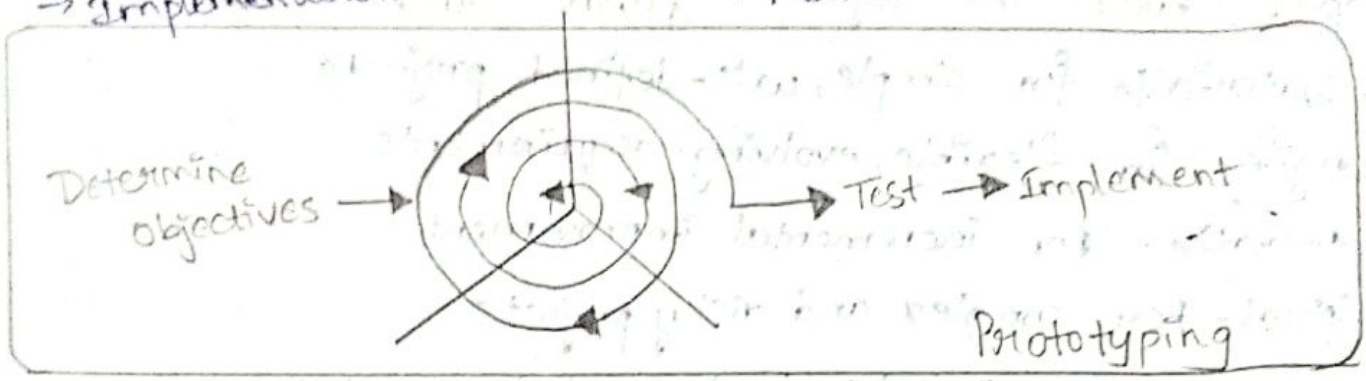
UNIT-II: PROJECT LIFECYCLE & EFFORT ESTIMATION.

* Explain about Software process.

A) A software process is a collection of activities, methods, practices and transformations used to develop software products.

> Common software processes are
- → Requirements Gathering
- → Design
- → Implementation
- → Testing
- → Deployment
- → Maintainance.



Activities in Software Process:-

> Requirement Gathering & Analysis:- understanding user requirements
> Design:- structure how the software fulfill these needs
> Implementation (coding):- writing the actual code.
> Testing - Ensuring the software work as expected.
> Deployment - Making the system available for use.
> Maintainance - Fixing issues and making improvements over time.

→ Some of the process models are waterfall model, V-model, Agile model & Spiral model.

Advantages:-
- > Structured and easy to understand.
- > Documentation is thorough
- > phased approach allows for clear understanding
- > Easy to manage.

Disadvantages:-
- > Requirements cannot be changed midway
- > Delays in testing
- > Not suitable for large projects.

* How to use Software process Models for project Implementation

Software process models are structured approaches to software development that guide the planning, execution of systems.

Following steps are how to use Software models for project Implementation →→.

Step 1:- Choose the right Software Process Model.

- Waterfall - for simple, well-defined projects.
- Agile - for flexible, evolving requirements.
- Iterative - For incremental improvement
- Spiral - For complex and risky projects.
- V-model - for projects that need heavy testing.

Ex:- For building a library management system choose the waterfall model.

Step 2:- Follow the phases of chosen model.

let us take waterfall model as an example.

> Requirement Analysis
- Meet with stakeholders (admin, staff, students)
- Gather all functional needs (catalog reposits, fines)
- create a SRS document.

> System Design.
- create system architecture, database schema.
- Tools - Figma, MySQL workbench.

> Implementation (coding)
- Start coding based on design documents.
- Divide project into modules.
- User registration/login.
- Book search
- Fine calculation
- Admin Panel.

Tools: Java / python / php / + MySQL

> Testing.
It includes:
- unit testing – check individual modules.
- Integration Testing – check interaction b/w modules.
- System Testing – Test full System.
- Tools: JUnit, Manual testing checklist

> Deployment
- Host system on a server and make it available to users.

> Maintainance.
- Monitor performance, fix bugs, update features based on user feedback

Step 3:- Track progress & Document Everything
- Use tools like Trello / Jira – task tracking
  Github – version control
  Google docs / word – documentation.

Step 4:- Sample Implementation plan Table.

Phase	Description	Tools Used	Output
Requirement	collect & define user needs.	SRS document	functional specs.
Design	UI, DB, architecture planning	Figma, Draw.io	Design specs,
Implementation	Develop and code modules.	IDE, Git Hub	Source code
Testing	Test modules and System.	Manual, JUnit	Bug report, Test cases
Deployment	Deploy on live/local server	XAMPP, cloud	working software
Maintainance	Fix, update, enhance.	Feedback loop.	updated version logs.

Outcome:- If we followed step by step we'll have:
→ A well-structured project → clear documentation.
→ Better quality control. → Higher scalability

* Why Model Selection is important?

Model selection is important because it aligns software development with specific goals and nature of project.

> It enhances team coordination by defining responsibilities.
> It is essential for maintaining productivity
> Selecting right model aids in managing complexity and meeting delivery timelines.
> It leads to higher-quality product & increased customer satisfaction

Criteria for choosing Software Process Model:-

> project size and complexity.
  - Small, simple projects – waterfall or v-model.
  - Medium to large projects – Spiral, Agile or Iterative.
> clarity & stability of requirements
  - clear requirements – waterfall
  - uncertain requirements – Agile.
  - incrementally defined requirements – Spiral.
> Time to Market
  - short deadlines – Agile
  - Flexible Timelines – Spiral
  - Fixed, longterm plans – waterfall.
> Level of Risk
  - High risk – Spiral
  - Moderate risk – v-model
  - Low risk – waterfall.
> customer involvement & feedback
  - Frequent customer involvement – Agile.
  - Minimal customer involvement – waterfall
  - Feedback based on risk discovery – Spiral

Sample project – E-commerce website (Agile Vs Spiral).

It includes: • product listings • user accounts • Admin panel
• shopping cart • payment integration • order tracking

Agile model for e-commerce website

> used when requirements change frequently
> development is broken into sprints (2-3 weeks)
> Quick updates
> Better adaptability
> Best for dynamic markets

Spiral model for E-commerce website.

> Used when security and scalability are top concerns.
> Each phase is repeated with refinement.
> Risk analysis performed.
> More Controlled than Agile
> Best for healthcare systems.

* Advantages
→ Flexibility → Informed choices → Risk awareness.

* Disadvantages:-
→ Require deep understanding.
→ wrong model can delay and confusion.

* What is Incremental Delivery? How is it useful for projects?

A)> Incremental Delivery is a software approach where system built and delivered in small, functional segments called increments.
> It allows part of the software to be released.
> It used earlier in project lifecycle to adjust future development.
> Once working base is developed and delivered, additional modules are added in increments.
> Each increment builds upon the previous one.
> Every increment goes through full Software development cycle.
> It makes model highly suitable for projects that are flexible to success.
> Ex:- Online Banking System, increments are user login, viewing account balance.

→

* Incremental Delivery is especially useful for software projects
* It allows teams to deliver working parts of system early and improve them over time.
* This approach is ideal when client feedback is important.
* Development becomes more organised by breaking project into smaller chunks.

Benefits:-
> Early delivery of functional Software.
> Improved feedback loop
> Better Risk Management
> Simplified Testing & Debugging.

Challenges:-
> Requires careful initial Planning
> Integration complexity
> version control issues.
> Not suitable for all Systems.

Ex:- Sample project - Online Banking System.
An online application initially releases core features:
login
account balance
view
fund transfer.
later adds investment options: customer support chat
Bill payments.

Advantages:-
> partial functionality
> feedback improve increments
> Easy risk management
> Simplifies testing

Disadvantages:-
> require good planning
> modules must integrate well
> version issues.
> Not suitable for all systems

* Explain Rapid Application Development (RAD) & its phases? (8)

How RAD is used for project Implementation?

(A) RAD means Rapid Application Development.

> It is a type of agile software development methodology.
> It emphasizes quick and iterative development cycles..
> The goal of RAD is to speed up entire software development process using prototype and early testing..
> It is ideal for projects where requirements are well understood
> It allows for multiple iterations of development.

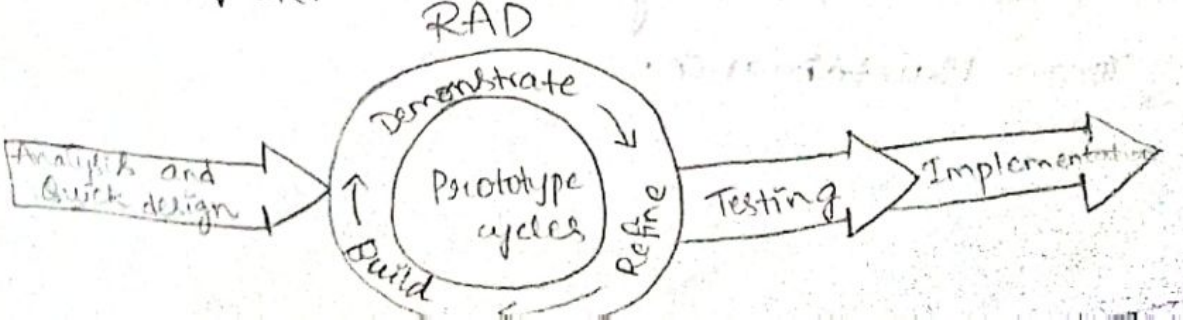
phases of RAD:- (4).

→ Requirements Planning:-
> includes users, stakeholders, developers, critical requirements.
> Detailed Specifications are not finalized here.

→ User Design:-
> It is interactive & iterative.
> Through workshops, JAD sessions.
> users & developers work together to create models.

→ construction:-
> It is built using reusable components and GUI tools.
> Frequent changes and refinements are allowed during this phase.

→ cutover (Deployment):-
> The final system is tested, data is migrated.
> Most of testing is done during construction.
> It is quick.



* RAD is useful for project implementation.
* It shortens the development cycle by focussing on prototype than planning & documentation.
* It ensures that a version of software is delivered early
* It ensures it is improved iteratively based on feedback.

Ex:- project : To develop a Student Attendance Management System.

RAD helps team to quickly develop a prototype with features as student login, report generation.

Benefits:-

- Speeds up development & deployment.
- Encourages stakeholder engagement.
- simplifies debugging & testing
- Allows parallel development
- Improves efficiency.

Advantages:-

> Faster Delivery
> Higher user satisfaction.
> Early detection of issues.
> Flexibility
> Reduce duplications.

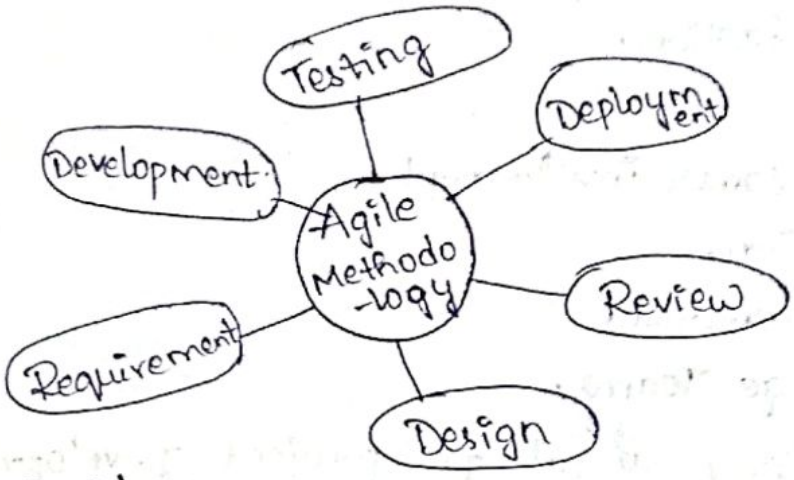
Disadvantages:-

> Requires continuous user involvement
> Not ideal for large, complex systems.
> Less focus on scalability
> Long Term Maintainance.

* Explain about Agile Methodology?

Agile software development refers to set of practices for software development.

> The main goal of Agile is to deliver small, functional parts of software, enable teams to adapt to change requirements.



* Core principles:-

> Customer collaboration over contact Negotiation.
> Responding to change over following a plan.
> Working Software over Comprehensive Documentation.
> Individuals & Interactions over Processes & Tools.

* Agile Framework & Methodologies:-

> Scrum → It focusses on delivering incremental improvements.
→ It involves Sprints.
→ It gather feedback and improve the process.

> Kanban → visual method.
→ suitable for teams
→ Require flexibility to manage workflows.

> Extreme Programming - emphasizes simplicity in design, constant feedback.

> Lean Software development → focusses on eliminating waste.
→ It derives principles from lean manufacturing & emphasizes delivering value to customers.

→

* Advantages:-

> High responsiveness to change.
> Improved collaboration with stakeholders.
> continuous Testing and Improvement.
> Early & Regular delivery
> Customer Satisfaction.

* Disadvantages:-

> Requires high customer involvement.
> Daily coordination.
> less for Documentation.
> Difficult for large Teams.

* How Agile Methodology useful for project Development?

(A) Agile Methodology used for project Development to manage product software.

> Encouraging iterative Development:- Allows team to develop parts of software quickly & regularly, progress made continuously. It includes:
• Early deliveries • Frequent updates • Continuous Feedback.

> Prioritizing features:- It is a key component of Agile development, with features based on business value.
It includes:
• focus on Core requirements first.
• User-centered Approach.

> Facilitating strong communication & collaboration:- Agile methods emphasize communication & collaboration between team & stake holders.
It includes:
• Daily Standup Meetings • Customer collaboration

> Fostering Flexibility & Adaptability:- It means that teams are flexible and adopt to changing requirements evolve user needs
It includes:
• Responsive to change • Continuous Improvement.

> Improving Product Quality:- It promotes continuous testing & quality checks throughout the development process. It includes.
- Test-drive Development
- Frequent Releases.

> Faster time to Market:- Agile breaks down development into smaller, manageable chunks. It includes:-
- Early Product Releases
- Early value Delivery.

> Risk Management:- It helps to mitigate risks throughout the life cycle by identifying potential issues.
- Early problem Identification
- Risk Mitigation.

> Enhancing Team Morale & Productivity:- Agile promotes a collaborative work environment to make decisions. It includes;
- Autonomy - sense of responsibility and trust.
- Transparency - regular progress reviews, sprint goals.

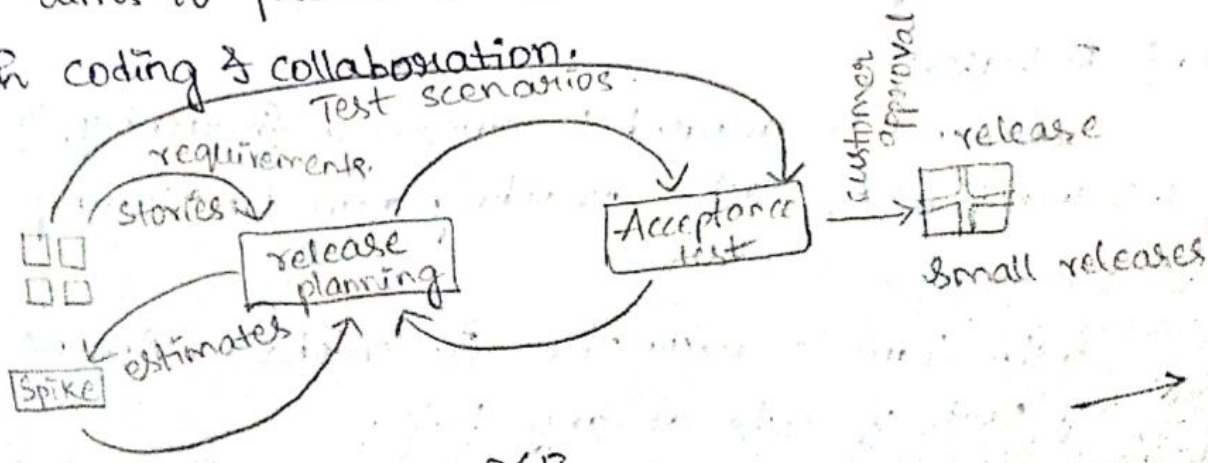
* Explain about Extreme Programming & its practices.

(A) Extreme Programming (XP) is a Agile software Development methodology to improve software quality & responsiveness to customer requirements.

> It emphasizes continuous improvement, collaboration.

> XP was created by Kent Beck in late 1990s to address challenges in software development.

> It aims to provide a framework for small, flexible to both coding & collaboration.



XP.

* Core practices of XP:-

XP incorporates a set of key practices that are designed to ensure that software is developed, high-quality and customer focussed manner.

> Pair Programming:- • Two developers work together.
One is driver who writes code.
other is observer who reviews code.
• It encourages real time feedback & collaborati

> Test-Driven Development:-
Developers write tests for a feature before writing the actual code and implement the feature.
• It ensures code is always tested reducing defects and ensuring that software meets requirements.

> Continuous Integration (CI):-
Developers integrate their code into main codebase frequently.
• It helps identify bugs early, reduces integration issues and ensures system remains in working state.

> Refactoring:-
process of continuously improving structure of existing code without changing its external behaviour.
• It ensures code remains clean, modular and adaptable to future changes.

> Frequent Releases:-
software is released to customers frequently, typically every 1-3 weeks with each providing new functionality.

> Collective Ownership:-
Entire team is responsible for codebase. Anyone can modify any part of code at any time.
• It promotes collaboration and ensures that the whole team is maintaining high code quality.

> Simplicity:- Developers focus on writing simple, straightforward code that meets requirements for future needs.
- It ensures that it is easier to understand, maintain & extend.

> Customer Involvment:-

It is actively involved throughout the development process, defining requirements and reviewing progress.
- It reduces risk of delivering a product that doesnot involve customer expectations.

> Sustainable ~~Place~~ Pace:-

Teams work at a pace that they can maintain indefinitely without burning out.
- It is to avoid excessive overtime or stress.

* Advantages:-
> Improved code Quality
> Faster Delivery
> Better collaboration
> Flexibility.
> customer satisfaction.

* Disadvantages:-
> High resource usage
> customer Availability
> Risk of over Engineering
> Not suitable for fixed ~~contract~~ Projects.
> Difficult to Scale.

Ex:- Sample project: Real-Time chat Application.

A chat System with messaging, notifications and media sharing is developed using XP.

→

* What is SCRUM and its outcomes of SCRUM methodology.

(A) SCRUM is an Agile framework designed to help teams manage complex projects, in software development.

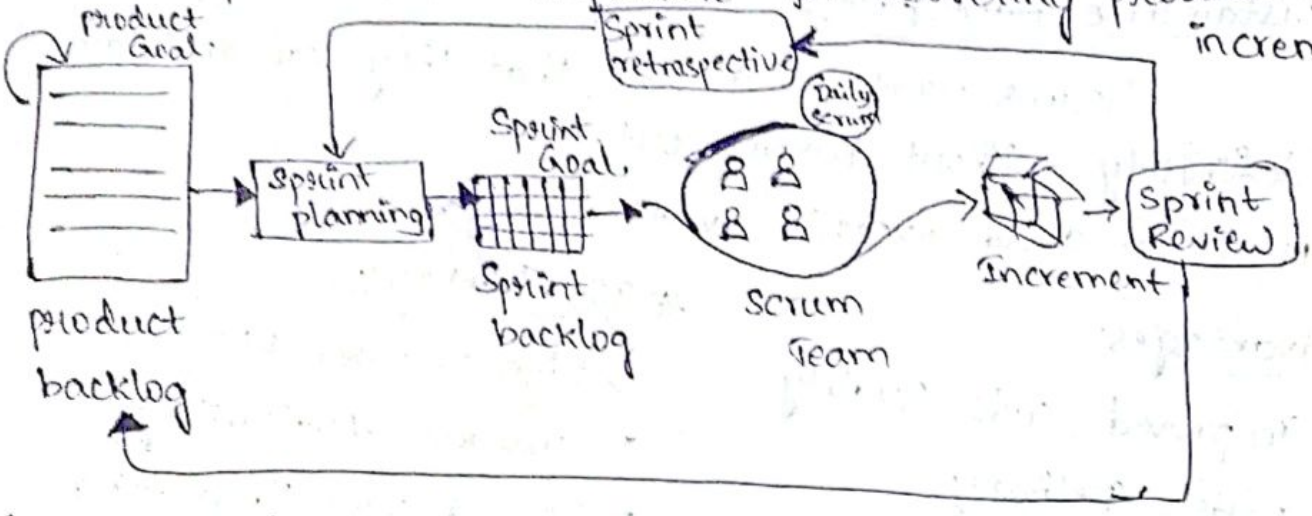
- SCRUM organises work into time-boxed iterations called sprints;
- SCRUM is based on defining roles, events ~~and~~ that ensure clear communication, regular feedback.

key roles:

product owner - defines vision & priorities for project

SCRUM Master - facilitates the process.

Development Team - responsible for delivering product increment.



* Components of SCRUM:-

> Sprints :- This is time boxed approach.
- Ensures regular progress
- Allows teams to adjust based on feedback.

> Product Backlog :- • Product Owner is responsible for managing & prioritizing the backlog based on customer needs.

> Sprint planning :- The team breaks these items into tasks and estimates the effort required.

> Daily-Stand-ups :- A brief daily meeting where the team discusses progress, plans for the day and roadblocks encountered.

> Sprint Review :- At the end of each Sprint, review is held where the team demonstrates completed work to

- Sprint Retrospective: It helps the team continuously refine their processes and became more efficient over time.
- Optimizing project outcomes with SCRUM Methodology

SCRUM is most beneficial in projects where requirements evolve over time and where there is a need for frequent stakeholder involvement and feedback.

> Encourages collaboration & Transparency.
> Adaptability to changing requirements.
> Faster Time-to market.
> Continuous improvement
> Better risk Management
> Customer satisfaction.

Advantages:-
> Encourages Teamwork
> Adaption to change
> Accountability
> Transparency through ceremonies.

Disadvantages:-
> Role confusion in new Teams.
> Requires experienced SCRUM Master.
> Sprint Scope may be unstable.
> less documentation.

* Explain about Managing interactive processes.

(A) Managing interactive processes in software development involves the design, implementation and optimization of systems that interact directly with users.

Ex:- touch screen kiosks, mobile applications, websites.

> These processes is crucial for ensuring a high quality user experience & user interface. (UX).

* Key aspects of Managing interactive processes:-

> User-centered design:

- The software effectiveness is directly tied to how well it meets these needs.
- Designers often create personals, that represent different user types to guide design decisions.

> Prototyping & iteration:-

- Interactive systems often undergo multiple iterations where prototype of systems are tested.
- It involves quickly developing a working model of the system, even if it is incomplete.

> Continuous user feedback:-

- For interactive systems it is vital to have ongoing feedback from actual users.
- Feedback provides insights into how units are interacting with the system.

> Real-Time Interaction Management:-

- System designed for interactive use must be able to handle real-time interactions.
- It provides useful feedback, adapts to changing needs during interaction.

> Adapting to changing user Behaviour:-

- It also involves staying responsive to changes over time.
- A/B testing technique is often used to compare two versions of a system to determine which one performs better.

→ Challenges:- > Time consuming Design phases
> High resource demand.
> Maintain balance b/w User needs & constraints.
> Consistency in UI/UX Across platforms.

Advantages:-
> prioritizes user needs.
> Improves UI/UX Quality
> promotes usability Testing
> Adaptive to change:

Disadvantages:-
> Requires iterative prototyping
> Time-consurring design phase.
> High cost for studies.
> Need continuous feedback cycles.

Ex:- Airport touchscreen kiosk system
It allows check-in, print boarding passes, access flight info

* What are basics of Software Estimation.

(A) Software Estimation is process of predicting effort, time, cost and resources required to successfully complete a software project.
> It forms the foundation of project planning, budget, risks efficiently.
> It helps project managers allocate right resources.
> Several techniques are used in software estimation like top-down approaches, three point estimation and models like COCOMO.
> A well-executed estimation process reduces risks, improves decision-making & increases likelihood.
It is crucial for: planning & scheduling
Budgeting
Resource allocation
client communication
Risk Management.

* Parameters:-
> Time - helps in scheduling work, setting deadlines, track progress
> cost - helps in budgeting, cost control, stakeholder communication.
> Scope - includes features, functionality & deliverables.
> Risk - Allows planning mitigation strategies early.
> Resources - Estimation helps ensure right people & tools needed.
> Quality - affects time, cost & resources on quality expectations

* Common Software Estimation Techniques:-

> Top-down Estimation:-
- Start from overall project size, break into small tasks.

> Bottom-up Estimation:-
- Estimate each task individually & add them up.
- offer high accuracy.

> Expert Judgement:-
- Based on past experience & domain knowledge.
- Fast and practical.

> Analogous Estimation:-
- Use data from similar past projects.
- Quick but depends on how close project matches

> Parametric Estimation:-
- Use formulas and models (Lines of code).

> Three-point Estimation:-
- PERT or triangular use optimistic, pessimistic.
- PERT => (O+4M+P)/6
- Triangular => (O+M+P)/3.

Steps:-

> Define the project.
- Identify project type (web, mobile)
- choose Methodology.

> Understand Scope.
- Document features, requirements.
- Get approval for all Stakeholders.

> Break down the work.
- Split project into smaller tasks or modules.

> prioritize tasks.
- Focus on high priority features first.

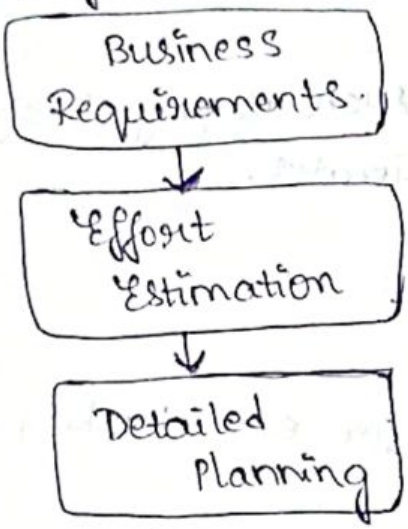
> choose Estimation Technique.
Use one or combine several techniques based on:

• project size • Available data • Experience level.
Some tools are JIRA, Microsoft, COCOMO, Spreadsheets.

* Explain about Effort & cost Estimation Techniques.

Effort and cost estimation techniques are used to predict the amount of effort, time & financial resources to develop a software product.

> An accurate estimation ensures that the project can be delivered on time, within budget with expected quality.
> Different techniques are applied based on project's size, complexity, data and phases.



① → Expert Judgement
② → Bottom-up Estimation.
③ → Three-point Estimation
④ → planning poker
⑤ → Agile story points

→ Expert Judgement:-
• This relies on the knowledge and experience of software professionals who have worked on similar projects.
• They may do this individually or in groups through delphi methods.
Strengths:- Quick, practical
useful data not available.

→ Analogous Estimation:-
• It compares the current project or modules to similar past projects.
• Adjustments are made based on differences in size, complexity, technology.
Strengths:- Realistic & based on proven data.

→ Top-down Estimation:-

In this approach, estimates are made at a high-level often by senior management, without detail into individual

Strengths:- Fast & useful in early project stages.

→ Bottom-up Estimation:-

This technique involves estimating each component or task in detail & aggregating them to form total estimate

Strengths:- very accurate if detailed requirements are available

→ Parametric Estimation:-

This method uses mathematical models and statistical data to estimate effort & cost.

- Models like COCOMO use variables such as project size, complexity, developer productivity to calculate estimates.
- Strengths:- Quantitative & objective.
  useful for large projects.

→ Three-point-Estimation:-

This considers 3 scenarios for each task: optimistic, pessimistic, PERT.

Strengths:- Accounts for uncertainty & risk.

Advantages:-

> useful in early stage planning.
> combines expert knowledge & data.
> Helps in budget allocation.

Disadvantages:-

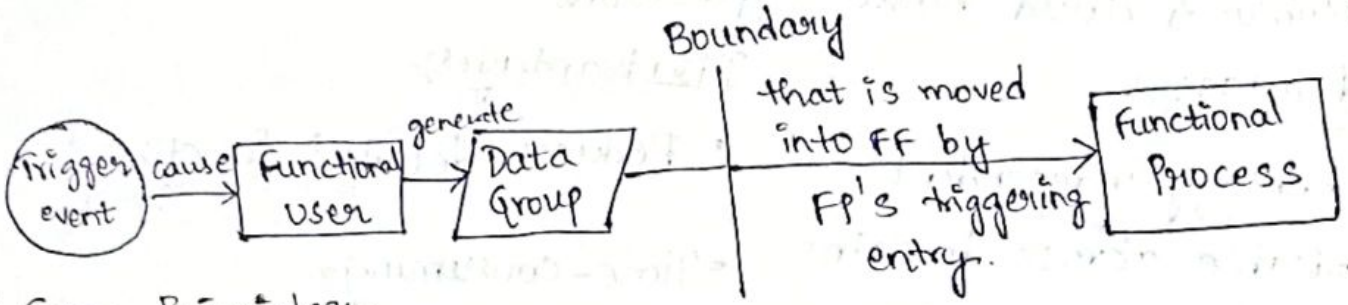
> Time-consuming
> prone to cost overruns.
> Dependence on expert availability.
> Difficult to estimate.

* Explain COSMIC full function Points (FFP).

COSMIC Full Function Points (FFP) is a standardized method used to measure the functional size of a software system.

> COSMIC FFP developed by Common Software Measurement International Consortium (COSMIC).

> It quantifies amount of functionality a system provides to users, independent of how it is implemented.



* Core Principles:-

COSMIC measures functional user requirements by identifying and counting data ~~se~~ movements.

- A data movement is a interaction involving data b/w users & software or within system itself.
- There are 4 types of data movements and each one is counted as one COSMIC function point.
  - Entry (E):- Data that enters the system from a user or external source.
  - Exit (X):- Data that exits system to a user or external destination.
  - Read (R):- Data that is retrieved from persistent storage.
  - Write (W):- Data that is written or updated in persistent storage.

Example:- Hotel Reservation System.

let's apply COSMIC.

- Guests input reservation details (entry)
- System reads room availability from database (Read)
- System sends confirmation message (Exit)

- System stores booking information (write)
- One functional process would be sized as

1 Entry + 1 Read + 1 Write + 1 Exit = 4 FP's.

Applications:-

- Business applications
- Real-time systems
- Embedded systems
- Mobile & cloud based applications.

Advantages:-

- Technology-independent
- Applicable across domains
- Supports benchmarking
- Reliable for outsourcing.

Disadvantages:-

- Reduces detailed functional spec.
- Time-consuming
- Limited automated tools.

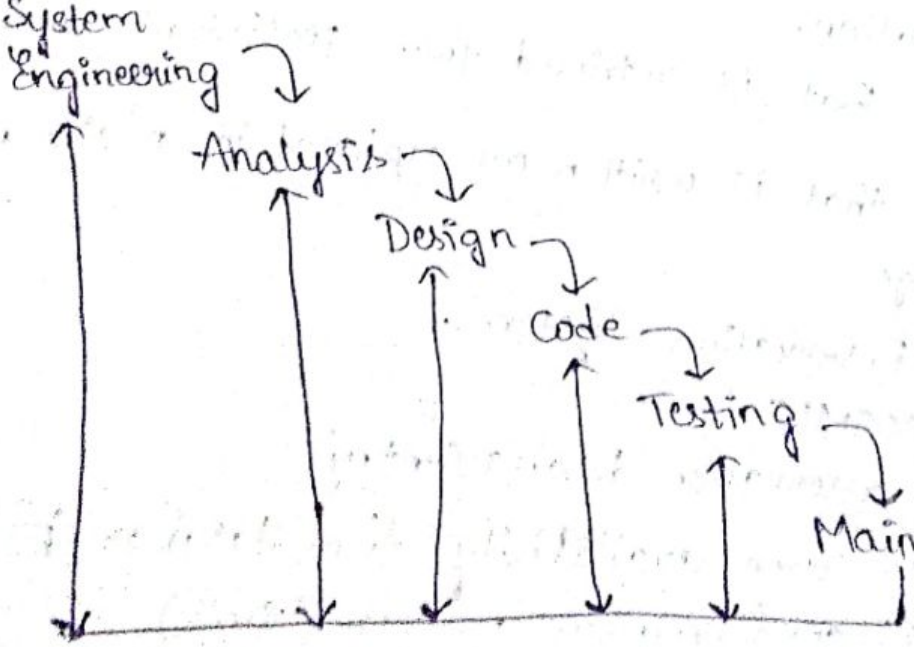
* What is COCOMO II? $

COCOMO II (Constructive Cost Model II) is an evolution of the original COCOMO.

> It was developed by Barry Boehm in 1981.

> It is one of the most widely used parametric models for software cost estimation.

> COCOMO II was developed to address shortcomings of original COCOMO model.



COCOMO II Model Components:-

COCOMO II model breaks down the software development process into different phases and uses multipliers.

>It has 3 distinct submodels of Software development.

- **Applications Composition Model:-**

Submodel is used for estimating effort for projects that involve composing software from reusable components.

- used for projects that build systems primarily from reusable software modules.

- **Early design Model:-**

It is used in early stages of project where requirements are not fully defined.

- It provides high-level estimates for projects based on limited information.

- **Post-Architecture Model:-**

It is used for detailed planning & estimation once system architecture is fully defined.

- It is used for projects that have passed through requirements gathering phase and are in design, testing phases.

COCOMO II Estimation formula.

$$E = a \times (size)^b \times \prod_{i=1}^{n} EM_i$$

where; E = effort required (in person-months)

a = A constant

Size = Size of software.

b = scaling exponent that adjust complexity.

EM_i = effort multipliers for each of the factors.

→

Parameters in COCOMO II:-

> Size Estimation — size can be measured in lines of code (LOC), function points or other relevant units.

> Effort Multipliers — to adjust estimated effort based on various attributes.

- These are 2 types.

→ Cost drivers – include attributes like team capability, process maturity, system reliability, product complexity.

→ Scale drivers – help adjust for scale of project. Scale factors account for non-linear relationship b/w size & effort.

Some of the Effort Multipliers are:-

> product complexity
> Software reliability
> Developer capability
> System Architecture
> User of Modern programming tools.

Advantages:-

> provides detailed cost
> widely accepted
> helps in resource planning
> supports 'what-if' analysis.

Disadvantages:-

> Inflexible
> Requires collaboration
> Not ideal for small projects
> Intensive data collection

Ex: Hospital Information Management System.

It includes system complexity, team capability & reliability.

* Explain about staffing pattern.

* Staffing Pattern:-

A staffing pattern is a structured plan or framework that defines the roles, responsibilities and allocation of resources through Software Development life cycle.

> It ensures that the right people with the right skills are assigned to the right tasks.

> It ensures that project goals are met within required time & budget.

Components of Staffing Pattern:-

- Roles & Responsibilities:-

A staffing pattern outlines the various roles included in the project as project Managers, developers, testers, architects, designers & support staff.

> Each role is associated with specific responsibilities & tasks within the SDLC.

- Resource Allocation:-

It involves determining the number of people and specific skill sets required at different stages of project.

- Timing and Duration:-

It ensures that the resources are available during the critical phases of project when heavy coding is required, testing is crucial before deployment.

* Importance of Staffing Pattern:-

> Optimizing Resources:-

By assigning the right personnel to each task, staffing patterns optimize the use of resources.

> Cost Efficiency:-

- Proper staffing patterns ensure that the project is not ~~oversatisfied & undersatisfied~~ overstaffed & understaffed.

- The pattern allows for better control over the budget by managing personnel effectively based on need of project.

> Improved Coordination:-

• When roles are clearly defined, each team member knows their responsibilities, leading to conflicts and misunderst problem solving.

• It ensures smoother collaboration & more efficient

> Adaptability to Project changes:

• Staffing patterns need to be adaptable as projects often evolve during development lifecycle.

*Types of Staffing Patterns:-

> Traditional Staffing Pattern:-

• This ~~mode~~ model assigns roles in a structured & sequential manner.

• It can be less efficient & complex.

> Agile Staffing Pattern:-

• These are more flexible than team members might play multiple roles across project.

• It is flexible & adaptable.

> Matrix staffing pattern:-

• Here, resources are assigned across different dimensions projects but complex.

• It allows for better resource allocation across multiple

> Hybrid Staffing Pattern:-

• It combines both traditional and agile staffing pattern

• It follow overlapping roles.

Advantages:-

> Balances workload
> Increase productivity
> Enhance team coordination

Disadvantages:

> poor planning
> overstaffing increase cost
> Fixed roles.
> Difficulties in distributed teams.