SOFTWARE ENGINEERING

Name :- Yashraj Deepak Devrat

III[nd] Year :- Artificial Intelligence & Data Science.

Java : Platform independent because • Class file which created can be executed by any operating system.

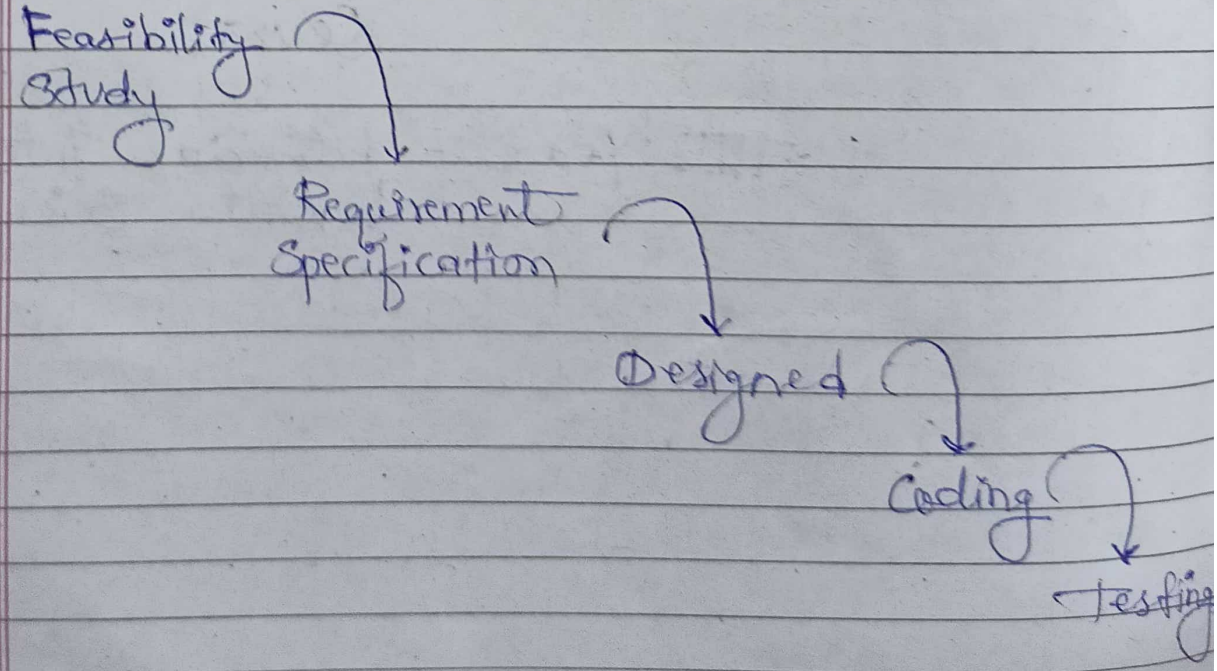
What is software Engineering ?

Software Engineering is a technique through which we can develop or create software for computer systems and any other devices.

Purpose of S/W Engineering :

1. To manage large software
2. for greater scalability
3. To manage the cost.
4. To manage the nature of the software.
5.

Waterfall Model :



Home Automation System.
Draw diagrams in Software Engineering paper
Use case diagram

Unit ① Introduction to Software Engineering and Software Process models.

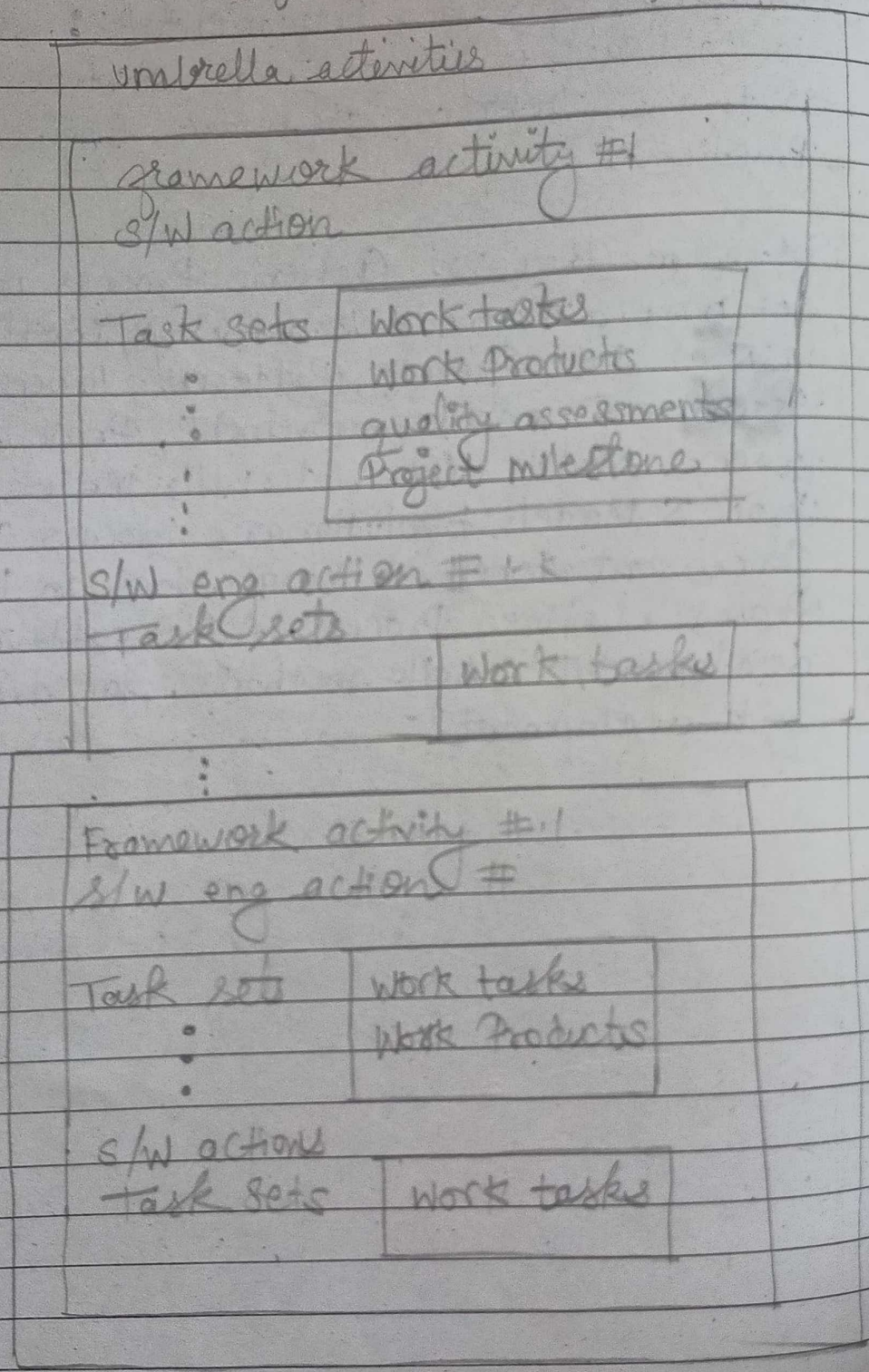
Software Engineering Fundamentals : Introduction to software engineering. The Nature of the software, Defining software Engineering, Practice.

Software Process: Generic Process model, defining a framework activity, identifying a task Set, Process Patterns, Process Assessment and Improvement, Prescriptive process Models. The Waterfalls Model, Incremental Process Models, Evolutionary Process Models, Concurrent models. A Final Word on Evolutionary Process. Unified Process. Agile software development : Agile methods, plan driven and agile development.

Generic Process Model:

Software Process

Process framework



Software Development Life Cycle or SDLC

Steps to be followed to design and develop a software product efficiently.

SDLC

- Communication
- Requirements gathering
- Feasibility study
- System Analysis
- Software Design
- Coding
- Testing
- Integration
- Implementation
- Operations & Maintainence
- Disposition

(1) Communication: Very first steps where user contacts the service provider is software organization and initiates request for a software product.

(2) Requirement Gathering: Stakeholders communicate with customers and system users to gather the business requirement like
- i Who will use the system?
- ii How user will interact with system?
- iii.

Depending on these Requirement Specification document is prepared.

(3) Requirements
- User Requirement
- System Requirement
- Functional Requirement.

Ways of Collecting Requirement
- Referring to database
- Studying Old software
- Interviews user & developer
- Collecting info about the

③ Feasibility Study: After Requirement gathering team analyzes if a software can be designed with the requirements gathered by the customer & users.

② Analyze that if the project is financially & practically feasible for developing it.

④ System Analysis: (# planning phase)
i. Developes decide the Roadmap of the plan and try to implement the same.

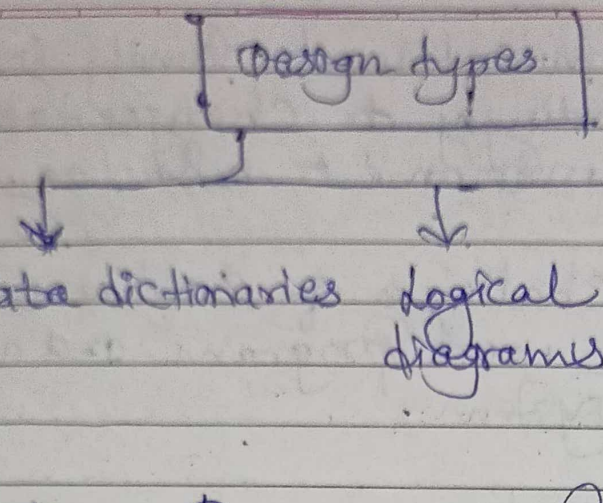
System Analysis Includes.
- Understanding of product requirement.
- Changes to be done in the existing system
- Identifying the impact of project on organization.

ii. Project team analyzes the scope of the project and manages it accordingly.

⑤ Software Design: i. Whole knowledge of the requirements & Analysis is taken together and and planned up for software product.

Input → From Users.
→ Information is gathered during requirement gathering phase.

Output Design → Logical Design
→ Physical Design.



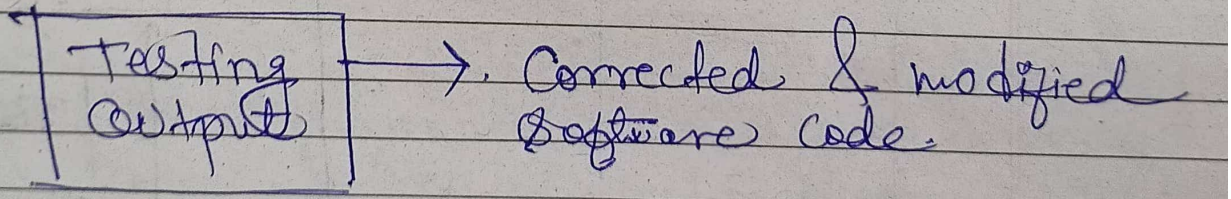
⑥ Coding : Programming phase.

It is the phase in which code are properly written in suitable programming language and developed an error free program.

⑦ Testing :

i. In this phase actual code of the system is tested by testers.

ii. Various techniques of testing are applied on the code like integration testing, unit testing, system testing.



⑧ Implementation : i. Software is installed on the system.

Check if it is portable, adaptable and integrable.

⑨ Operation & Maintainance :

This phase confirms operations in terms of more efficiency and less errors.

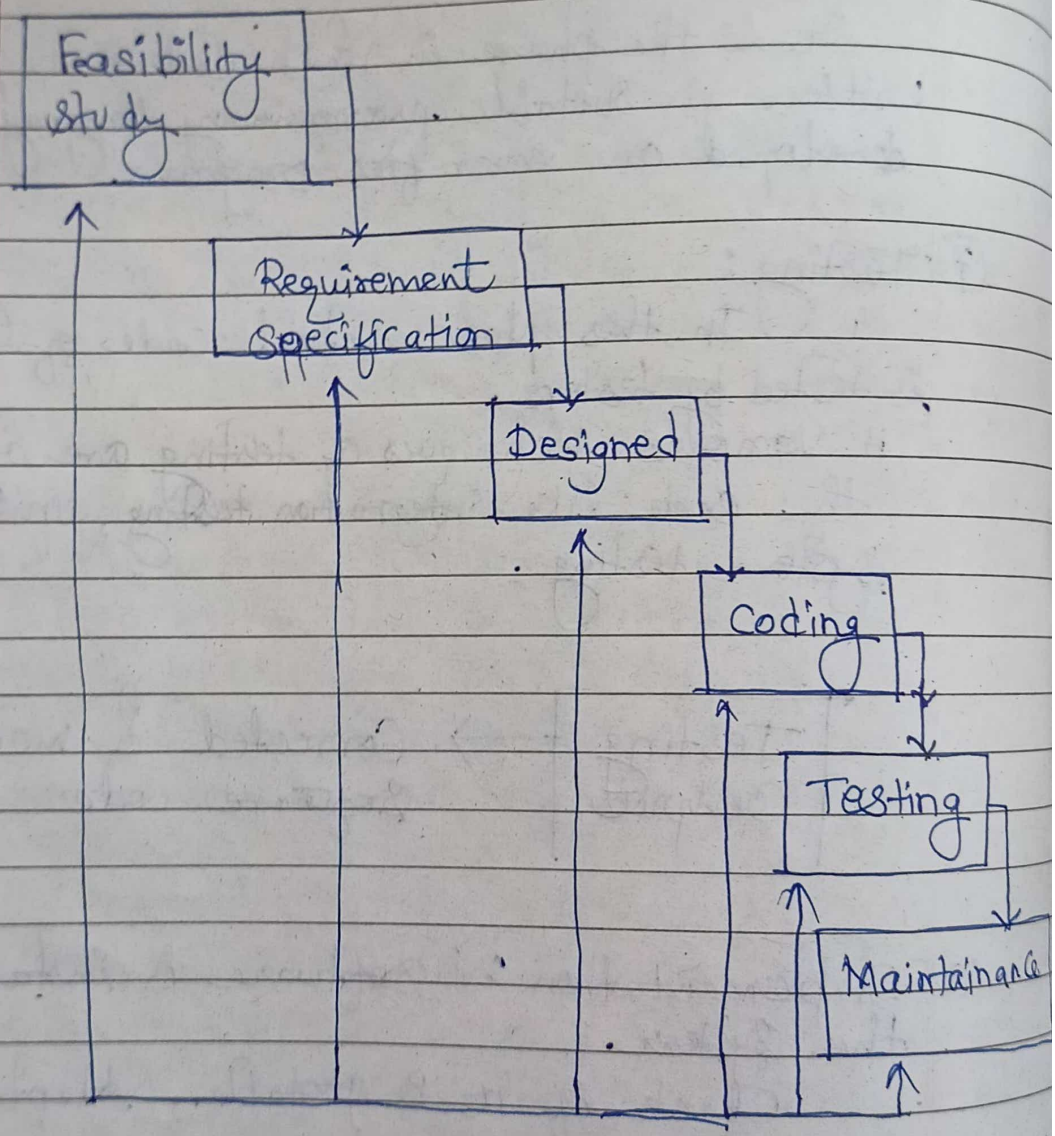
Waterfall model

output of step 1 will be input of step 1.

ii. Software is maintained timely by updating the code according to changes taking place in user and environment technology.

iii. It is the challenge of software development team to solve users problems and accordingly manage the system.

- Waterfall model:



Review.

(1) The waterfall model is the simple and classical model of all the model we have.

② This model is also known as linear sequential.

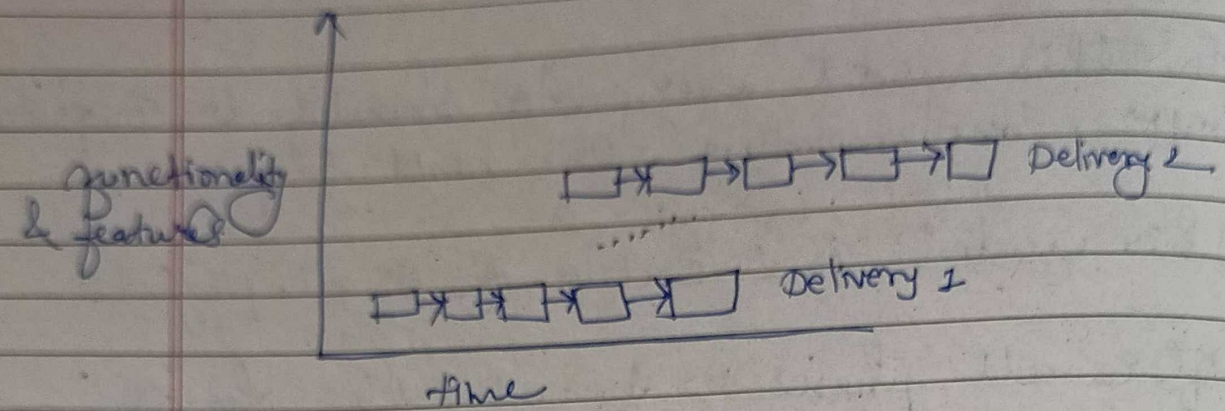
③ In this model every step should be completed before implementing the next step.

④ This model is suitable for small projects.

⑤ After all phases review all the phases.

- Requirement Analysis: All the business requirement are gathered for the project from the stakeholders, customers or user.
- Design: Based on the requirement a design of the system is created, which is architecture of software.
- Implementation: Actual code is constructed by the developers by use of software and hardware architecture of the system.
- Testing: Coding done by developers is verified that is it proper or changes are necessary for proper functioning of software.
- Maintainance: Developed software may generate bugs or problems in the software so the software development team should timely keep the maintainence.
  And also able to modify according the new requirements.

- Incremental Model:



i. Based on customers feedback, core product is delivered, which can be used by the customer.

ii. Next plan is developed for next increments, & modifications are made accordingly.

- Evolutionary Process Model

i. This model is based on the principle that ~~Suggests~~ stages consist of growing increments of an operational Software development product, with way of evolution being discovered by operating experience.

ii. According to the change in requirement passed by the user or customer need, improvements are done in the model.

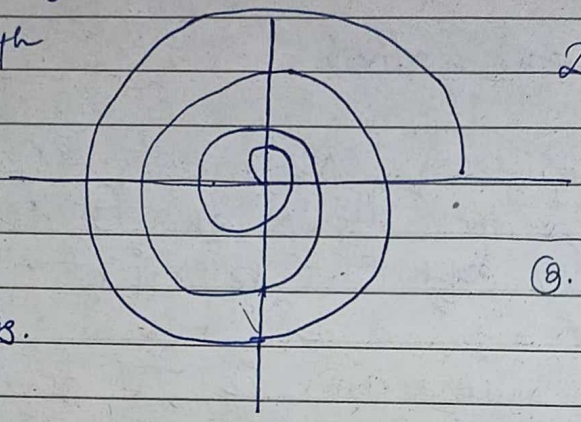
iii. As lot of improvement are required in the product hence this is iterative model.

Evolutionary Models.

- Spiral Model
- ~~Prototyping~~ Prototyping

Spiral Model purpose : Redue the Risk

① Spiral Model: i. Combination of both iterative & waterfall model.



Spiral Model.

(I) Identification: This phase identifies all business requirement of System at beginning. Clear understanding of requirements by communuth between Stakeholders & customer.

(II) Design: In first iteration, developers develop conceptual design of the System based on initially gathered requirements.

(III) Construct: In this Spiral, code developed by the developers are analyzed. Feedback of the customer is taken from the customer or users.

(IV) Evaluation & risk analysis: In this Spiral, risk associated with the model is identified.
i. Risk in Sake of potential of System.
ii. Budget of the System & important factors are evaluated.
iii. Customers are not sure about their requirements then Spiral model is usyl than waterfall model.

* Agile Methodology :

(1) Customer satisfaction by early and continuous delivery of valuable software.

(2) Changing in requirements, can be done at end stages also.

(3) Working Software is delivered frequently in less amount of time rather than other models.

(4) Here we can converse face to face.

(5) Simplicity is essential.

- Advantages of Agile Process :

(1) Stakeholder engagement.

(2) Transparency.

(3) Allow for Change.

(4) Early & predicatable delivery.

* JIRA :
  i. Web-based 'Issue tracking System' or 'Bug tracking system'.

UNIT (2) Software Requirements Engineering & Analysis.

Modeling: Requirements Engineering, Establishing the Groundwork. Identifying Stakeholders, Recognizing Multiple viewpoints, working toward Collaboration.

Asking the first Questions, Eliciting Requirements, Collaborative Requirement Gathering. Usage Scenarios, Elicitation Work Products.

Developing Use Cases, Building the Requirement model: Elements of the Requirement Model, Negotiating Requirements, Validating Requirements

(1) Requirement Engineering:

1. Requirement: Information which describes the user's expectation about the system performance is called as requirement.

Characteristics of Requirements:

(1) Requirement should be unambiguous.

(2) Requirement should be verifiable.

(3) Requirements should be clear.

(4) Requirement should be understandable.

(5) ——— be feasible.

(6) ——— be Consistent.

ii. Requirement Engineering: The procedure of collecting the users requirement about the software and then analyze, ~~and~~ document it is called Requirement Engineering.

Activities involved in requirement engineering

① Requirement inception.

② Requirement elicitation.

③ Requirement analysis & negotiation.

④ System modelling.

⑤ Requirement specification.

⑥ Requirement validation.

⑦ Requirement management.

- Business Analyst: The business analyst carefully analyzes user requirement and carefully conducts and documents a set of high quality system requirement ensuring that requirements meet certain quality characteristics.

Functional Requirements:

"Functional requirements specification create document regarding the operations as well activities which system must be able to carry out."

FR includes elements such as calculation, data manipulation and processing.

UNIT-3 Design Engineering

(1) Design Process & Quality

i. Once the required documents for developing the software is presented, the phase of software design gets started.

ii. In the design phase, customer, business requirements & technical considerations formulate a product or a system.

iii. Design model is assessed for quality & reviewed before generation of code & execution of tests.

iv. Design model gives detailed information, regarding software data structures, architecture, interfaces & components which are necessary to employ the system.

(2) Basics of Software Design.

i. Software design is considered as a phase in software engineering which develops a blueprint that will be a base of constructing software, system.

ii. Software design ~~components~~ encompasses the set of principles, concepts that lead to development of high quality of system or product.

iii. Design concepts must be understood before design practices are applied.

iv. Design practice itself leads to the creation of various representation of software.

v. It is guide for construction activity.

A. Firmness : Program should not have any bugs that inhibit its function.

B. Commodity : A program should be suitable for the purposes for which it was intended.

C. Delight : The experience of using the program should be pleasurable one.

Q① What are the important guidelines for evaluation of a good design ?

Q② Explain the quality attributes, considered in software design ?

Q③ Explain the following design concept :

① Abstraction.

② Architecture

③ Patterns

④ Modularity.

Q4. What are the elements of Component-level design for WebApps ?

Q5. Explain the Software Architecture ? what is Architecture Important ?

Q6. Explain Data-Centred Architectures & Data-flow Architectures ?

#1 Design Quality Guidelines:

1. A design should be modular in nature.
2. A design should contain appropriate data structure, recognisable patterns.
3. Design should represent data, architecture, interface and components in distinct way.
4. A design should ~~create~~ use notation that must lead to effective communication and understanding of design.
5. A design should use application, design principles and proper methodology.

#2. Quality attributes:

Qualities of good software design include:

1. Innovative: New design gives unseen value to market and redesign ~~includes~~ improves the quality of existing product.
2. Functional: New functionality improves the the performance and quality of product. Optimization is done for making new product.
3. Honest: Promise kept for updating & modifying is fulfilled. Good software design functions properly.

④ Supportability: It includes the extensibility software will be extendable in future it can accomodate changes.

⑤ ~~Portability~~ Reliability: Program should be able to recover from failures and give accurate output.

#3. Explain the following Concepts:

① Pattern: We use pattern to identify solutions to design problems that are recurring and can be solved reliably.

② Abstraction: When we consider a modular solution to the problems, there are many levels of abstraction possible.

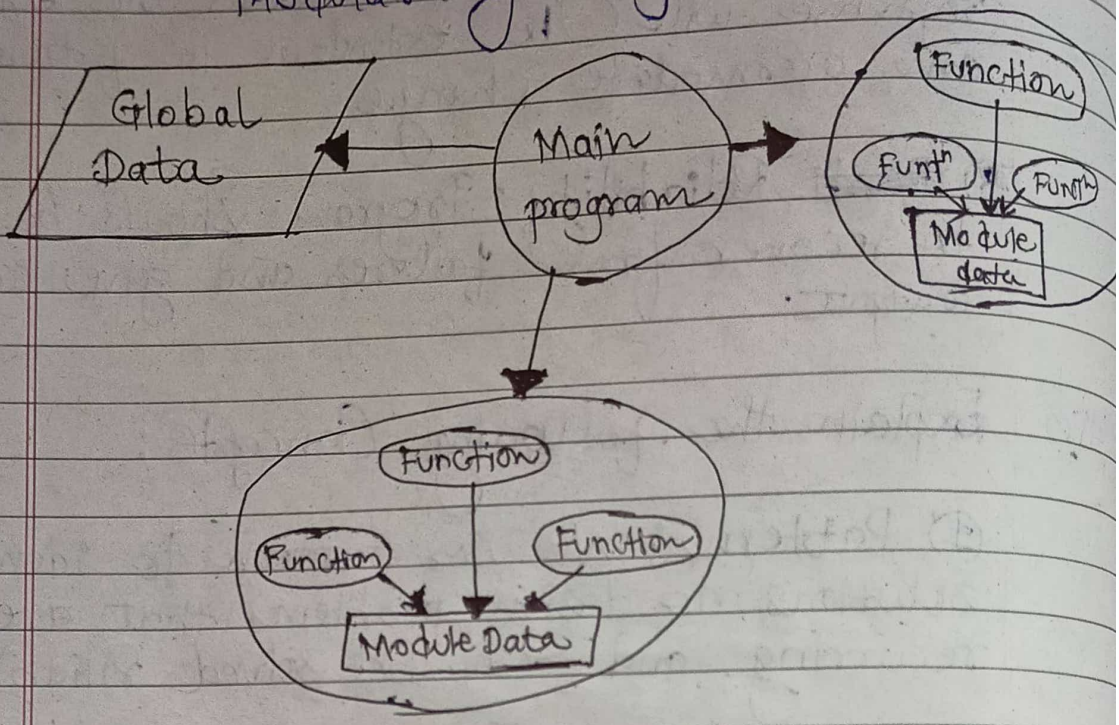
③ Data abstraction is actually a collection of data that describes a data object.

#4. ④ Architecture: Software Architecture provides detailed description of Conceptual integration for a system and complete description of a software.

Main goal of the software design is to provide an architectural framework of a system and can be represented using one or more number of different models.

⑤ Modularity: Modularity is modularization of one module of software into number of small parts that can be easily managed.

Modularity Diagram



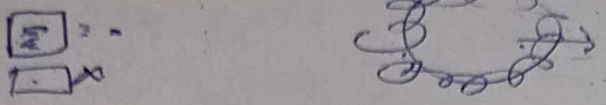
Major Criteria of modular System:

(1) Modular decomposability: Complex problem can be solved easily by decomposing into smaller problems.

(2) Modular Compasability: Different types of modules are integrated to make a new a new system.

(3) Modular Understandability: While integrating a system, if each module is understood well then a system can be build in efficient way.

(4) Modular Continuity: Changes should be made module wise instead of making changes system-wise.



#4 Functional Independence :

#5 : Refinement :

i. It is the process of elaboration.
ii. In each step, one or several instructions of a given program are decomposed into more detailed instructions.
iii. Refinement is the movement from higher to lower levels.

#6. Refactoring :

i. It is the process of changing a software system in such way that it does not effect external behaviour of code and improves internal structure.
ii. It improves design of existing code.

Importance of refactoring :

i. Refactoring helps code understanding.
ii. Error handling is easy.
iii. Programmer do not write perfect code, so refactoring helps in improving the code overtime.

#7:

Abstraction	Refinement
i. Provides compact design process.	i. Provides flexible design process.
ii. Description of something that omits some detail	ii. Detailed description that is even not relevant

Architectural Design

Q① Explain: Architectural Design.

Q② Discuss architectural patterns in detail.

Q③ Explain basic design principles.

Q④ What do you mean by Software architecture

Q⑤ Short note: Modeling Component level design

Q.6. Explain: Basic Design principles.

① Introduction to Architectural Design.

- Architectural design is backbone of any software system and is responsible for overall data structure of the system.
- In almost all the models of software process, architectural design is considered as first stage of development process.
- Output of architectural design process is an architectural model that explains the overall data structure of the system and also explain the working of the system.

★ Functions of Architectural Design:

1. It describes overall function and performance behaviours of the system.
2. It develops and documents top-level design for the external and internal

interfaces.

#② Architectural Patterns:

Architectural patterns are set of system which are internally linked with each other and share structural properties.

① Data-Flow architecture:

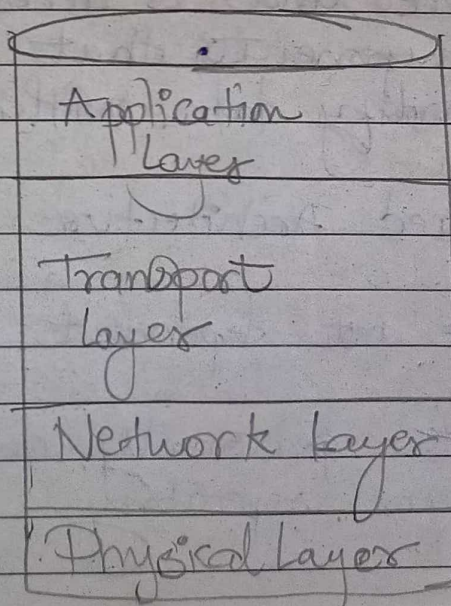
Main purpose of Data-flow architecture is to accept some input and transform it into required output by applying several transformations.

② Object-Oriented Architecture:

In this pattern, components are represented as objects and they communicate with each other through methods.

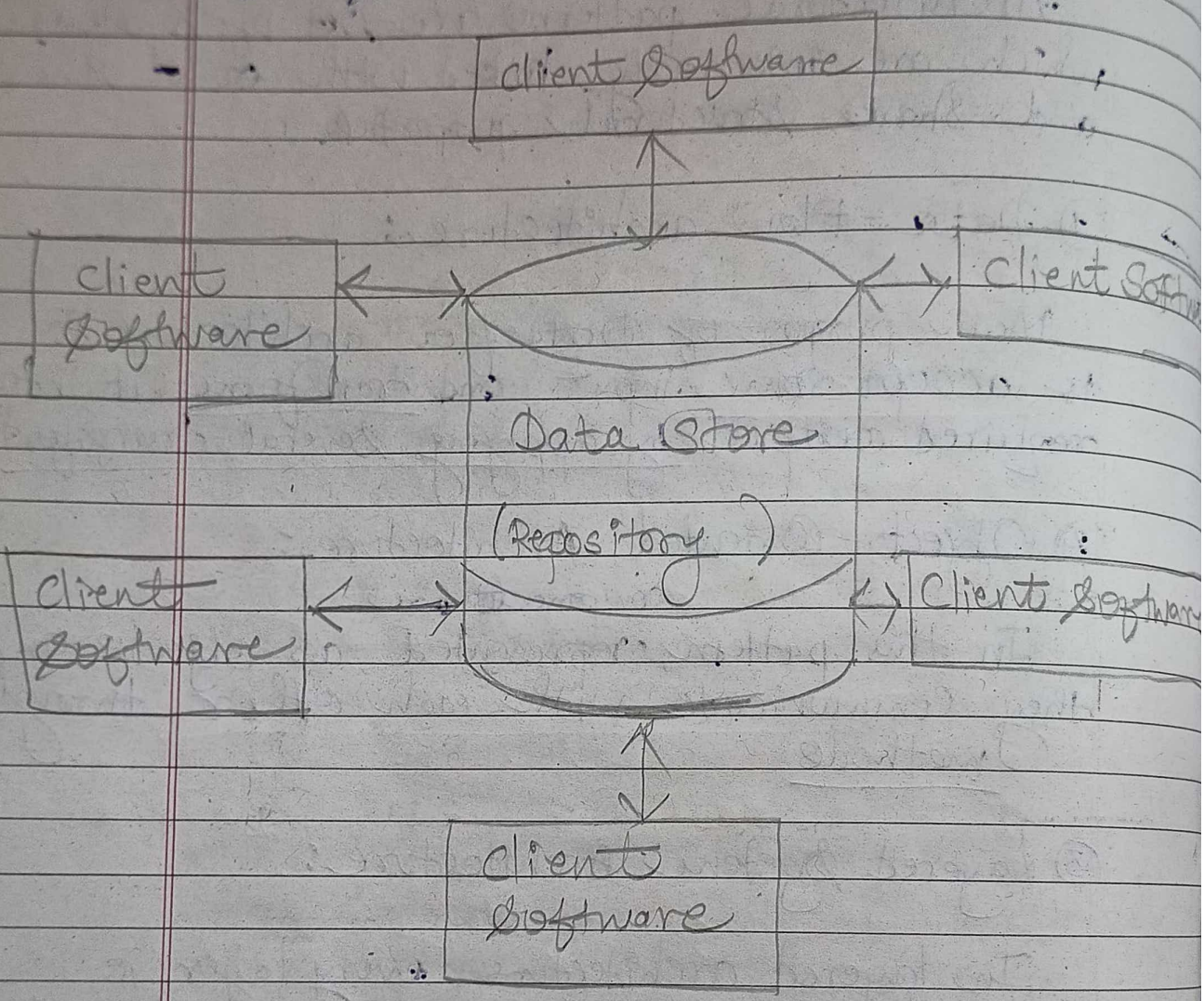
③ Layered System Architecture:

In layered architecture, every layer is responsible for performing a well-defined set of operations.



* ④ Data-Centred Architecture:

- A central data structure is also known as data store.



- Data Store is at centre of this architecture and is accessed frequently by other components that update, add, delete, modify data within the Store.

★ Data Centred Architecture Advantages.

① Client are not dependent on each other.

② Client work personally and deposit the data in Data Store repository.

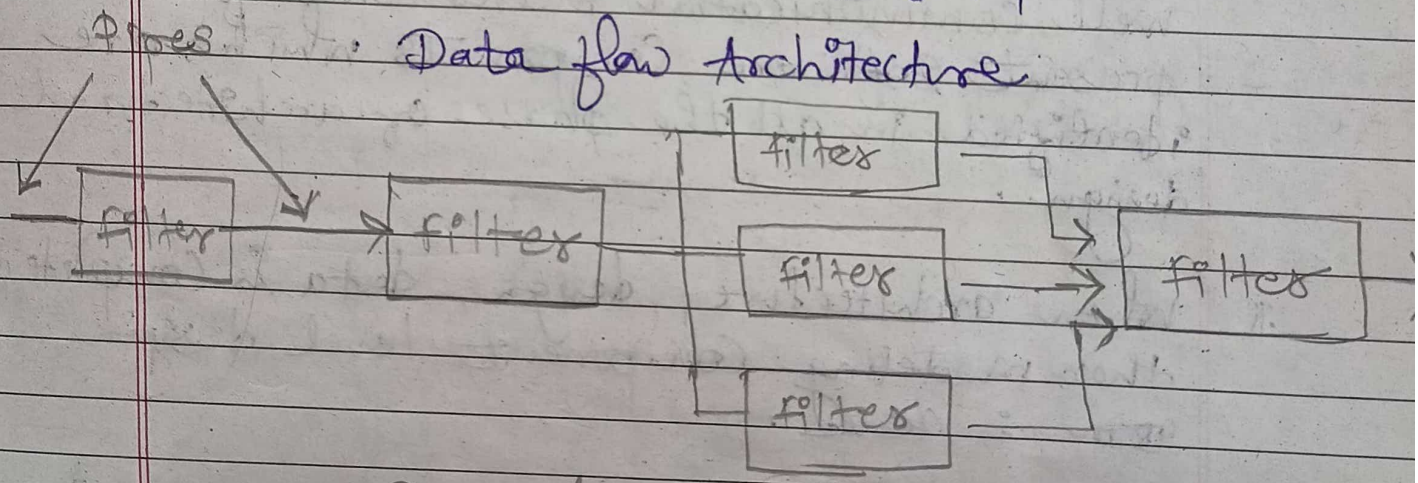
③ Adding new clients can be easy.

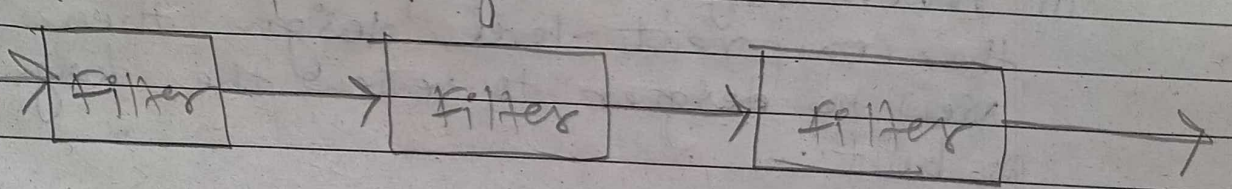
④ Modification is easy.

② Data Flow Architecture:

Main purpose of data-flow architecture is to accept some inputs and transform it into required outputs by applying several transformation

1. Each component act as filter and accept the data and transforms it and then passes to next filter with the help of pipe.





Both Sequential

Advantages

① It supports reusability, modifiability, maintainability.

② Concurrent execution is supported by this cycle.

Disadvantages:

(1) It is ~~co-ordina~~ not easy to co ordinate 2 different ~~filters~~ streams.

(2) Applications which need more user interaction is not supported by data-flow architecture.

#(3) Modeling Component level Design.

i. Modeling Component level Design: Main function of this is to define data structures, algorithms, interface characteristics well communication mechanisms for all present Software Components which are identified in all the phase of architectural design.

ii. When architecture, design, data is Completed then modeling Component level design ~~is~~ is done.

iii. When Software is ~~represen~~ represented by the component-level design then it gives view to designer to review the design which is built.

#(4) Explain: Basic Design Principles:

(1) The design process should not suffer from "tunnel" vision".

(2) The design should be traceable to the analysis model.

④ Design should be flexible and able to accomodate changes.

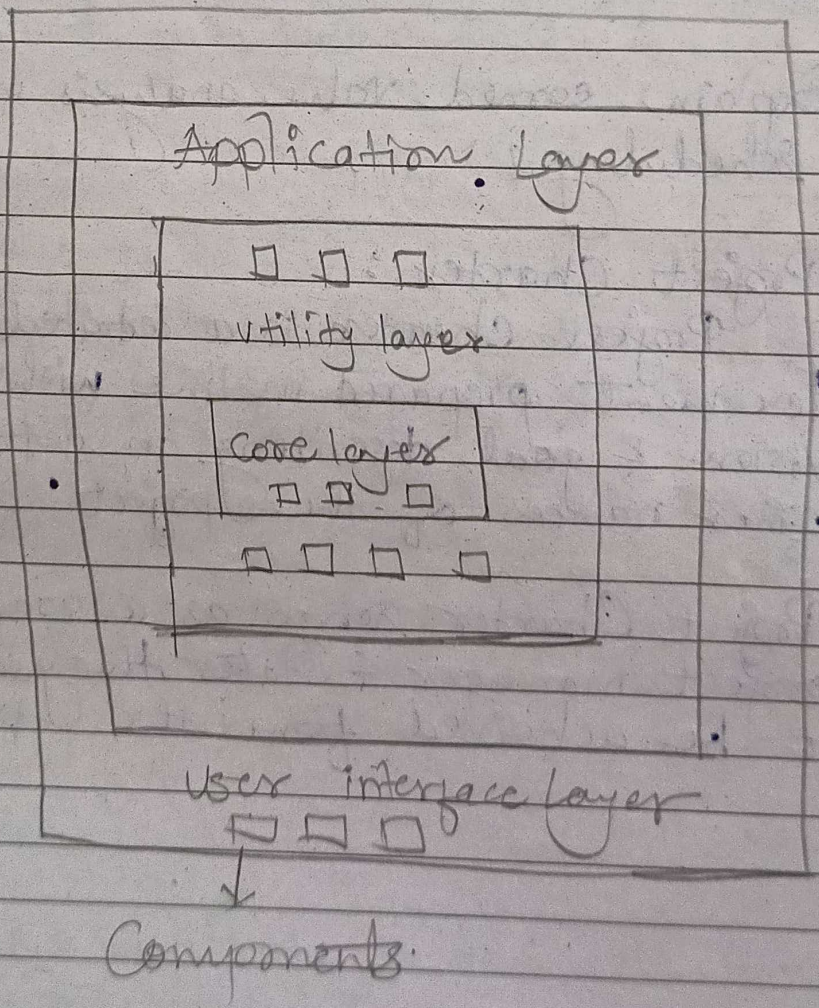
④ Design should not reinvent the wheel.

⑤ Design should be reviewed to minimize the conceptual errors.

⑥ Design should have Uniformity & Integrity.

#5 Component Level Design for Web Apps:

#6 Layered Architecture:



UNIT-3 (Estimation)

Q1 What is project scheduling ?

Q.2 What are basic principles of project scheduling ?

Q.3 Define time line chart ?

Q.4 How time line chart is used in scheduling of software project ?

Q.5 Short note on: Schedule Tracking Tools.

Q.6 Short note on: DART.

Q.7 Explain earned value analysis in project scheduling ?

#1) Project Charter :

Project Charter is a detailed official document prepared in line with company's vision & goal describing in details the finer nuances of the project.

Project Charter serves as a road map for the project manager & states the goals that are to be achieved from the project.

Q8 Explain the quality attributes, considered in software design ?

Q.9 What are the categories of Software design engineering resources ?

Q.10 Explain LOC - based estimation ?

Q.11 Explain COCOMO model for the project cost estimation ?

Q.12 Explain decomposition techniques ?

Q.13 Draw & explain the traceability table for requirement management ?

#(1) Attributes of a project :

Project : It is an undertaking task for non-repetitive nature having some objectives w.r.t time, quality & cost.

Following are the project attributes :

(1) Purpose : A project must have unique style and purpose that need to be implemented. Sole purpose of undertaking a project is to produce a tangible product of value to the company.

(2) Goal : Goal defines the, each activity, task, work and budget of the project. It is a motivating factor.
Project goal must have leading factor & vision otherwise project without goal is of no end.

0.52
0.65
1.17

0.01
82
0.02
3 0.

③ Time Frame: It is a component which gives start and end to our project. Completion date for the project is set accordingly.
Projects where completion date is fixed, the starting date is to be set backwards.

④ Ownership: Every project has project owner. Though project has various stakeholders, it has only one primary customer known as project sponsor.

⑤ Risks & assumptions: Every project has certain risk elements, the more complex & big project as a result risk is greater.

Risk & Size, Complexity

The project manager has to identify all the risks and assumptions that could impact the project.

#2 Resource Management.

i. All the elements required for developing a software can be considered as resources for the project.
ii. This include human resources, productive tools, software libraries.
iii. Allocating extra resources increase development in the cost at end. So it is therefore necessary to estimate and allocate adequate the resources for the project.

$F.P = \text{count total}\left[0.65 + 0.01 \sum (fi)\right]$

- Resources Management Includes:
  - Defining a proper organization project by creating a project team and allocating responsibilities to each team member.
  - Determining resources required at a particular stage & their availability.
  - Managing resources and generating. Generating accordingly.
- Hardware & Software Tools.
  - i. Each & every project manages knows that the main Components needed for completion of project are Hardware & Software tools.
  - ii. This tools help in monitoring, tracking, alerting, trouble shooting and Correcting.
- Count the lines

There are several ways to count the lines in the code. Depending on what you count, you get a high or low line count.

#3 Lines of Code (LOC).

LOC measures the size of project by Counting the number source instruction in the development program ignoring the commenting code & header lines.

★ Function Point Metrices (FPM).

This metric overcomes Shortcoming of the LOC Metric

Q Why we use it ?.

We can directly estimate our project size from problem specification.

i. We cannot measure functionality directly. Hence it can be derived indirectly.

Five information domain characteristics are determined and counts are provided.

① Number of user inputs : Each user input should be considered.

② Number of user outputs : Reports, Screen, error messages, etc.

③ Number of user inquiries : An online I/p which results on online o/p.

④ Number of files : Each logical master file is counted.

⑤ Nr —— external interfaces : Data files on Storage media.

Q. Write short note on: Component level design.

→

① Identify all of the respective design classes which are corresponding to the problem domain.

② Identify ——— infrastructure domain

③

Q8. Explain COCOMO model with example.

→ i. COCOMO (Constructive Cost Model) has a set of 3 development modes, 3 levels.

COCOMO model levels:

① Basic Level: only predict Software Size used to estimate the development effort.

② Intermediate level: Consist of predicted Software plus a set of subjectively assessed cost drivers used to estimate development effort.

③ Advanced level: In addition with intermediate level model this model allows phase based cost driver adjustments, and also some adjustments at the module, component and system level.

C.O COCOMO Development modes:

① Organic Mode: In this mode small & simple projects are handled in which small and simple teams with good application experience work to set of flexible require

② Embeded Mode: Used for Software projects that have tight Software, hardware & operational constraints.

③ Semi-detached: used for intermediate size projects in which teams with mixed experience levels have to meet mix of rigid and less rigid requirements.

Basic COCOMO model computes Software development & cost of a function of program size expressed in estimated lines of codes.

Formula:

$$E = a(S)^b \quad \text{KLOC}$$

E represents effort in person-months.
S is the size of Software development measured in KLOC.

a & b values are dependent on development mode.

Short note on: Shedule Tracking Tools

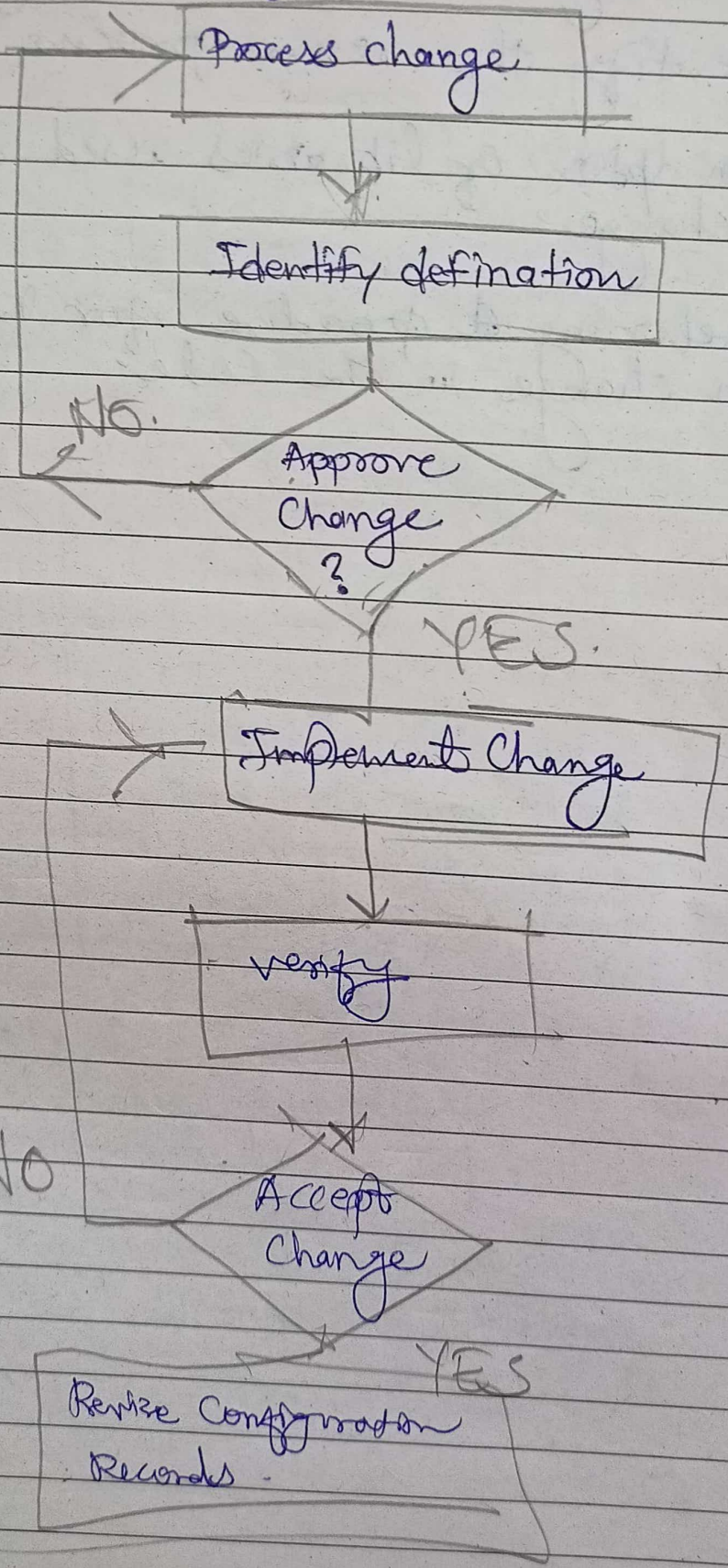
→ i. Microsoft project is a project management Software program, developed & sold by Microsoft.

ii. It is designed to assist a project manager in developing a plan, assigning resources to task, tracking progress, managing budget, and analyzing workload.

iii. The Microsoft Office family, it is available currently in 2 editions Standard & professional.

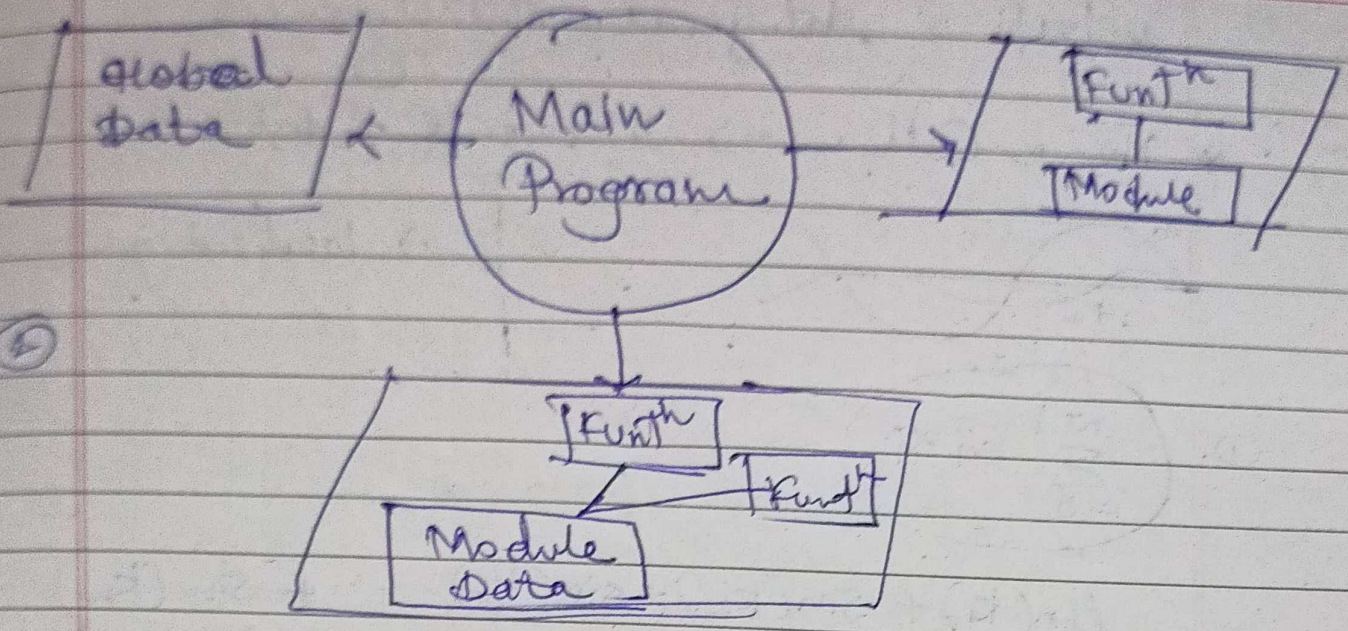
9. Software change control Mechanism in SCM.

→

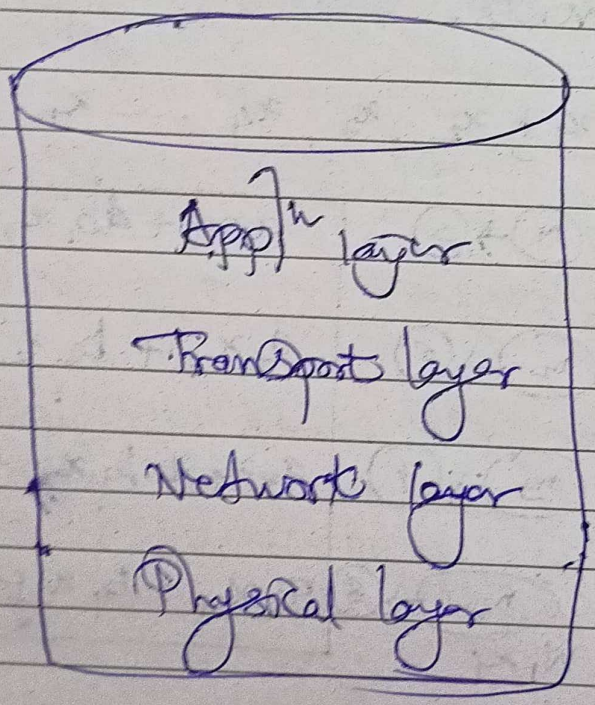


Minimum Activities needed for processing changes include:

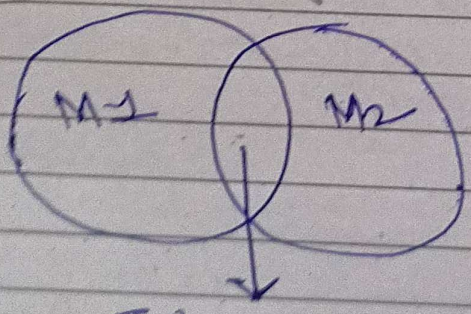
1. Defining change information needed for approving the requested change.
2. Identify the review process
3. Description of libraries used to process the change.
4. Developing a procedure for implementing each change in the code.
5.



(2)



(3)



Integration Testing

UNIT 4 Design Engineering

27/3/24

Design

- Mitch Kapor, the creator of Lotus 1-2-3 presented a "software design manifesto" in Dr. Dobbs Journal. He said:

Good software design should exhibit:

firmness: A program should not have any bugs.

commodity: A program should be suitable for the ~~program~~ purposes for which it was ~~to~~ intended

Design Delight: The experience of using the program should be pleasurable one.

- Divergence & Convergence: Diversefication
- Analysis Model → Design Model.
- Design within the context of Software Engineering.

The architectural design defines the relationship between major structural elements of the software, the architectural styles & design patterns.

- The framework of a computer based system:

P1 P2 P3

Scheduling

② Project scheduling ~~men~~ methods.

- Time-line chart
- Gantt chart
- Use of ~~chart~~ is use track the progress of our project: table schedule
- Timed box.
- Object Oriented approach
- ⑧ Progress on an OO Project - 1.

Technical milestone: OO analysis completed, OO testing

* Implies a flow of information and a specific type of behaviour.

- The component-level design transforms structure elements of the software architecture into a procedural description of software components.

- Design provide:
  - quality in software engineering
  - representation of software
  - Way to the accurately translate Stakeholder's requirement into a finished software product or our system.

- Design & Quality
  - Requirement are translated into a "blueprint".
  - Represented by at a high level abstract system objective and more detailed data, functional & behavioral requirements.

Characteristics of a good design:

- The design must be implemented all of the explicit requirements contained in the analysis model, and it must accomodate all of the implicit requirement desired by the customer.

- Component design, Quality Design.
- Guidelines for quality design.

1 Software evolutionary fashion.

2 Has been created using recognizable architectural styles or patterns.

- Design module: should be logically partioned into elements or subsystem.
- A design should contain distinct representation of components, architect
- Interdependency of module & functional characteristics.
- Design should reduce the complexity of connections between components.
- A design should be derived using a repeatable method.

Quality attributes. / Technical attributes

- FURPS: functionality, Usability, Reliability, Performance, Supportability

1. functionality: capability of the program.
2. Usability: consistency, documentation

3. Reliability : Recovery, failure of the software, predictability of the software or program.

4. Performance : speed of software, efficiency, processing.

5. Supportability : contains the ability of the program. modification, adaptability, quality of the software.

* Fundamental concepts of object oriented Programming languages.

- Software development

- Abstraction, Procedural ~~Abstraction~~, Data abstraction.

Architecture : Properties of architectural properties of architectural design

① Structural properties :

② Extra - functional properties :

③ Families of related system :

- five (5) types of architectural design.

Patterns

* Separation of concerns.

- Solution Pattern.
- Smartly reduce the time, into smaller & therefore manageable pieces of concern.

* Modularity.

- modularity is then single attribute of the software that allows the program to the ~~instantly~~ intellectuality manageable.
- monolithic software (cannot be grasped ~~as~~ by a software program.)

- The number of control path, span of reference, number of variables and overall complexity would make the task impossible.

- In almost all instance, you should break the design into many modulas, hoping to make the understanding easier and as a consequence, reduce the cost required to build the software.
- Number of module should be in limit so that it can be tracked accordingly.
- Modularity - Trade off.
- Information ~~for~~ hiding:

- Why information hiding ?
  - Reduces the likelihood of 'Side Effects'.
  - Limits the global impact of local design decisions.
  - Emphasizes the communication through controlled interfaces.
  - Discourages the use of global data.
  - Lead to encapsulation :- an attribute of high quality software.
  - Results in higher quality software.
  - Inadvertant errors are less likely to propogate.

WT # 4. Functional Independence

- Functional Independence is achieved by developing modules with "single-minded" function and an "aversion" to excessive interaction with other modules.
- Cohesion is an indication of the relative functional strength of the module.
  - A cohesive module perform a single task, ~~req~~ require

Protocol book

Class diagram
Data flow diagram
control flow diagram

13/4/22

- Stepwise Refinement
- Aspects, Refactoring. (used for increasing efficiency)
- Object Oriented Design Concepts.
- Types interface classes.

① User Interface classes :- Define abstraction for human computer interaction (HCI)

② Process Classes :

③ Persistent Classes :

④ System classes :

⑤ Buesness Domain classes

- Classes define four characteristics of a well performed designed class.

① Primitiveness
② High Cohesion
③ Low coupling
④ Complete and Sufficient.

** The Design Model :

① Analysis Model

② Design Model :

③ Data Design Elements :

Data Mode --> Data Structures

- Refined into progressively is more implementation specific.
- Essentiability of high quality to the program (a) or component level application.
- Database -> Architecture
- At the application level, Database is important.
- At business level, a stored database is used & recognized database into data warehouse, enables Data Mining.

Architectural Design Elements.

- Overall view of the data structure and software.

* Derived from:

  ① Application Domain: Project belongs to which domain.

  ② Analysis classes, their relationships, collaboration.

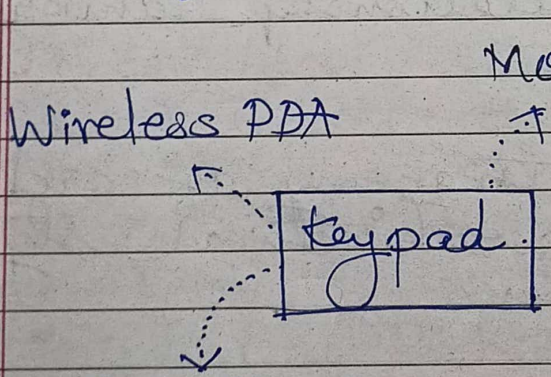
  ③ Patterns & "styles".

- Architectural System: A set of interconnected system.

* Interface Design Elements:

- It gives the information flows out and how the information is communicated among the components.
- The User Interface (UI): Color, design elements, UI navigation, technical elements interaction mechanisms.
- Internal Interfaces between various design components.
- Only necessary information is provided from the system: Internal structure of the system is generally hidden from the outside world.

* Interface elements.



* Component-level Design Elements.

- Description of the Internal software component.
- All Data structure, Algorithm, with objects for all processing.

- Activity diagram can be used for processing logic.
- Sometimes Pseudocode or some other diagrammatic representation can be used for component level design.

Deployment-Level Design element

- How actual software functionality and subsystems is done on computer system of computing environment.

Architectural Design:

- Structure of the design is the base and accordingly the components ~~are~~ like Software component, the externally visible properties of those components; relationship amongst them.

- Communication, taxanomy of architecture, Data Central Architectural styles, Call & Return.

① Main Program

② Sub-program

③ RPC: Remote Procedure Call.

Object Oriented Programming Designing ~~Styles~~ Architectural Style.

- Layered Architectural Style

Risk Components :

- Performance Risk : Uncertainty that product meet requirements
- Cost Risk : Budget of the degree of uncertainty will be maintained.
- Support Risk : Software will be easy to correct & enhance.
- Schedule Risk : Uncertainty that the project will be done in time and maintained properly.

Risk Projection (Risk estimation)

attempts to rate each risk in two ways

- Four risk projection steps :

1. Establish the Scale
2. Declinate the consequences of the risk.
3. Estimate the impact of the risk.
4.

- Building a Risk Table.

1. Estimate the probability of occurence
2. Estimate the impact of the project on scale 1 to 4.
3. Sorts the table by probability ? accordingly to the risk.

- This accomplishes first order risk prioritization
- Define the cut-of line ~~and~~ and then predict the
- Diagram of occurrence of probability
- Risk Exposure (Impact)

1. If the risk occured then 3 factors: Nature, Scope & timing.
2. Steps to determine the overall consequences of the risk:
   i. Determine the average impact of the risk.

- Risk Identification:
- Risk Impact:
- Risk Exposure formula:
- Risk Probability:
- Risk Refinement:
- Risk mitigation, Monitoring & Management:

  Mitigation: How can we avoid the risk?
- Reactive & proactive Risk

- Monitoring: Continuously observing the risk.
- Management: What contingency planning if the risk becomes in reality.
- Risk may increse the cost of the projects.
- Risk information sheet:

28/4/25 * Software Configuration Management:

- Software Configuration Management is a set of activities that have been developed throughout the cycle of the Computer Software.
- Software Configuration Management refers for the changes before handing it over to the client.
- Output of the software process:

Software Configuration items:

First Law of System Engineering:

- Changes, requirement, for Software development.

- An SCM Scenario
- goals and activities required for the goal achievement
- Elements of Software Configuration Management:
  Component element, process element, Construction elements, Human elements.

Baselines:

- Software Configuration Items:

02/5/20 SCM Repository

- Software Configuration items were maintained as per requirement:

1. Finding the configuration item when it was needed was often difficult
2. Determing the challenge of the items which were changed by whom & when
3. Constructing the ~~old~~ new program by modifying the old one.
4.

- It is mechanism that help Software mangers to easily use & allow Software team to mange change in effective manner.

Repository performs the following functions

- Data Integrity, Data hiding, information sharing, Document Standardization

- Meta-Model : Total information & details of the file.
- General features & content
  Main purpose :
  (1) What is stored in repository.
  (2) What services are provided by the repository
- Classes of Services :
  1. Database Management System Services
  2. Services that are specific to the Software environment.
- Repository Features
  1. ~~Integrate~~ Versionality / versioning.
- Dependency Tracking & Change management
- Requirement tracing
- Configuration Management
- Audit Trials

SCM process model :

Objectives :

- Version control.

4/5/22 Change control Process - 1

- Legacy Software
- need for change requirement

SCI → Software Configuration Items.
Black Box, SQA → Software Quality Assurance

- Change Control Process - 2
  ↓
  assign people to SCI's
  ↓
  checkout SCI's
  ↓
  make the change
  ↓
  review the changes
  ↓
  establish a baseline for testing

Change Control Process - 3
↓
Perform SQA and testing activities.
↓
Check in the changed SCI's
↓
promote SCI for change.

* AUDITING

- Status Accounting
- SCM - for web - Engineering - 1

1. Content : An array of content: videos, graphics, scripts, audio/video files, forms, active page,

- Scalability :

Content Management - 1

- Operating Systems is the major content of MIS and resources.
- Content Management System (CMS).
- Customer Relationship Management (CRM).
- Enterprise Resource Management (ERM).
- Operating System.
- Web Servers
- Any other apps.

Software Selection: It is more critical than hardware selection
depends on whether software is centralized or decentralized.
Easy Maintainence.
Compatibility.

- Major levels of Software:

① System Software:

② Communication ~~System~~ Software: for testing Software and Communicating with Colleagues.

③ Application Software: productivity Software.

Software platforms:
- Open Source Software
- Linux.
- Java.
- Web Services.