SE.

Software

* Software is a collection of executable program codes
* It is associated with libraries & docs.

Characteristics of Software - new thing

* Software is developed or engineered, not manufactured → modify/copy.
* S/w does not wear out. - no expiry date.
* S/w is custom built. - acc to customer.

Evolving Role.

* Software has dual Role.
  - Product
  - Vehicle.

i) S/w as product.

Produces, manages & displays info.

2) S/w as vehicle.

for delivering the product (process).

Vehicle:- Information transformer (Single bit info / multimedia)

→ It supports or directly provides System functions.

→ Control of other programs. eg OS.

→ Effects communication. eg N/w.

– Helps in building other software eg S/w Tools.

CHANGING NATURE OF S/W.

* Software is everywhere & anywhere.
* It keeps on changing from situation to situation to accomodate user chores
* 7 categories.

1) System Software :-
* Collection of programs written to provide service for other programs.
* eg :- Compilers, file management utilities, Editors.

2) Application Software
* Designed to help users to perform specific task.
* For example :- If we want to calculate something, we use calci app etc
* eg :- C and Java.

3). Engineering & Scientific S/W.
* Basically to do all complicated numeric calculations
* eg :- Calculus, Statistics

4) Embedded S/W
* it lies in the hardware of system & is used to control functions
* eg :- Microwave oven controls.

5). Product line S/w
It provides specific capabilities for SE
eg:- CG, DBMS.

6). Web Application S/w
These are used over a n/w.
eg:- Websites

7) AI S/w
Non numeric algorithms to solve complex problems
eg:- Robotics, Gaming.

LEGACY SOFTWARE

What is a legacy S/w?
* It is a very old S/w
* which is developed decades ago
* Inextendible design (poor quality)
* No documentation
* Can't meet the needs of new technologies

Solution for Legacy S/w
* S/w should be adapted to meet new computing environments & technology
eg:- We are using 5G, but pure mobile which use 3G, 4G can't accomodate 5G

* S/w must be enhanced (improved) to meet new business requirements
* Basically legacy S/w is unextendable so improved this we must make Extendible-design to make it inter operable with other systems.
* We have to redefine the architecture so that it is suitable to new event.

SOFTWARE MYTHS (assumptions (may/may not be ✓)) — (3).

* Management Myth :- (mangements thinks ki ...)

1) * Myth :- Available Standards and procedures defined in a book are enough.
   * Reality :- Is that book correct?
     Is that book complete?
     Is that book adaptable to the newly coming S/w?

2)
Myth :-
"We can add workers when we get behind the Schedule.

Reality :- Who will give the knowledge to the new people at last minute?
The existing people must do their work or train this new people?

Therefore:- From the begining itself we must hire all people and take all the measures to complete the project In the between if we add workers, then it will not help

3)
Myth:-
We can outsource the project to third party and relax. They will take care of our project
Reality:-
Who will take their follow ups?
Who will keep on checking them for the mistakes?

Customer Myths:-
1)* Myth:-
General Statement of project is enough, details can be filled later.
* Reality:- to clear.
- Perfect, details must be given then only we can get an effient project.
- Also, continious communication must be b/w customer & developer.

2) Myth:-
Software requirements can be changed easily because S/w is flexible

Reality :-
When you are changing a particular requirement, there is some cost associated with it and also time

Practioner / Developer Myth.

1) Myth :- Once the program is done written, job is done

Reality :- In the time of maintance & deployment also, we must take care of it

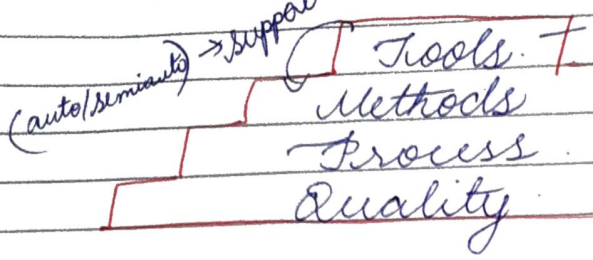
2) Myth :- Until the program starts to run, there is no way to access the quality.

Reality :- We have formal Technical reviews which are more than testing. So on daily basis we need to keep ~~checking~~ assuring the quality

3) Myth :- The only deliverable work product is the Software program code.

Reality :- Along with the code, we must give user manual, because customer also needs to understand. and backups, catalog

LAYERED TECHNOLOGY



→ Software engineering comprises of a process, a set of methods for managing & developing the software and collection of tools.

→ The backbone that supports software engineering is Quality

i) Quality :-
* Degree of pureness as per client req
* Correctness as per client requirement
* Maintainability for future.
* Usability of users

ii) Process :- (What to do)
* A framework must be established for effective delivery of S/W.
* Management & control of S/W projects

(iii) Methods (How to's)
* Provide technical "how-to" for building a S/W.
* Each method consists of multiple tasks.
* Ex: requirement analysis, testing & support.

iv) Tools-
* Provide automated/semi automated support for process & methods.
* A set of tools working at output as input to 2nd tool is an computer aided software.

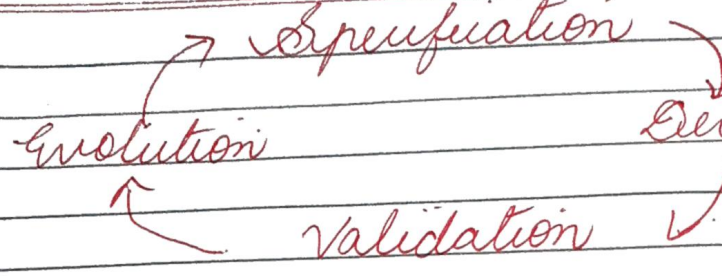
SOFTWARE PROCESS:
A software process is a set of activities & associated results that produce a s/w product

i) Software specification - where customers & engineers [define] a s/w that is to be produced & the [constraints] on its operation.

ii) Software development - where s/w is [designed] and [programmed]

iii) Software Validation - [Checking] whether the s/w is upto the client requirements

iv) S/w Evolution - The s/w must evolve to meet changing customer needs. (Further changes should be possible

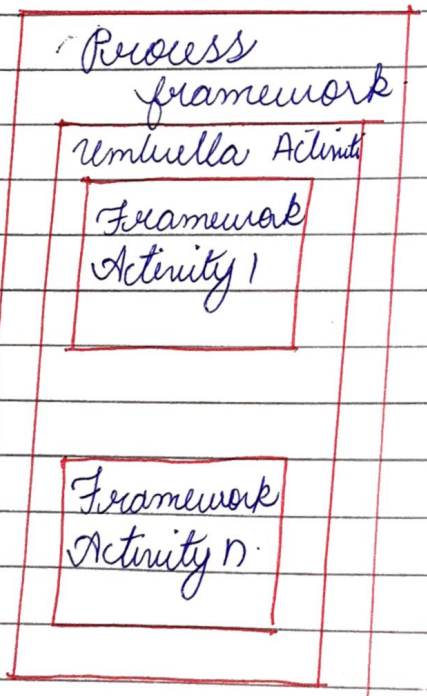


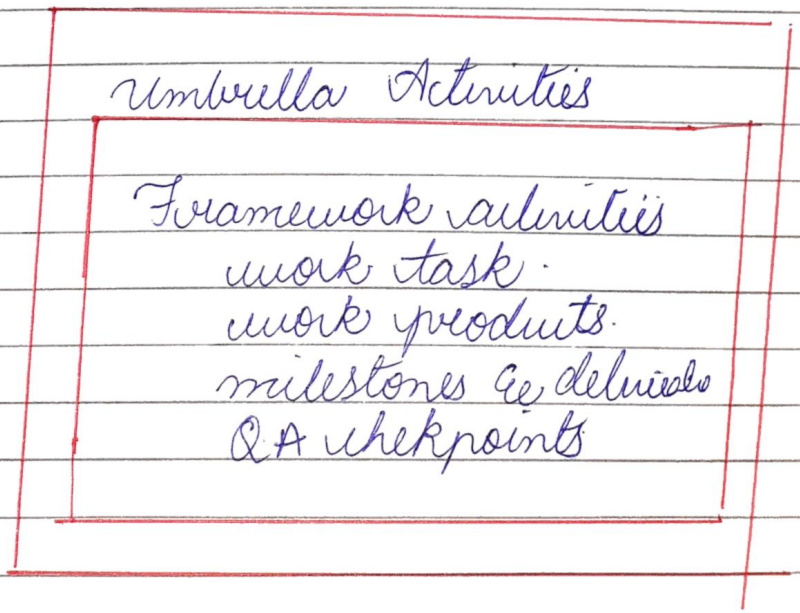
PROCESS FRAMEWORK.

* Establishes the foundation for SE by identifying small no. of framework activities and umbrella activities that are applicable for the entire S/w process

S/w Process

Process framework.





Framework Activities → (5). CPMCD

1) Communication:-
It involves communication with customer & gathering requirements.

2) Planning:-
It establishes a plan for a project

3) Modelling:-
→ blueprints
Constructing models so that customer can understand the requirement & design in a better way.
has (2) actions.
1). analysis - work tasks
2) design - creation

4) Construction:-
Combo of code generation & testing.

5). Deployment:-
S/w is delivered to customer
↓
evaluate s/w & send feedback.

UMBRELLA ACTIVITIES (8)

Framework activities — small projects
Umbrella activities — large projects

↓
to track control & manage the project.

i) Tracking & Controlling S/w projects
* Assessment of progress of the project against the plan of the project
* Maintain the Schedule of the project by taking appropriate action based on the above assessment

(ii) Managing Risk:-
* Evolution of those Risk that have serious impact.

(iii) S/w Quality Assurance (SQA)
* This activity ensures the S/w quality by defining & organizing the activities needed for assurance of quality of software (Identify analyse solve

(iv) Formal Technical Reviews:-
* This activity tries to eliminate any errors as quick as possible

before they spread to other modules

v). Software Measurement:-
The activity describes as well as gathers measures of process, product & project through S/w team (cost, time, manpower)

vi) S/w Configuration Management (SCM):-
The changes to the S/w components can be done here. (effects of changes)

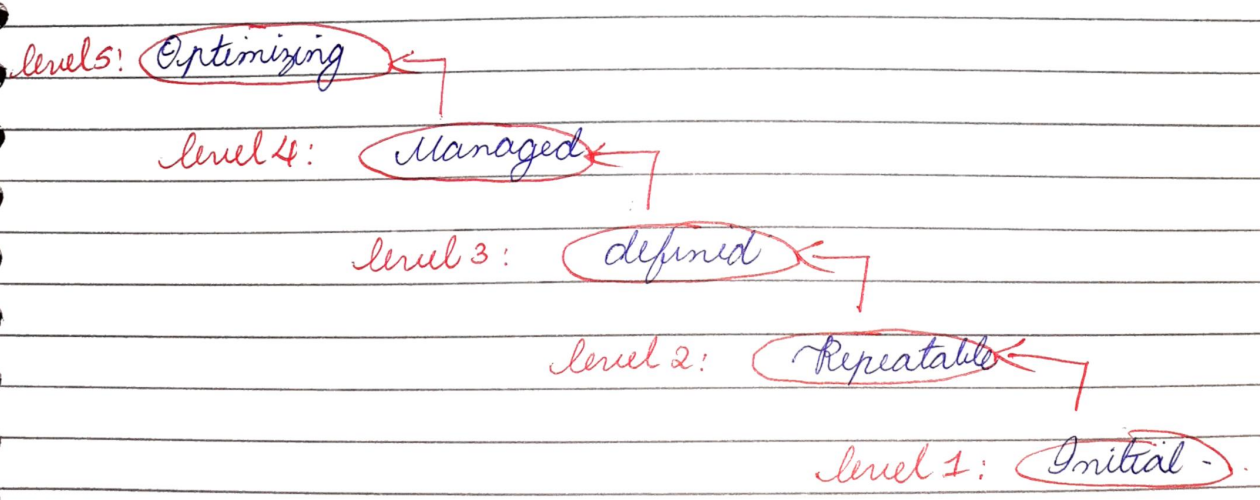
vii) Managing Reusability Factor
Reusing the previous made projects modules for faster delivery.

(viii) Preparing & producing S/w products
The Final activity, models, logs, documents are maintained here

//

CMM:/CMMI CAPABILITY
MATURITY MODEL:/MODE
INTEGRATION:

* CMM was developed & promoted by S/w Engineering Institute (SEI) a research & development centre.
* CMM is not a sw process model.
* CMM is a benchmark to measure the maturity of an organisation's software process.
* The model describes a five-level evolutionary path of increasingly organized & systematically more mature process
* It has 5 levels of Maturity. (Initial, Repeatable, Defined, Managed, Optimizing IRDMO.



* Level 1: INITIAL.
→ Process are basic, poorly controlled & reactive.
→ It is immature

→ They are not well defined.

* Level 2: REPEATABLE
→ It mainly focusses on establishing basic project management policies.
→ Previous experience will be taken into consideration.
→ Project planning will be done.
→ Configuration management will be done
→ Software Quality Assurance will be done

* Level 3: DEFINED:
→ Here mainly the documentations are done.
→ Peer reviews will be done.
→ Intergroup communication will be done
→ Training programs will be done

* Level 4: MANAGED.
→ Quantitative & Qualitative goals are set.
→ Software Quality Management is done
→ Quantitative Management is done.

* Level 5: OPTIMIZING.
→ Work is based upon continuous Improvement.
→ The key characteristic of this level is Focusing on continously improving performance.

→ The main features are
a) process change management.
b) Technology change management.
c) Defut prevention.
eg: level 5 → Cognizant, Wipro.

PROCESS PATTERN:

* A s/w process can be defined as a collection of patterns that define a set of activities, actions, work tasks, work products & similar related behaviour followed in s/w development cycle.
* In common words, while a project is being developed, the team do face multiple problems & they need to resolve it first, so few common problem solutions are pre written so the issues can be resolved faster (using process pattern)
* The collection of patterns will give us a template
* The template will have all the important features
* It has.
  Process Template
  Initial Content
  Problem

Solution
Resulting Context
Related problems.

*) Process template :-

a) Pattern name :
name with the specific issue with has no ambiguity.

b) Forces / Intent
The environment in which the pattern is encountered & the issues that makes the problem visible, [Objective or purpose of the pattern]

c) Types :
- Stage pattern
  It is associated with framework activities
  eg : Communication
- Task pattern :
  It is associated with action or work.
  eg : Requirement Gathering
- Phase
  It is associated with sequence of the framework activities.
  eg : Spiral Model.

* Initial Context :
- It describes the condition under which a pattern applies
  - We are defining it before the starts of the pattern

* Problem:-
The specific problem to be solved by the pattern.

* Solution:-
Describes how to implement process pattern.

* Resulting Context:-
Describes the condition that will result once the pattern has been successfully implemented

* Related Problems:-
Provides a list of all process patterns to the current pattern.

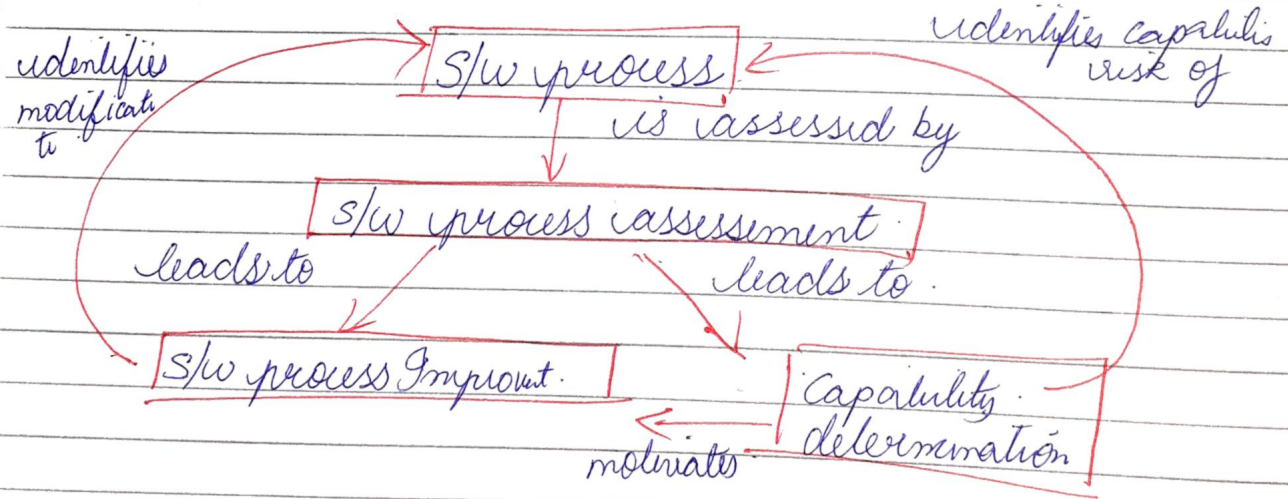
* Known Uses & examples:-
Indicates specific instances on which the pattern is applicable

PROCESS ASSESSMENTS:-

* Whenever we apply the process patterns to the ongoing software engineering project, it does not mean that the software is going to satisfy all the essentials

(i.e on time delivery, customer satisfaction, high quality etc) which are essential in success of software

* Hence inorder to achieve this, the software patterns should be collaborated with highly valued S/w engineering practice & the entire process should be sufficiently assessed as required
* Process patterns + S/w project → S/w obtained (req/not).
* ∴ S/w patterns + real time practices
* And this entire process should be continiously assessed.
* This process is called process assessment



Techniques in Process Assessment

1). Standard CMMI Assessment method for process Improvement (SCAMP) 2 steps

- Initiating
  It is when we are starting something
- Diagnosing
  It is when you are testing something
- Establishing
  It is when you are implementing
- Acting
  It is when you are performing "
- Learning
  It is when you are learning.

2) CMM - Based Appraisal for Internal Process Improvement.

* It provides a diagnostic technique for assessing related maturity of the software organization.

3) SPICE - The SPICE (ISO/IEC 15504.

* It defines a set of requirements for spw processing assessment (standards).

4) ISO 9001 : 2000 :

* Any Software Industry by adapting ISO 9001 : 200 standards can breath quality peaks in terms of its products system Plan, do, check, Act.

PERSONAL AND TEAM PROCESS MODELS

A) PERSONAL S/W PROCESS (PSP).

* It concentrates on personal s/w models.
* It measures the work product of its quality
* There are basically 5 Objectives
  - Planning :-
    A Sequence of steps you will be assuming.
  - High Level Design
    For each and every component, you will be choosing external requirement & then a component level Design will be created. also we will be identifying issues & errors
  - High Level Design Issues.
    If we have left out any errors in the high level design stage, those will be rechecked.
  - Development :-
    The component level design which we have developed in the High level design that will be reviewed, code is generated, it is compiled & also we will be collecting all the metrics
  - Postmortem :-

It will use the measures & metrics collected and it will measure the effiency of the S/w

B) Team SOFTWARE PROCESS (TSP).

* It concentrates on building a project team that provides high quality S/w.
* It has objectives
  - Building Self Directed Teams:
    One's own control to direct a team and the team will have 2-20 members
  - motivate the teams:-
    Motivating the team is done by the manager
  - accelerate S/w process improvement
    Taking it to the highest level or the last stage
  - Improvement Guidance
    Providing improvement guidance to high machurity organization

PROCESS MODELS:-

* A process model describes how each step in SDLC works & what are the process required for each step in SDLC

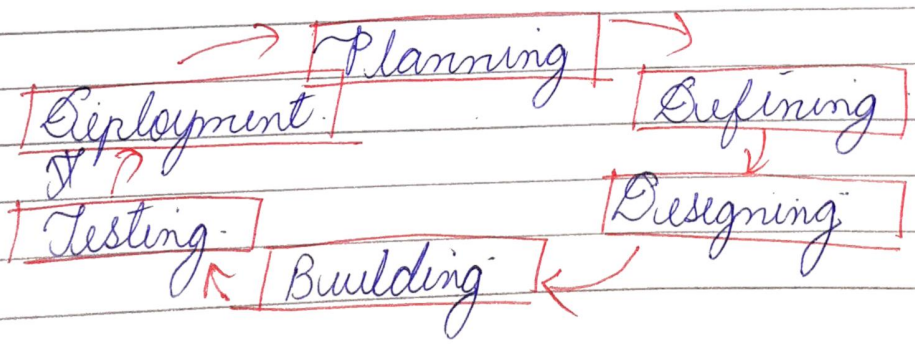
We have 5 process models.

1). Waterfall Model.
2) Incremental Model
   — Incremental
   — RAD.
3) Iterative Model.
   — Prototype
   — Spiral.
4) Evolutionary Model.
5) Unified Model.

//

SOFTWARE DEVELOPMENT LIFE CYCLE (SDLC)

* SDLC is a process used by Software industry to design, develop and test high quality Softwares.
* ISO/IEC 12207 is an International Standard for S/w development life cycle
* SDLC is also called S/w development process



1) Planning & Requirement Analysis

* Requirement Analysis is the most & fundamental step in SDLC
* It is performed by the senior members of the team with i/p's from the client.
* This info is then used to plan the basic project approach & to conduct product feasibility study in the economical operational & technical areas.
* Planning for the QA requirements & identification of risks associated in the project is also done in planning stage.
* The outcome of this step is various technical approaches that can be followed to implement the project successfully with minimum risks

2) Defining Requirements

* Once the requirement analysis is done, the next step is to clearly define & document the product requirements & get approved from the client
* SRS (S/w requirement Specification) document consist of all product requirements to design & develop.

SE

3). Designing the product Architecture
* Based on the requirement specified in SRS, usually more than one design approach for the product architecture is proposed & documented in DDS (Design Document Specification).
* Among which, one is finally selected & all the diagrams like data flow representation.

4). Building Or Developing the product:-
* During this stage, DDS is converted to programming code.
* Developers follow SRS, based on that, they start working and based on the requirement they use specific lang (C, C++, Py etc).
* Here the outcome will be the S/w.

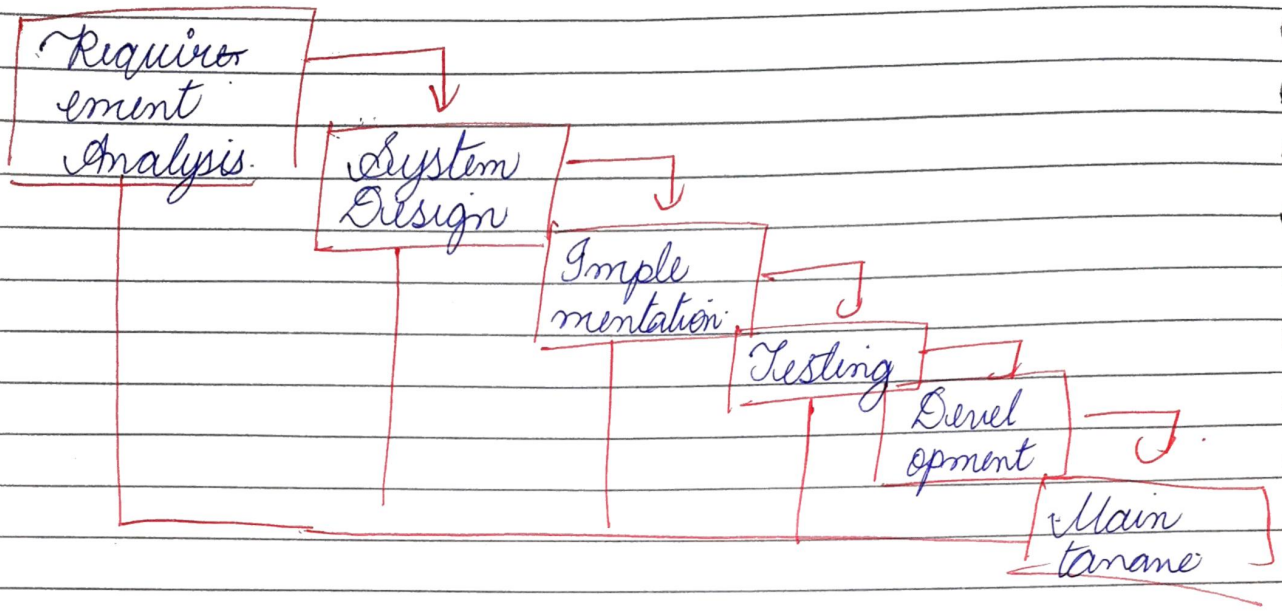
5) Testing the Product
* Once the development is done, testing of the S/w is necessary to ensure its smooth execution.
* Here all flaws are fixed & re tested.
* This ensures that the product confronts the quality requirements of SRS.

6) Development in Market & Maintenance-

* Once the product is tested & ready to be deployed, it is released formally in the appropriate market
* Even few products are released in specific countries before global release & based on the feedback, changes are made tested & published global.
* Frequent maintenance is also done based on the feedback & client requirements

WATERFALL MODEL:-

* Waterfall model was the first SDLC model used in SE
* It is used to ensure the success of the project
* In the waterfall approach, the whole process of software development is divided into seperate phases
* In the waterfall model, one's outcome of one phase acts as an input for the next phase sequentially.



1) Requirement Gathering & Analysis.
All the possible requirements of the System to be developed are captured in this phase & doumented

2) System Design :-
System Design helps in Specifying hardware & System requirements helps in defining the overall System architecture

3) Implementation :-
In this phase, at first small programs called units are written & tested individually called unit testing

Integration & Testing :-
All the units developed are integrated into a System & testing is done.

Deployment of System :-
Once testing is done, the product is ready for deployment.

Maintenance :-
If there are any Issues, they will be resolved here

ADVANTAGES OF WM :-

* Simple & easy to use.
* Phases are started & completed one at a time (Phases do not overlap)
* Requirements are very well understood.
* low cost.

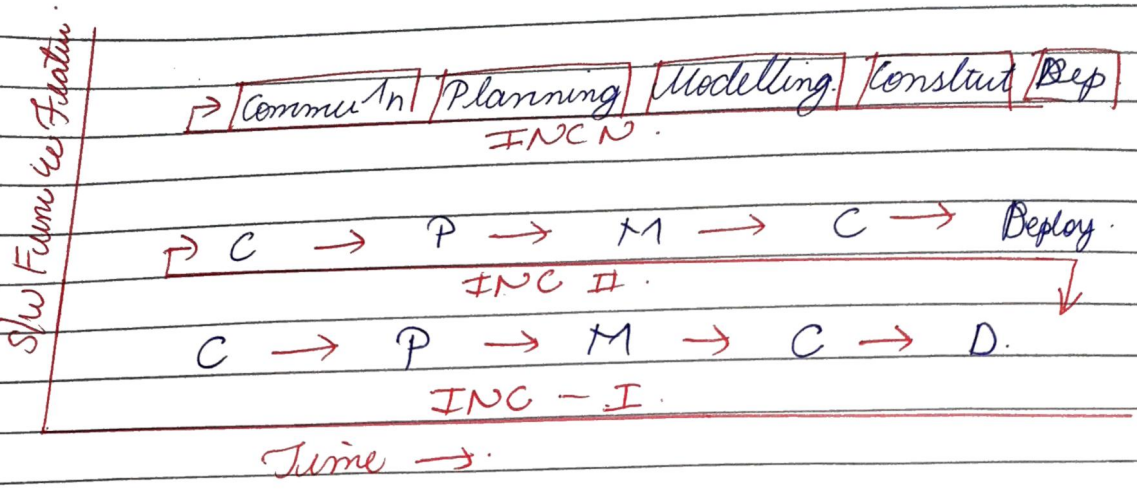
DISADVANTAGES :-

* Difficult to acquire requirement in starting.
* High risk
* not feasible.
* Not suitable to accomodate any change.
* Not good for large size projects.

INCREMENTAL PROCESS MODELS.

a) INCREMENTAL MODEL :-

* IM divides its S/W dev process into certain no. of increments.
* Each ↑ consists of same steps (communication, planning, modelling, construction & deployment)
* Hence to make a big project into multiple parts, we must deal with the requirements
* It is based on the idea of developing an initial implementation, exposing this to user feedback & evolving it through several versions until an accepted system has been developed.
* Important functionalities of the S/W are done in the 1st iteration
* Each subsequent release of S/W module adds fun to the prev release. The process continues till the complete system is achieved
* It is used for S/W with less features.
* Used in day to day applications
  Eg :- Banking
* Less manpower is required



The 5 Stage.

i) Communication
ii) Planning.
iii) Modelling.
iv) Construction
v) Deployment

Advantages:-

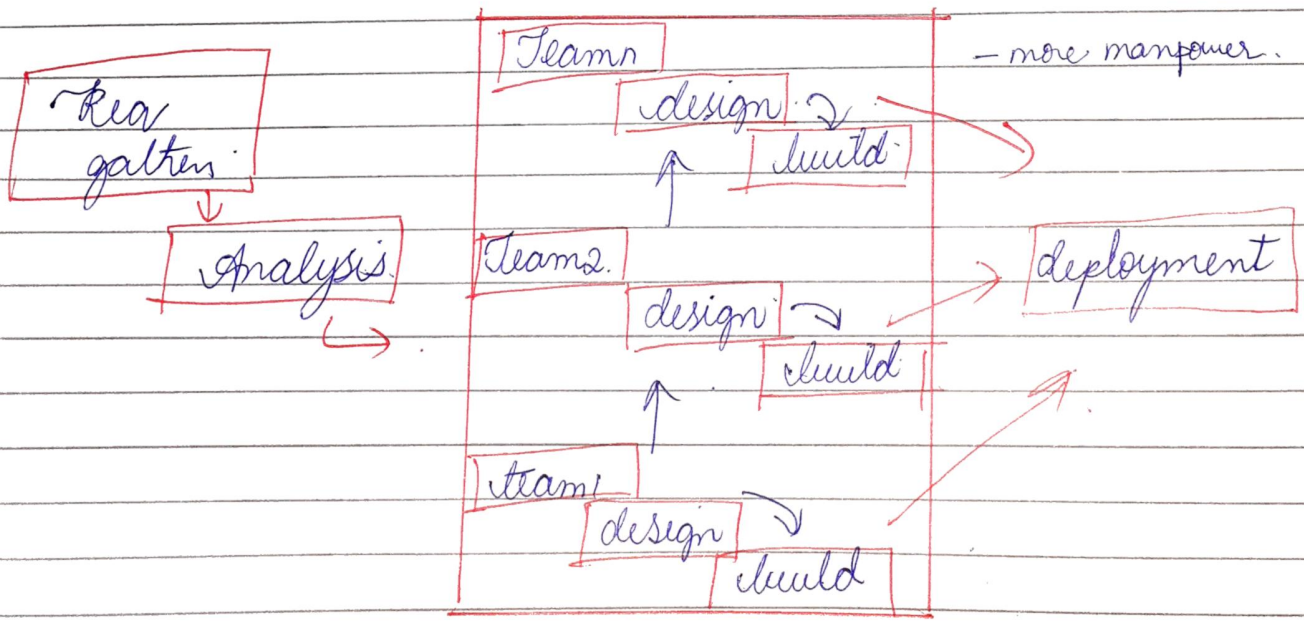
* Generate working S/w quickly & early during the S/w life cycle.
* This model is more flexible, less costly to change Scope & requirements
* It is easy to test & debug during small iteration.
* Customer & client can respond to each build & give feedback if any changes needed
* easier to manage risk

Disadvantage :-

* Total cost will be higher (as can lead to multiple iterations)
* Needs a clear & complete definition of the whole system before it can be broken down & build incrementally
* Needs very good planning & design

RAD :- RAPID APPLICATION DEVELOPMENT MODEL :-

* Rapid Application model proposed by IBM in 1980, is suitable when the customer requirement is clear but schedule is very short



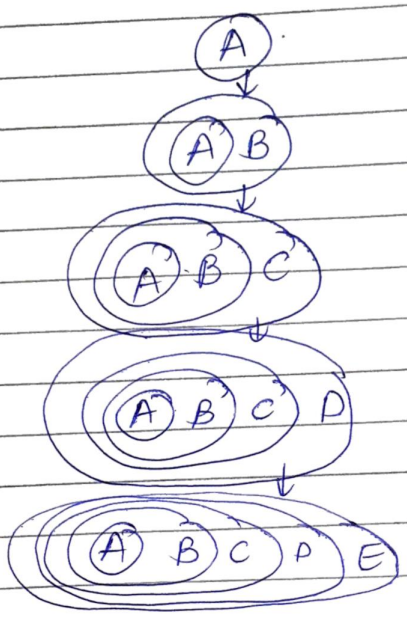
* Here the project is divided into different teams & develop over a short period of time simultaneously
* When all the independent modules are available, Do Integrate the module to get final product.
* Note:- Teams work parallely (not one after the other)

Advantages:-
* Reduced development time.
* Increase reusability of component.
* Quick review is possible
* Encourage customer feedback.
* Requirement changes can be accomodated.

Disadvantages:-
* Require high skilled developer
* Should not be used with small projects.
* Only system that can be modularised can be build using RAD.

EVOLUTIONARY MODEL:-
* Combo of iterative & Incremental model
* Here we break our work into smaller parts.
* Prioritize those parts & deliver to

the customer one by one
* It is also called as Successive Version Model)
* Example:- 5 phases A, B, C, D, E.



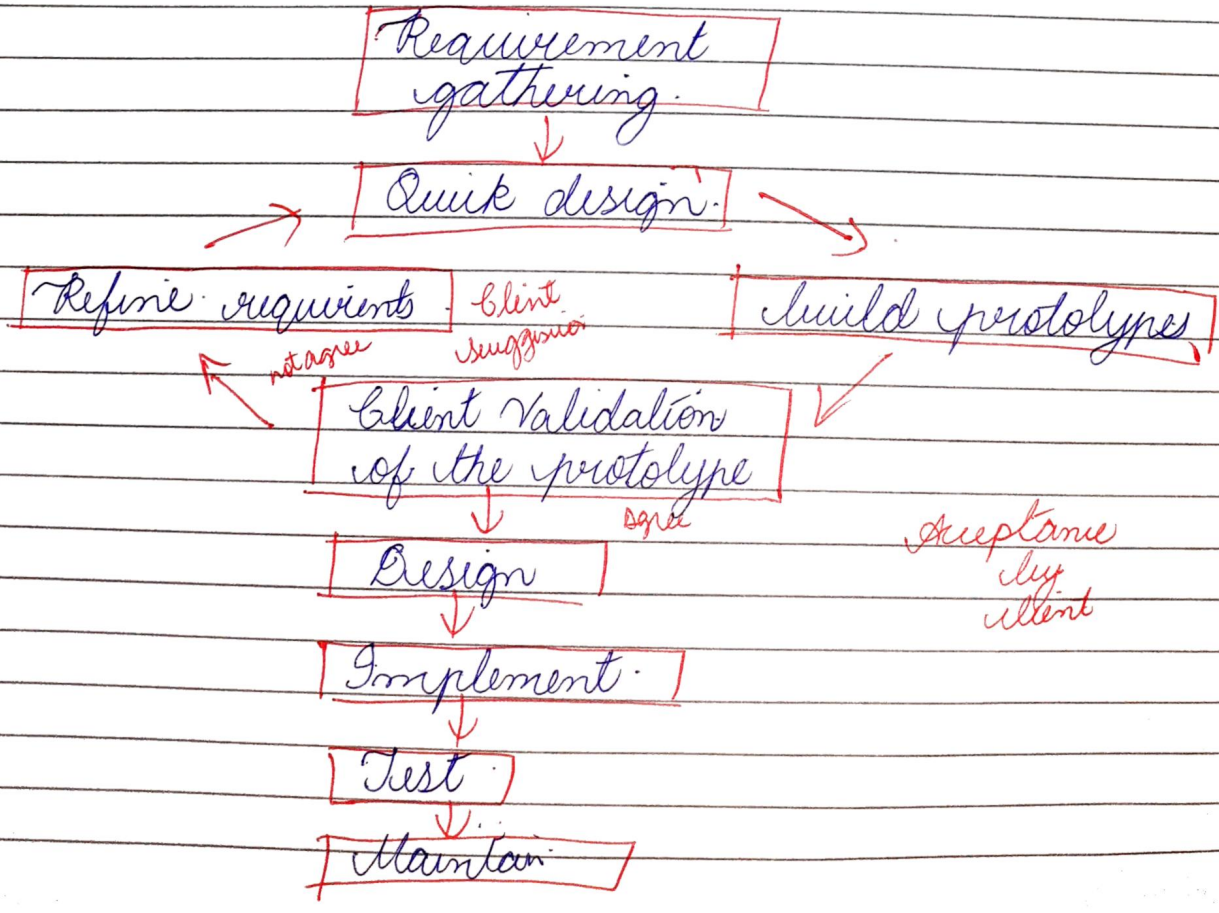
— final software.

* Iterative Models:- 2 types
  — Prototyping Model.
  — Spiral Model.

PROTOTYPING MODEL.

* Most of the customers are not sure about the functionality they require from the software.
* As a result, the final software is not according to the exact demand
* It is also called. RAPID THROWAWAY MODEL

* One requirement gathering is done, a basic design & prototyping is done & show to the client for validation.
* Prototyping means the sample or the dummy model.
* If client informs no this was not my requirement, we must collect the requirement again & do basic design & prototype & ask for validation until he accepts it.
* One the client accepts it, we move on to other common phases like Design, Implement, Test, maintain



Advantages:-

* Customer gets a chance to see the product early in the life cycle we give important feedback.
* There is a scope to accomodate new requirements
* Developers will be more confident (as prototype was accepted) hence risk is reduced.

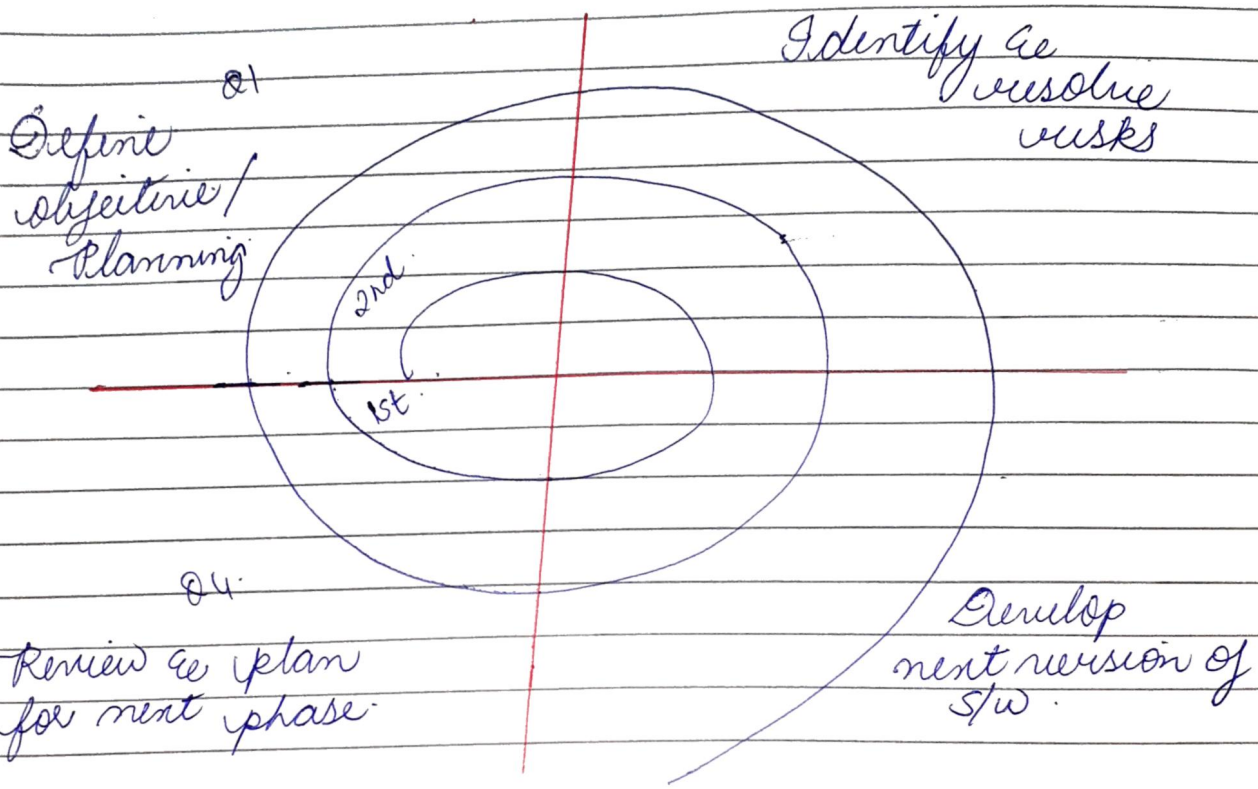
Disadvantages:

* After seeing the early prototype user demand the actual system as per his/her requirements
* If not managed properly, the iterative process can run for a long time.
* If the client is not interested he may lose his interest in the project

B) SPIRAL MODEL

* It was developed by Barry Boehm in 1986
* The main objective of this model is to handle risks.
* Each path/Spiral has its own cost we keeps ↑
* Each Spiral is divided into 4 activities

* Radius of spiral ↑, cost ↑.
* angle indicates the progress.
  180° - half, 360° complete.



→ Planning :- Here we discuss the key objectives & alternatives if there is a risk. (to handle them)
* Various constraints while development are also considered.

→ Risk Analysis - Identify & resolve risks.

→ Development & Testing
  ↳ Design
  ↳ Code
  ↳ Test
  ↳ Release.

→ Assessment:- Feedback from the Customer evaluation) based on that, the process continues in spiral format

In Spiral 1:-
Planning → Risk Analysis → Prototype is build, Customer evaluation & feedback is collected

Spiral 2:- based review update prototype II is build (with risk Analysis), SRS is written & sent again for validation

Spiral 3:- Here, based on feedback, final prototype II made (avoiding risks) & here we start the design → code → Test → releases. or any Standard SDLC method

Advantages

* Provides early & frequent feedback of the customer, Additional funtionality can be added at a later date.
* Resolve all possible risks involved in the project early in the life cycle
* Continuous customer involvement. so better customer Satisfation

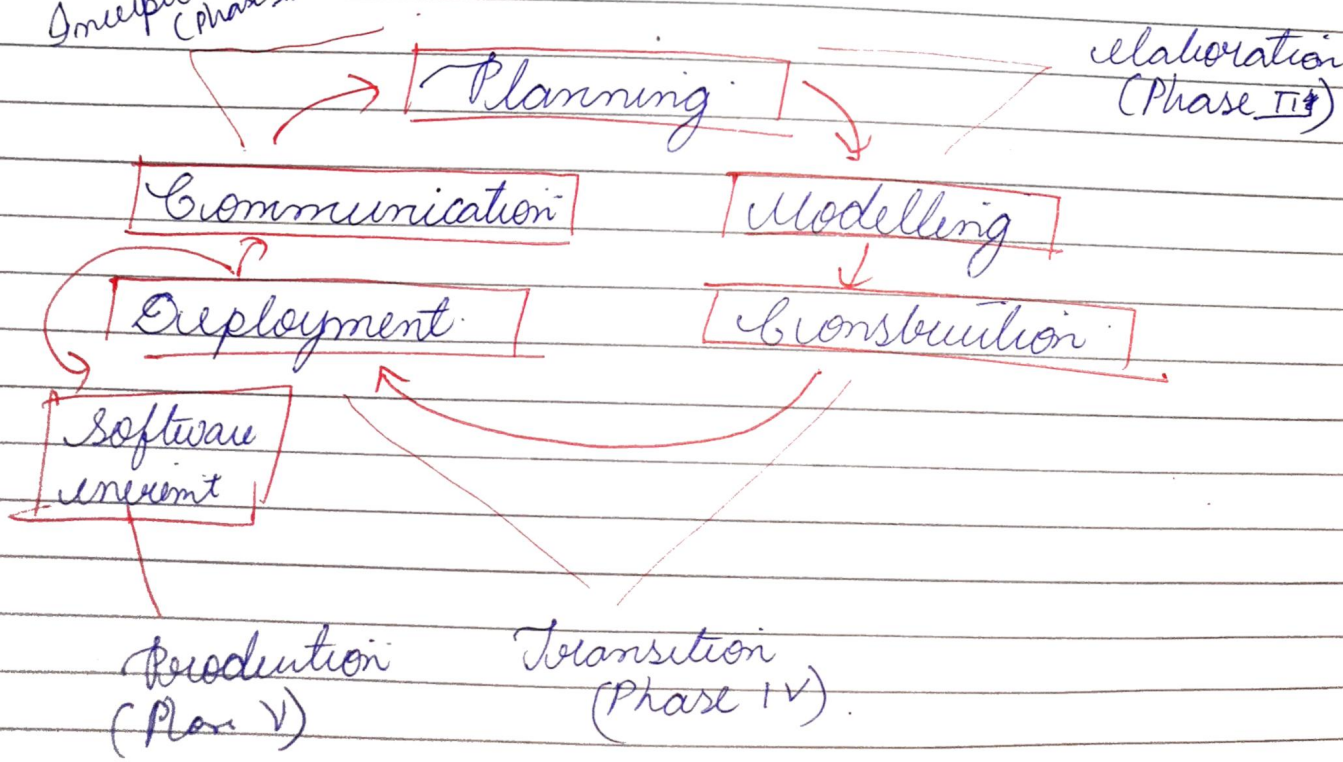
* good for large & mission critical projects.

Disadvantage :-

* not suitable for small size project.
* Complex to use.
* Risk analysis requires highly specific expertise.

UNIFIED PROCESS MODELS :-

* It is used only in case of Object Oriented Systems (real time systems.
* It is a framework using UML.



I) Inception Phase:-

* The inception phase of UP include with customer communication & activities by collaborating with the customer & endures to collect business requirement & plans are proposed.

II) Elaboration Phase:-

* The elaboration phase encomposes the planning & modelling activities of the generic process model.
* Here elaboration for the usecase. to analyse, design, implementation & development model (in UML)

III) Construction Phase:-

* Using architectural model as input the construction phase develops the software component that will make each usecase operation & further carried out for testing

iv) Transition Phase.

* Here the S/w is given for testing to the end user for B testing manual & Installation manual is made.
* At the end, it is ready to release

V) Production Phase:-
After release, any more needs by the user will be added as per requirement.

Advantages:-

* It covers the complete S/w development life cycle.
* The best practices for S/w development are supported by unified process model.
* It encourages designing in UML

Disadvantages-

* Could be complex to Implement.
* heavy weight process.
* experts are in need for it.
* risks can't be determined

UNIT-2.

Requirement :-

A requirement can range from a high level abstract of a service or of a system constraint to a detailed mathematical functional specification.

Requirement Engineering :-

Requirement engineering is the process of establishing the services that the customer requires from a system and the constraints under which it operates & is developed.

Functional Requirements :-

* The functional requirements specify the features of the S/w system.
* The functional requirements describe what the product must do.
* TFR specifies the actions with which the work is concerned.
* Eg :- For a library management system, allowing user to read article online is a functional requirement

Refer Table.

Non funtional requirements

* The NFR specifies the properties of a S/w system
* The NFR describes how the product should perform
* The NFR specifies UX the user experience while using the system
* Eg:- For a library management system, for a user who wishes to read the article must be authentuated first

USER REQUIREMENTS

* The user must describe the funtional & non funtional requirents in such a way that they are understanddble by the system users who don't have detailed knowledge
* User requirements are defined using natural lang, table & diagrams. Because these are the representations that can be understood by all

Problems of Using Natural lang for Defind user & System Req

* Lack of Clarity

Lack of clarity - Sometimes requirements are given in ambiguous manner. It is required that the client should be clear & precise understanding of requirements

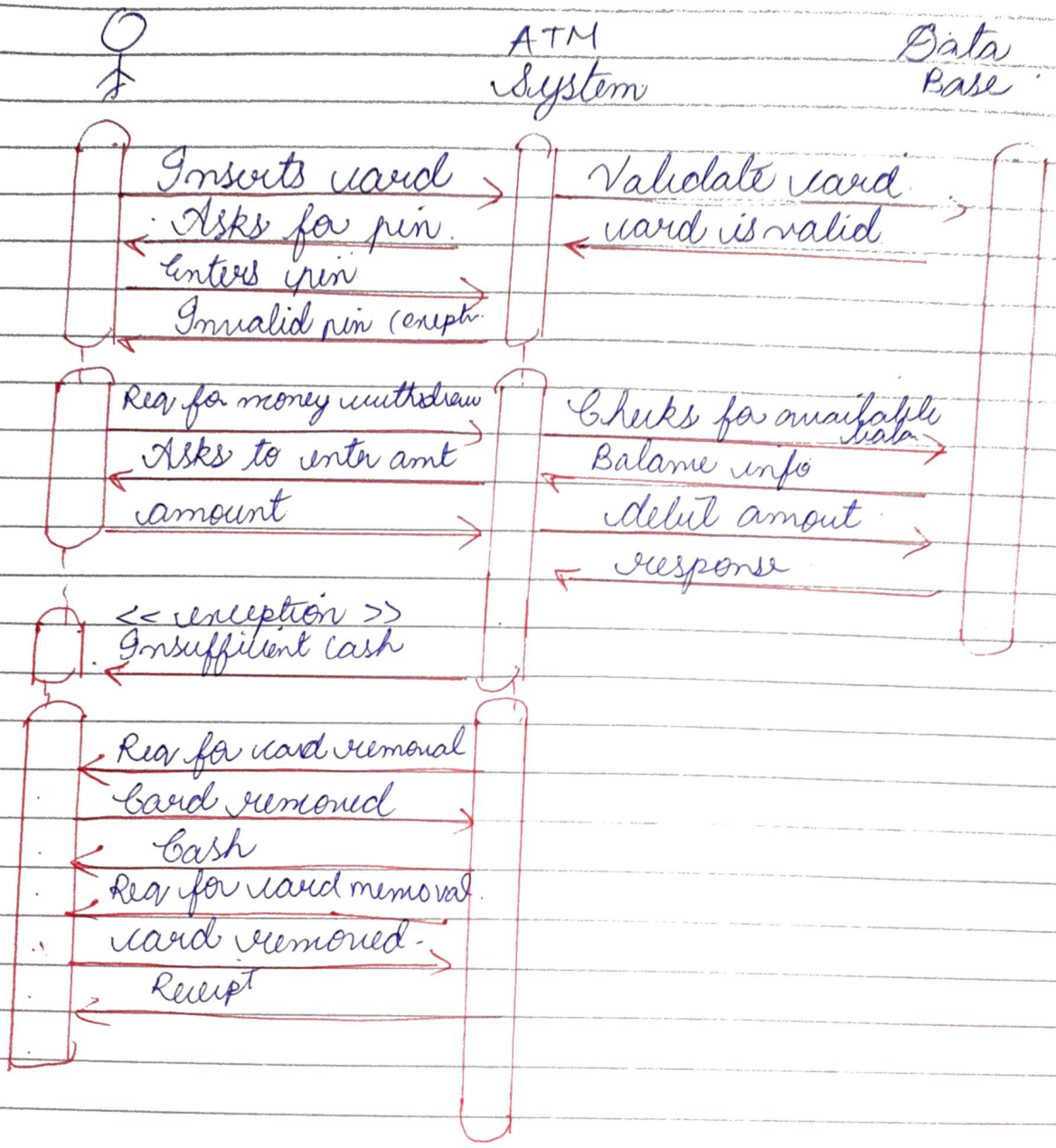
Requirement Confusion - There may be confusion in functional requirements & non functional requirements, system goals & design information

Requirements Mixture - There may be change of specifying several requirements together as a single requirement

SYSTEM REQUIREMENTS -

* System Requirement are more detailed specifications of system functions, services & constraints that user requirements
* The SR may be expressed using system models
* Requirement specifies what the system does & design specifies how it does
* SR should simply describe the external behavior of the system & operational constraints

STRUCTURED LANGUAGE SPECIFICATION



* All requirements should be written in a standard way while using Structured lang specifications

* The advantage of specifying requirements using this method is that the requirement is understandable & expressive
* The only necessary thing while writing requirements using natural language is that some degree of uniformity must be maintained
* Extra info can be added when the requirements are written using NL
* This info can be represented using tables or graphical models.
* One way of using a graphical model is use of sequence diagram.
* The sequence diagram represents the sequence of actions that the user performs while interacting with the system
* Ex:- following is a sequence diagram for withdrawal of cash from ATM.

Interface SPECIFICATION

The connection of old system with new system is called system interface

3 types

1) Procedural Interfaces
2) Data Structures
3) Representation of Data.

THE SRS.

Document Title

Author (S).
Affiliation
Address
Date
Document Version

1) INTRODUCTION:-

1.1:- Purpose of the document

Describes the purpose of the document

1.2:- Scope of the doc

Describes the scope of requirements definition effort. This section also details any constraints that were placed upon requirement elicitation process such as schedule costs.

1.3:- Overview:-

Provides a brief overview of the product defined as a result of the requirement elicitation process.

2) General Description-

* It describes the general functionality of the product
* Describes the features of user community

3) FUNCTIONAL REQUIREMENTS.

3.1) Description.
A full description of the requirement

3.2) Criticality
Describes how essential this requirement is to the overall system.

3.3) Technical Issues
Describes any design or implementation issues involved in satisfying this req

3.4) Cost & Schedule
Describes relative or absolute costs of the system.

3.5) Risks.
Describes the circumstances under which this requirement might not be satisfied.

3.6) Dependencies with other requirements
Describes interactions with " ".

3.7) Any other appropriate
Any other appropriate is used.

4) INTERFACE REQUIREMENTS.

4.1) User Interfaces
Describes how the product interfaces with the user.

4.2) Hardware Interfaces
Describes Interfaces to hardware devices.

4.3) Communication Interfaces
Describes s/w Interfaces

4.4) Software Interfaces.
Describes any remaining software

5) PERFORMANCE REQUIREMENTS:-
Specifies Speed & memory requirement

6) DESIGN CONSTRAINTS.
Specifies any constraints for the design team such as S/w or hardware.

7) OTHER NON FUNCTIONAL ATTRIBUTES.
7.1) Security :-
7.2) Binary Compatibility

8) OPERATIONAL SENARIOS
This Section should describe a set of Scenarios that illustrate, from user's perspective, what will be experienced when utilizing the System under various Solutions.

9) PRELIMINARY SCHEDULE.
This Section provides an initial version of the project plan.

10). PRELIMINARY BUDGET.
This section provides initial budget for the project

11) APPENDICES

11.1:- Definitions, Acronyms, Abbreviations
Provides defination terms, and acronyms can be provided.

11.2:- References
Provides complete citations to all the documents & meetings referenced

CHARACTERISTICS OF GOOD SRS.

* SRS must be correct
* " " " unambiguous
" " " complete
" " " consistent
" " " traceable

REASONS WHY Quantitative requirement practice is difficult

* Customer may find it practically difficult to translate the system goals into Quantitative requirements.
* Ex:- If the goal is maintainability then there is no metric to translate into quantitative measure.
* Ex2:- For instance if reliability is 4 the customer may find it difficult to understand what the no 4 stands for in regards to reliability

REQUIREMENT ENG PROCESS

The process of gathering, analyzing, documenting, validating, managing requirements

1) Feasibility Study.
2) Requirement Gathering
3) S/w requirement Specification.
4) S/w requirement Validation

participants :- S/w Engineers, Managers, Customers, Users.

1) Feasibility Study :-
   * Client approaches the organisation
   * A rough idea about what all funtions the s/w must perform & which all features are expected from the s/w
   * The output of this phase should be feasibility study report

2). Requirements Gathering
   * Analyst & engineering gathering requirements from the customer.
   * System Analyst creates a SRS.
     SRS should
     1) User requirements in natural lang
     2) Technical req in structural lang
     3) Design description written in Pseudo code

4) GUI Screens.
5) DFD

TYPES OF FEASIBILITY STUDY.

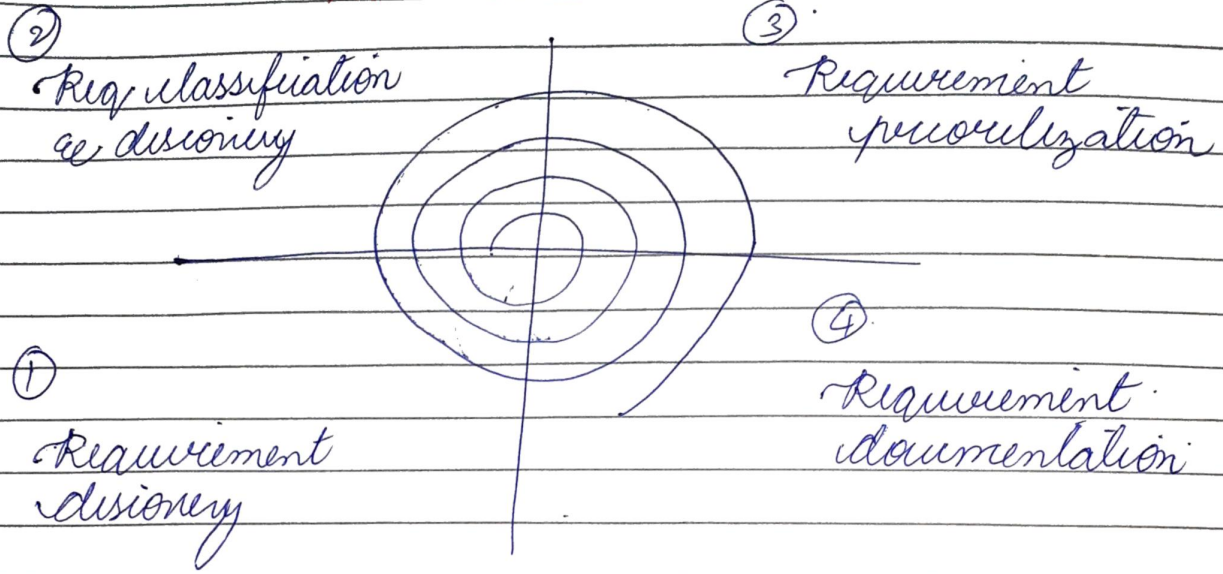
1) Technical Feasibility - Study of config of sys
2) Operational Feasibility - Based on human & politics
3) Economic " - Cost & benefit analysis
4) Management " - Managment agree upon proj idea.
5) Legal " - proj is legally acceptable or not.
6) Time " - proj will be completed in time.
7) Social - proj will be accepted by people

Software Scope of Feasibility-

* A feasibility study is a study made to decide whether or not a proposed system is worthwhile
* The S/w scope clearly defines all functionalities & artifacts to be delivered as a part of S/w.
* The Scope identifies what a product will do & what it will not do, what the end product will contain & what it wont.

Human Centred Activity (or)

REQUIREMENT ELICITATION & ANALYSIS.



ACTIVITIES OF REQ ELICITATATION & ANALYSIS.

The Spiral model depicts the requirement elicitation & analysis process

i) Requirement Discovery.

By having effective communication with the customer the requirements can be identified.

2) Requirement Classification & Discovery.

All the unstructured requirements can be categorized systematically depending upon the nature. And they are arranged in groups

3) Requirement Prioritization - There are some conflicting requirements which are to be prioritized first. If any conflicts occur, then it is resolved by requirement engineers.

4) Requirement documentation - This is the specification of all requirements. And can be any req doc is created.

//

USER AND SYSTEM REQUIREMENTS.

User requirements -

* They are statements in natural lang plus diagrams of what services is expected to provide to system users & the constraints under which it must operate.
* The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system func.

System requirements -

* They are more detailed description of the software's system func, services & constraints.
* The System requirements document should define exactly what is to be implemented

* It maybe a part of contract b/w System & buyer & S/w developers.

DESIGN PROCESS AND DESIGN QUALITY.

Process Design Process

* Design process is an step by step & repetitive procedure where S/w is simulated according the requirement
* The simulation will be in detailed (high level of abstraction
* The design represented by this level can be directly traced to the specific system objective & requirements such as data, functional & behavioral requirements -
* As said earlier, design process is required for better quality software

3 Main Characteristics are-

* Design must consider & fulfill all the customer requirements that are implicit & must implement explicit requirements contained in analysis model

* The design must be readable understandable & as a guide for those who generate code & for those whose test & subsequent support for the S/W.
* Design should provide a complete picture of the S/W, addressing data, Funtional & behavioral domains

DESIGN QUALITY:-

* These are the few guidelines which are the characteristics of a good design.

1) A design should exhibit a architecture that has
   a) creation by recognizable architectural styles or pattern
   b) Components leading to good design characteristics & evolutionary fusion which facilities implementation & testing.

2) A design should be modular base

3) A design should have diff representations of data architecture interfaces & components.

4) Design should explore the data structures useful for the classes to be implemented.

5) A Design should consists of components that have independent functional characteristics

6) A Design should lead to surface that reduces the complexity.

7) Design should be achieved from. iterative method.

SOFTWARE DESIGN AND ITS LEVELS

The design phase of SE deals with transforming the customer requirements as described in SRS document into a form implementable using programming lang.

There are 3 levels:-

1) Interface Design
2) Architectural Design
3) Detailed Design.

INTERFACE DESIGN

* Interface Design is the specification of the interaction b/w a system & its env.
* During Interface Design, the internal of the system are completely ignored & the system is treated as black box.
* Interface design must include the foll details.

→ Precise description of events in the environment, or messages from agents to which the system must respond.

→ Precise description of the events or messages that the system must produce.

→ Specification of data, data coming into & going out of system.

→ Specification of the ordering & timing relationships b/w incoming events or messages & outgoing events or output

ARCHITECTURAL DESIGN

* Architectural Design is the specification of the major components of the system.
* It includes the responsibilities, properties, interfaces, relationships & interactions b/w them.
* In Architectural Design, the overall structure of the design is chosen, but the internal details of major component are ignored

Issues in Architectural Design includes

- Gross decomposition of the system into major components.
- Allocation of funtional responsibilies to components.
- Component interfaces
- Component scaling & performance properties
- Communication & interaction b/w components

The architecture design adds important details ignored during the interface design.

Design of the internals of the major component is ignored until the last phase.

DETAILED DESIGN -

Design is the specification of internal elements of all major system components, their properties, etc.

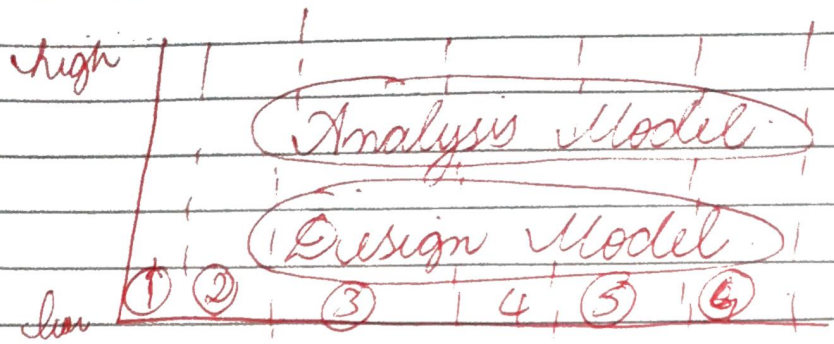
The detailed Design may include

- Decomposition of major system components into program units.
- Allocation of functional responsibilities to units
- User Interfaces
- Data Structures & Algos
- Unit States & State changes.
- Data & controlled interaction b/w units

DESIGN CONCEPTS.

1) Abstraction - hide irrelevant data
2) Refactoring - reconstruct something
3) Information hiding - hide info.
4) Pattern - a repeated form
5) Refinement - Removes impurity
6) Architecture - design a skeleton of something
7) Modularity - subdivide the system

DESIGN MODEL.



1. Data/class elements.
2. Architecture elements.
3. Interface elements.
4. Component level elements.
5. Deployment level elements.

* The design model can be viewed in two different dimensions.
* Horizontally the process dimension indicates the evolution of the parts of the design model each design task executed.
* Vertually, the abstraction dimension represents the level of details as each element of the analysis model is transformed into design model & iteratively refined.
* Elements of the design model use many of the same UML diagrams in analysis model.

* They are only refined & elaborated
* more implementation specific is provided

① Data Design elements:-

* It is also refered to as data architecturing
* Data design creates a model of data / info that is represented at a high level of abstraction.
* As in any projects data plays a key role hence designing it properly helps a lot in further (coding & testing steps)

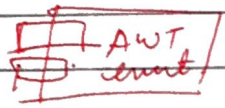
② Architectural Design elements-

* It is a photocopy of the end product to achieve that, follow.
  i) knowledge on the application domain
  ii) relationship & UML diagrams are drawn.
  iii) Proper architecture & styles

③ Interface Design element.
* Tells how info flow into & out of system
* Includes user interface, ext interface & internal "

④ Component Level Design.

* It provides all details of the given S/w component.
* In order to achieve this, the component design describes the use of data structure.



* The component will be involved into the Interface.

⑤ Development level Design

These indicate how S/w functionality & subsystems will be allowed allocated within the physical computing env. that will support the S/w

ARCHITECTURAL DESIGN:-

* S/w architectural design represent the structure & program components that are required to build a CBS.

* It is not a operational s/w code but its just a representation
* S/w architecture provides a uniform, high level view of the system to be built.
* It depicts the structure of the Software component, also properties & relationships

Importance of S/w Design

* A s/w architecture describes a scenario through which various components interact & communicate with each other.
* It provide an overview of each S/w dev aspect which remains important for overall success of s/w dev.
* Rep of s/w Arch are can enable for communication b/w all stakeholders interested in dev of CBS

DATA DESIGN:-

Data Design actions translate data objects into data structure further into database arch