Software Engineering Concepts

Software Characteristics & Development Lifecycle Models

FUNDAMENTALS

1. Software ki Characteristics

Note: Software hardware se multiple aspects me completely different hota hai.

Characteristic	Meaning
Developed, not manufactured	Software banaya/develop kiya jata hai, factory me physical goods ki tarah manufacture nahi.
Does not wear out	Hardware ki tarah physical wear-and-tear (ghisna) nahi hota.
Intangible	Software ko physically touch nahi kar sakte, ye logical program hota hai.
Easy to modify	Changing requirements ke according update aur modify kiya ja sakta hai.
Maintenance is important	Future changes, bug fixes aur performance improvements ke liye maintenance zaroori hai.
Mostly custom-built	Bahut se software specific client requirements ke dynamic specs par depend karte hain.

SDLC MODEL

Waterfall Model 🌊

> **English:** Waterfall is a linear and sequential software development model. Each phase must be completed before the next phase begins.
>
> **Hinglish:** Waterfall ek step-by-step model hai. Ek phase complete hone ke baad hi next phase start hota hai.

Requirements → Design → Coding → Testing → Deployment → Maintenance

ITERATIVE APPROACH

Prototype Model 🧩

English: The Prototype model develops an early working/sample version of the software to understand and clarify customer requirements.

Hinglish: Jab customer ko khud bhi clear nahi hota ki exact software kaisa chahiye, tab ek sample/demo version (prototype) banaya jata hai.

Initial Requirements → Prototype → Customer Feedback → Refinement → Final Product

💡 **Formula: Prototype = Unclear Requirements + Customer Feedback**

RISK-DRIVEN

Spiral Model 🌀

English: Spiral is an iterative and risk-driven software development model. Its major focus is risk identification and management.

Hinglish: Spiral model me software ko multiple cycles/iterations me develop kiya jata hai aur har cycle me risks ko identify aur manage kiya jata hai.

Planning → Risk Analysis → Engineering → Evaluation → Next Cycle

🌀 **Summary Rule: Spiral Model = "Thoda banao → Risk check karo → Feedback lo → Phir aur banao"**

Project Estimation Notes

Software Engineering Concepts & STET Key Takeaways

OVERVIEW

Software Project Estimation

> **English:** Software Project Estimation means estimating the size, effort, cost and development time required to develop software.
> **Hinglish:** Software banane se pehle estimate karna ki: Kitna bada software hai? Kitna effort lagega? Kitna cost hoga? Kitna time lagega?

Estimation Sequence for STET:

LOC → FP → COCOMO

METRIC 1

1️⃣ LOC — Lines of Code

> **English:** LOC = Lines of Code. It measures software size by counting the number of source code lines.
> **Hinglish:** Software kitna bada hai, iska estimate code ki lines count karke karte hain.

Example & Conversion:

- Software size: **1000 lines of code → LOC = 1000**
- **1 KLOC = 1000 LOC** (Therefore, 1000 LOC = 1 KLOC)

🧠 **STET Point: LOC measures → Software Size**

🚨 **Limitation:** Different programming languages me same functionality ke liye different number of lines lag sakti hain.

METRIC 2

2️⃣ FP — Function Point

> **English:** Function Point (FP) measures software size based on the functionality provided to the user, rather than lines of code.
> **Hinglish:** FP ye dekhta hai ki software user ko kitni functionality provide kar raha hai, na ki code kitni lines ka hai.

Example (Banking Application Functions):

- Login
- Account creation
- Balance inquiry
- Money transfer
- Report generation

🧠 **STET Point: LOC → Code size | FP → Functionality / user features**

⭐ **Advantage: FP programming language independent hota hai.**

MODEL

3️⃣ COCOMO (Constructive Cost Model)

Full Form: Constructive Cost Model | **Developed by:** Barry Boehm

English: COCOMO is a model used to estimate software development effort, cost and time based mainly on software size (KLOC).
Hinglish: COCOMO ka use estimate karne ke liye hota hai: Kitna effort lagega? Kitna time lagega? Kitni cost aa sakti hai?

🔥 COCOMO ke Types (STET Focus):

Type	Meaning	Complexity Level
Organic	Small/simple projects	Lowest complexity
Semi-detached	Medium complexity projects	Medium complexity
Embedded	Highly complex projects	Highest complexity

🧠 Trick to Remember: O → S → E (Organic → Semi-detached → Embedded)
Complexity Order: Organic < Semi-detached < Embedded

⭐ COCOMO Basic Formula & Coefficients:

Effort Formula: $Effort = a \times (KLOC)^b$ (Unit: Person-Month / PM)

Development Time Formula: $T = c \times (Effort)^d$

🚨 Basic COCOMO Coefficients Table:

Mode	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi-detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Example Calculation:
Suppose: **KLOC = 10** for an **Organic project**
$Effort = 2.4 \times (10)^{1.05}$ Person-Months

Software Engineering Revision Notes

SRS • STRUCTURED DESIGN • OBJECT-ORIENTED DESIGN

1. SRS — Software Requirements Specification Fundamentals

ENGLISH

> SRS (Software Requirements Specification) is a document that describes what the software should do and the requirements of the system.

HINGLISH

> SRS ek document hai jisme likha hota hai ki software me kya-kya requirements/features hone chahiye.

🧠 **STET Trick**

SRS = What to build, not How to build

2. Structured Design Function-Oriented

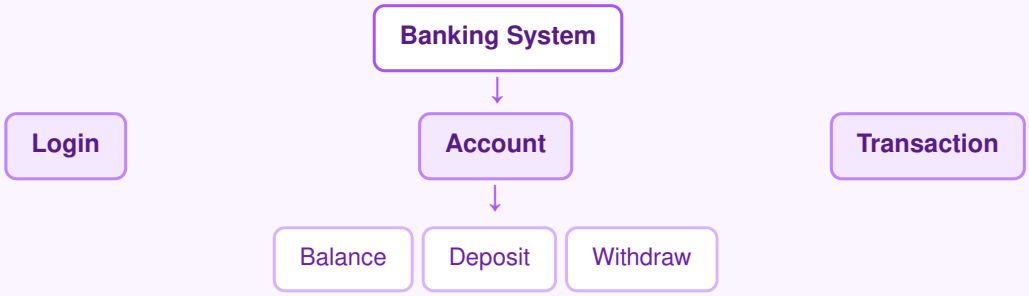
ENGLISH

Structured design is a function-oriented design approach where a system is divided into smaller functions/ modules.

HINGLISH

Bade software ko chhote-chhote modules/functions me divide karna = Structured Design.

📌 **EXAMPLE: BANKING SYSTEM FUNCTIONAL BREAKDOWN**



Yahan system ko functions ke according divide kiya gaya hai.

KEY FOCUS AREA

Structured design commonly focuses on: Functions + Modules + Data Flow

🧠 **STET Trick**

Structured Design = Function-oriented

3. Object-Oriented Design (OOD) Class & Object-Oriented

ENGLISH / CONCEPT

Software is divided and designed according to objects and classes, encapsulating data and behavior together.

HINGLISH

Software ko objects/classes ke according divide karte hain.

🧠 **Key Takeaway**

Object-Oriented Design = Class & Object-oriented (Data + Behavior)

Software Testing & Maintenance

Quick Reference Study Notes

1. Software Testing

ENGLISH

> Testing is the process of checking software to find errors/defects and verify that it works as expected.

HINGLISH

> Testing ka matlab hai software ko check karna ki wo sahi kaam kar raha hai ya nahi aur errors/bugs find karna.

2. Unit Testing

ENGLISH

> Unit Testing tests the smallest individual unit/component of a software separately.

HINGLISH

> Software ke ek chhote part/module/function ko alag se test karna = Unit Testing.

Example

Suppose Banking software me:

`deposit()` `withdraw()` `checkBalance()`

Agar sirf `withdraw()` function ko test kar rahe hain: 👉 **Unit Testing**

🧠 **Trick: Unit = One small part**

3. System Testing

ENGLISH

> System Testing tests the complete integrated software system as a whole.

HINGLISH

> Software ke poore system ko ek saath test karna = System Testing.

Example: Online Shopping System

Login ↓ Search Product ↓ Add to Cart ↓ Payment ↓ Order

In sab ko together test karna: 👉 **System Testing**

🧠 **Trick: System = Whole software**

4. Black-Box & White-Box Testing

🖤 Black-Box Testing

> In Black-box testing, the tester tests the software without knowing its internal code or implementation.

> Tester ko andar ka code/logic nahi dekhna. Sirf input dena hai aur output check karna hai.

🤍 White-Box Testing

> In White-box testing, the tester has knowledge of the internal code, logic and structure of the program.

5. Software Quality

ENGLISH

> Software Quality means the degree to which software meets specified requirements and satisfies user needs.

HINGLISH

> Software quality ka matlab hai software requirements ko kitna achhe se fulfill karta hai aur user ki needs ko satisfy karta hai.

Important Software Quality Factors:

Correct + Reliable + Efficient + Easy + Maintainable + Portable + Secure

6. Software Maintenance

ENGLISH

> Software Maintenance means modifying and updating software after its delivery/deployment.

HINGLISH

> Software deliver hone ke baad usme bug fix, changes, improvements ya new environment ke according updates karna = Software Maintenance.

4 Types — Most Important Table

Type	English	Hinglish	Trick
🔴 **Corrective**	Fix existing errors	Bug fix	`Fix`
🔵 **Adaptive**	Adapt to changed environment	New OS/environment ke according change	`Adapt`
🟢 **Perfective**	Improve/add functionality	Software ko better banana	`Improve`
🟡 **Preventive**	Prevent future problems	Future problems ko rokna	`Prevent`