one wants or loosing budgetary commitments.

- Technical Risk :
Concerned with quality, design, implementation, interface, maintenance problems.

- Project Risk :
Concerned with schedule, cost, resources, customer-related issuses.

## Risk Assessment

RA1 — RA — I, A, P
RC

- It is a process to rank the risks in terms of their damange.

- Determined average probability of occurence value for each risk.

- Determine the impact for each component based on impact assessment matrix (Severity value)

- Risk Assessment values are determined by multiplying the scores for the probability and Severity values together.

# Impact Assessment Matrix

| Probability \ Severity | Negligible 1 | Marginal 2 | Critical 3 | Catastrophic 4 |
|---|---|---|---|---|
| | 5 × 1 = 5 | | | |
| Frequent 5 | Medium - 5 | High - 10 | Very High 15 | Very High 20 |
| likely 4 | Medium 4 | High 8 | High - 12 | Very High 16 |
| occasionally 3 | Medium 6 | High 9 | High - 12 | High - 12 |
| unlikely 2 | Medium 4 | Medium 6 | High 8 | High 8 |
| Rare 1 | Low - 2 | Low - 3 | Medium 4 | Medium 4 |

Risk = probability x severity

## Risk Control vs Risk Mitigation

- Risk control is basically the specific actions to reduce a risk events probability of Happening.

For example :-
Correct inspection and maintenance of the car reduce the likelihood of mechanical failures.

- Whereas Risk mitigation is the set of actions to reduce the impact of a risk event.

For example :-
Airbags are used to reduce the impact of an accident on the passengers and driver.

Risk Mitigation and control strategies

- Risk Avoidance
- Risk Transfer
- Risk Reduction
- Risk Monitoring

# Project Estimation Techniques

- Estimation of various projects parameters is an important project planning activity.
- The different parameters of a project that need to be estimated inculdue —

  (1) Project Size
  (2) Effort required to complete the project
  (3) Project Duration
  (4) Cost

- Accurate estimation of these parameters is important for resource planning and scheduling.
- Estimation techniques can be classified as :
  (1) Empirical Estimation Techniques
  (2) Heuristic Estimation Techniques
  (3) Analytical Estimation Techniques

## Empirical Estimation Techniques :

- Expert Judgment Technique - Individual Person
- Delphi cost Estimation - Team of Experts.

### Single variable Models :

Estimated Parameter = $c_1 e^{d_1}$

### COCOMO (constructive cost Model)

- Was first proposed by Dr. Barry. Boehm in 1981
- Lines of code
- prog. lang. ranging
- Based on the waterfall model

Types
(1) Organic
(2) Semi-detached
(3) Embedded

## Versions of COCOMO

(1) Basic Version
(2) Intermediate COCOMO
(3) Complete COCOMO

(1)

$$EFFORT = a_1 \times (KLOC)^{a_2} \ PM$$
$$Tdev = b_1 \times (Effor)^{b_2} \ Months$$

### Merits :-

Basic COCOMO is good for quick, rough and early estimate of software costs.

### Demerits :-

- It does not account for differences in H/W constraints, personnel quality and experience, use of modern tools and techniques and so on.
- The accury of this model is limited because it does not consider certain factors for cost estimation of software.

## Intermdiate COCOMO Model :-

- It compute effort as function of program and size of "cost driver" that include subjective assesment of product, hardware, personnel & project attributes.

- This model uses a set of 15 cost drivers these cost driven are multiplied with the initial cost and effort estimates to scale the estimates up and down.

- Product Attributes
(1) Required software reliability
(2) Size of application database
(3) Complexity of the product.

$$E = a_i (KLOC)^{a2} \times (EAF)$$

↓

effort Adjustment factor

## Merits :

- This model can be applied to almost the entire software product for easy and rough cost estimation during early stage.
- It can be applied at the software product component level for obtaining more accurate cost estimation

## Demerits :

- The effort multipliers are not dependent on phases.
- A product with many components is difficult to estimate.

## Complete COCOMO :-

A management information system (MIS) for an organization having offices of several places across the country.

(1) Database part (semi-detached)
(2) GUI (Organic)
(3) Communication part (embedded)