software engineering

SOFTWARE ENGINEERING

* It is a program / set of programs containing instructions which provide desired functionality.
* Also comprises of data structures that enable the program to manipulate information

The process of designing and building something that serves a particular purpose

SE: A systematic approach to the development, operation and maintenance of desired SW.

Dual Role Of Software

1. As a product: It delivers the computing potential of a Hardware (H/W)
   - ↳ enables the H/W to deliver the expected functionality
   - ↳ Acts as information transformer
     - ↳ Business information → query retrieval SW based on Market data.
     - ↳ Personal info → eg: Salary cheque generation

2. As a vehicle for delivering a product
   - ↳ helps in creation and control of OTHER programs.
   - ↳ eg: Operating System.

Scanned with CamScanner

introduction

WHY SOFTWARE ENGG. IMPORTANT ??

* enables us to build complex systems (SW) in a timely manner.
* ensures high quality of S/W
* imposes discipline to work that can become quite chaotic.

WHAT IS WORK PRODUCT (Result/Outcome)

⊙ Software Engineer
↳ the set of programs, the content (data) along with documentation that is a part of SW.

⊙ user / Customer
↳ the functionality delivered by the SW that improves user experience.

SOFTWARE ENGG. FOCUSES ON ??

⊙ QUALITY → Functional: degree to which corre SW is produc
→ Non-Functional
↳ features other than fu of a SW → like robustness.
(Structural Attributes)
↓
while SW development

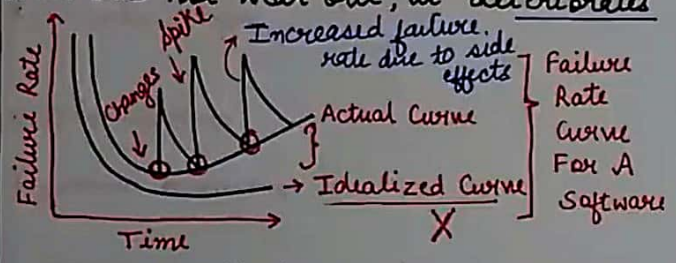
⊙ MAINTAINABILITY → should be easily enhanced and adapt to changing reqmts whenever required.
↓
after the SW has been developed and delivered.

Scanned with CamScanner

software vs hardware

Software

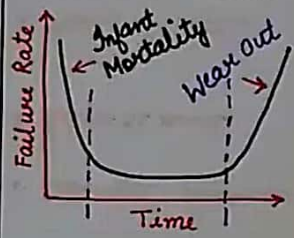
* SW does not wear out, it deteriorates



- Not an idealized curve
  - ↳ Undetected defects
- High failure rate in early life of s/w.
- Defects corrected & curve flattens.
- During its life, SW undergoes change
  - ↳ new errors may be introduced
  - ↳ cause spikes (rise) in failure rate
- Curve spikes with every change that caused new error
- Minimum failure rate rises

Hardware

Hardware wears out with time



⇒ Failure rate as a function of time for H/W.

Bath Tub Curve

- High failure rate early in life cycle
  - ↳ Design / Manufacturing Defects.
- Defects corrected & failure rate drops to a steady level for some time
- With time failure rate rises again
  - ↳ Environmental Factors: Dust, Temp., Pollution etc

Wearing Out of Hardware.

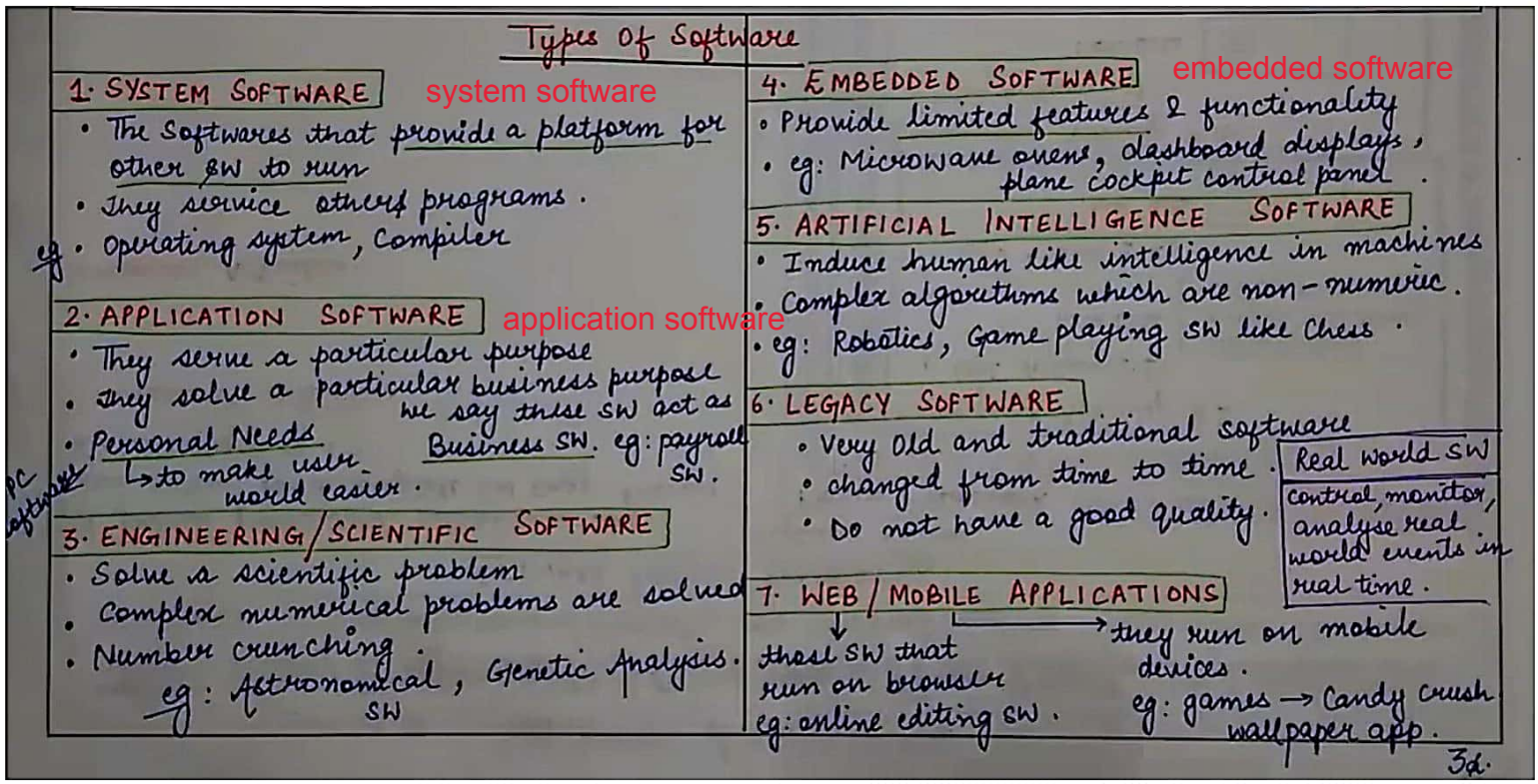
Scanned with CamScanner

table1

* Software is a logical unit. ↳ intangible in nature ↳ Developed by programmers.	* Hardware is a physical unit ↳ tangible in nature ↳ Manufactured in factories
* There are No Software Spare Parts. → SW modules may be dependant on each other	* Spare Parts for Hardware exists. ↳ independant of each other
* Problem Statement ↳ It may not be complete & clear initially.	* Problem statement ↳ Clearly specified at the beginning of the prodn process
* Requirements ↳ May change with time as SW is developed	* Requirements ↳ Are Fixed before HW Manufacturing.
* Multiple copies ↳ Less costly than HW.	* Multiple Copies ↳ are created using assembly lines etc ↳ Costlier than copying a S/W.

Scanned with CamScanner

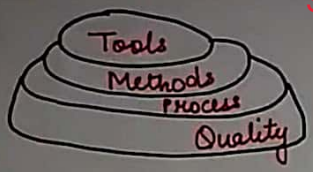
types of software



Scanned with CamScanner

FTWARE ENGINEERING : A layered technology

SE



Software Engineering comprises of a Process, a set of Methods for managing and developing the S/W, and a collection of Tools

- The bedrock that supports Software Engg. is Quality focus

QUALITY → Functional → degree to which correct SW has been developed
→ Non-Functional (structural) ↓ eg: Maintainability, Robustness

functional

six sigma, QM: Total Quality mgmt

PROCESS → A framework that must be established for effective delivery of S/W.
→ Timely development of the S/W.
→ Management and control of SW projects

↓ consists of

METHODS
↳ Provide technical "how-to" for building a software
→ each method consists of multiple tasks.
eg: requirement analysis, testing, support.

TOOLS → Provide automated / semi automated support for process and methods.
CAD or Computer Aided Software : a set of tools working together to deliver a functionality in the

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

SOFTWARE LIFE CYCLE MODELS

sdlc

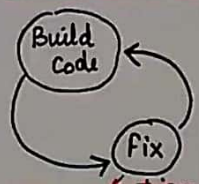
The Process followed in the development of the SW depends upon the LIFE CYCLE MODEL chosen for development.
Life Cycle Model defines the major phases during and after the S/W development process.

Types of Life Cycle Models

models

1. BUILD AND FIX MODEL
2. WATERFALL MODEL
3. INCREMENTAL PROCESS MODELS
   - A) ITERATIVE MODEL
   - B) RAPID APPLICATION DEVELOPMENT → RAD
4. EVOLUTIONARY MODELS
   - A) PROTOTYPING MODEL
   - B) SPIRAL MODEL
5. RATIONAL UNIFIED PROCESS MODEL

1. Build And Fix Model



error correction/ addition of new functionality.

- 2 Phase Model consisting of Writing Code → Fixing It
- Product constructed without any formal requirement specifications OR design.

* Directly developers start building the product which is fixed (changed/reworked) as many times as the customer desires.

build and fix

Disadvantages

1. Not suitable for large size S/W
2. Cost is very high as compared to a structured approach
3. Code becomes unfixable very soon.
4. Maintainence is very difficult
   ↳ Lack of requirement & design documents.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

SOFTWARE LIFE CYCLE MODELS

2. WATERFALL MODEL

(classical or traditional model)

- 5 Phase Model
- Each phase executed Sequentially without Overlap.

Reqmt Analysis And Specification → Design → Implementation & Unit Testing → Integration & System Testing → Operation & Maintenance

Purpose: To understand customer requirements & document them properly. (formal and non-formal)

Work Product:
SRS: Software Requirement Specification Document.

Use: used as a contract between the developer and customer.

▲ Design implemented
▲ SDD used
▲ Coding Phase
▲ Unit Testing: each module is tested independantly of other modules

* Modules combined together to form a complete S/W
* SW as a part of system
* Interfaces b/w modules are tested.

DESIGN PHASE — **Purpose:** Transforms requirements into a structure suitable for implementation in some programing language.
* SW architecture is specified in detail.

Work Product
SDD: Software Design Document

Use: useful to start coding.

waterfall model

Purpose: To preserve the value of S/W over time

⊙ After the delivery of the S/W to customer. (→ deployment)
⊙ error correction, enhanced, remove of obsolete fntionalit etc.

scroll

Scanned with CamScanner

SOFTWARE ENGINEERING - WATERFALL MODEL (LECTURE 2)

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

PROBLEMS WITH WATERFALL MODEL

1. Expects Complete and Accurate Requirements easily in S/W development process. (Not Realistic)
2. Working S/W is not available till very late in the SW development cycle. → delay in error discovery.
3. Risk
   - ↳ No assessment of risk
4. Change → NOT suitable for accomodating changing during developments
5. Sequential Nature
   - ↳ Not realistic in today's world AND this model is not suitable for large projects

WHEN TO USE WATERFALL MODEL

Follows the idea of (Define) Before (Design) AND Design Before (Code)

1. When the requirements are very clearly understood & they will not change during SDLC. (stable)
2. Can be used by an organisation :—
   a) that has an experience in developing a particular kind of SW.
   b) when it wants to build a new software based on an existing S/W.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Software Life Cycle Models

3. INCREMENTAL PROCESS MODELS

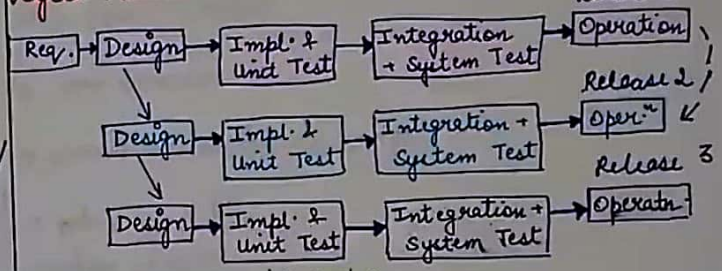
- Requirements are defined precisely.
  - ↳ No confusion about requirements
- Delivery of functionality is done in multiples/phases depending upon the priorities of the requirements.
- Every cycle
  - ↳ delivers a semi-complete product with limited functionality
  - → last cycle : Complete SW

Deliver Limited Functionality Quickly

(A) ITERATIVE ENHANCEMENT MODEL

Phases : Same phases as Waterfall Model :–
- ↳ they are present with less restriction
- ↳ occur in same order but in several cycles.

Requirements : Major reqmts are specified by the customer at beginning and SRS is prepared



↗ Reason for Multiple Cycles

Priorities decided for different reqmts by customer & developer.

Implementation of these requirements is done based on priority.

Waterfall Model	Incremental Model
1. One final product ↳ at the end of SDLC ↳ with all functions + features	1. Complete SW is divided into releases ↳ limited functions
2. Long wait for the SW	2. First release is available within weeks/months.

Scanned with CamScanner

SOFTWARE LIFE CYCLE MODELS

4. EVOLUTIONARY MODELS

PHASES : Same as Waterfall Model But they are applied in cyclical manner.

Difference as compared to Iterative Models

Iterative	Evolutionary
• A usable product is delivered at end of each cycle	• No usable product at end of each cycle.
• Reqmts implemented priority wise	• Reqmts implemented category wise.

Useful When: → when a minimal version of system is NOT reqd.
→ Complex projects w/ unstable reqmts.
→ Requirements are not clear
→ New Technology

4A) PROTOTYPING MODEL → continuous customer support.

→ First develop a working prototype of the S/W instead of the S/W.
- limited functionality + Not very good quality + low performance etc.
- Used to
  - ↳ Refine reqmts
  - ↳ prepare final SRS.

Requirements → (to finalise reqmts) Quick Design → Implement → Customer Eval^{n} → (Not Accepted By Customer) Feedback & Refinement → Quick Design
Customer Eval^{n} → (Accepted By Customer) Design → Implementation & Unit Testing → Integration & System Testing → Operation & Maintenance

prototype

→ Final System developed using Waterfall Model.

Scanned with CamScanner

Is Prototype our final product ?

↳ No it is not our final product.

↳ Prototype code is discarded after reqmts are finalised.

- Benefits of developing a prototype

↳ Helps us to unravel the customer reqmts

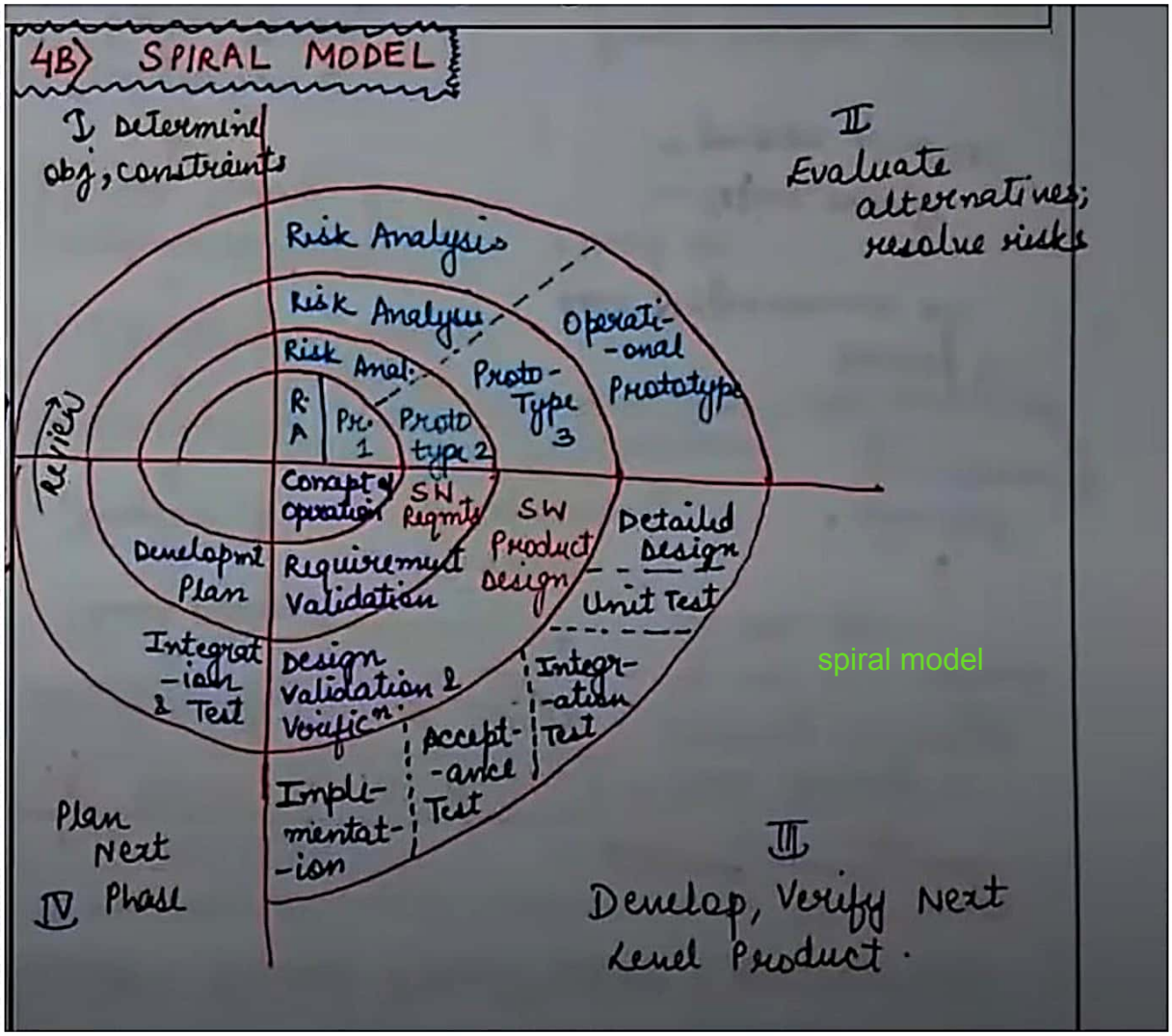
↳ helps us to gather experience for developing the final system.

Does Prototype Development incur Extra Cost

↓

Cost increases but Overall cost comes down.

Scanned with CamScanner



spiral model

Scanned with CamScanner

- Developed By : Barry Boehm
- MAIN Feature : Handling Uncertainty / Risk
- Phases : Multiple phases having 4 activities
- Activities :

1. Planning
   - → Determine objectives + Alternative
   - → Constraints
2. Risk Analysis & resolve
   - → Identify risks, Classify into levels
   - → Alternatives & Plan ahead.
3. Development and Testing
4. Assessment : Customer Evaluation

Phase 1 : Planning → Analyse Risk → Prototype is built → Customer Evaluation & feedback.

Phase 2 Prototype refined → Requirements documented and validated by customer → final prototype

Phase 3 : Risks are now known
+ Traditional approach for development

FOCUS AREAS → Plan for next cycle beforehand

- Identify Problem → Classify into different risk levels → eliminate them before they affect SW.
- Continuous validation (review) by concerned people (designers/ developers) to check the work product quality.

ADVANTAGES

1. Acomodate good features of other SDLC Models
2. SW Quality maintained during development by
   - → Continuous Risk Analysis
   - → Reviews conducted.

DISADVANTAGE : Expertise reqd in Risk Handling + Carrying out Spiral Model.

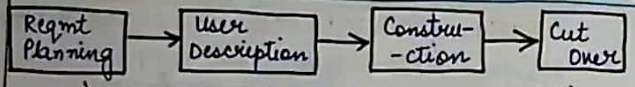
Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

rad → RAD

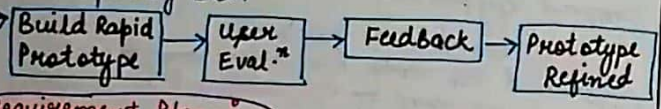
Software Life Cycle Models

3B) RAPID APPLICATION DEVELOPMENT MODEL



With Active User Participation.

→ Developed By IBM



Requirement Planning

→ as long as reqmts are finalised.
- using group elicitation techniques like brains--torming
- user communication reqd for good understanding.

User Description

→ A joint team of customer & developer understands & reviews the gathered reqmts.
→ Automated tools may also be used.

Construction • Design → Coding → Testing.
↳ Product is released. ↳ code generators + screen generators.

Cut Over
↳ Install the SW + Acceptance Testing + Training of user

Major Features

1. **Rapid Prototype**: Quick initial views about product.
2. **Powerful Development Tools** → Development Time ↓ eg: CASE tools.
3. **User Involvement**
↳ Acceptability of product ↑.

NOT USEFUL WHEN
- Users cannot be involved continuously ✓
- Tools & Reusable Components cannot be used.
- High skilled & specialized developers
- System cannot be modularized

Scanned with CamScanner

software requirements engineering re

Software Requirements Engineering (RE)

What is it ? A requirement is a feature of the system or a description of something the system is capable of doing in order to fulfill the system's purpose.
↳ WHAT to do and not HOW to do it.

Work Product : RE produces one large document written in natural language, containing a description of WHAT system will do without describing HOW it will do it.
↳ Software Requirement Specification DOC (SRS)

Input to RE : Problem Statement prepared by customer.
Overview of the existing system + New or additional functionality to be added.

Importance : Without well written requirements :-
- Developers will not know what to do.
- Customers may not be consistent abt reqmts
- It becomes difficult to validate/accept the SW

Understanding the reqmts of customer + Documenting the requirements

reqirement elicitation

4 Steps In Requirement Engineering

1. **Requirement Elicitation** : Gathering of requirements
   * requirements are identified with the help of customer. → existing system.
2. **Requirement Analysis** : Requirements are identified analysed to → find inconsistencies, contradictions, → omissions

requirement analysis

3. **Requirement Documentation** : End product of first 2 steps. Leads to
   - preparation of SRS → srs next
   - becomes foundation for design of SW.
4. **Requirement Review / Reqmt Verification** : To improve the quality of SRS

Scanned with CamScanner

requirement elicitation

Software Requirement Elicitation (Requirement Gathering)

Requirement Elicitation is an activity that helps us to understand what problem has to be solved & what customers expect from the S/W

Foundation:
Effective Communication B/w Customer - Developer

Method:
Developer questions customer → responds → Cross questioning

Hurdles:
- Misunderstandings / conflicts
- Communication Gap. (Omissions of reqmts)

Reason for conflicts between concerned people
1. Developer is efficient in the knowledge of his own development domain while customer is efficient only in his domain (diff. from dev. domain)
2. Lack of proper communication skills.

Requirement Elicitation Methods

1. Interviews

interviews

Purpose: To understand each-other

- Reqmt Engineers act as mediators between customer and development team.
- Correct Approach
  - Open-minded, co-operative, understanding, flexible.

Types of Interviews → Open-Ended
→ Structured

Structured
1. There is a set Agenda.
   - Questions may be prepared.

Open-Ended
1. No pre-set agenda.
   - No pre-defined list of questions prepared

- Questions should be short and simple, clear
- Both parties should be open for discussion to proceed in any direction.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Types of Stakeholders to Interview.

1. ENTRY LEVEL PERSONNEL. stakeholders
   - Do not have much domain knowledge.
   - They have new ideas / perspectives.
2. MID LEVEL STAKEHOLDERS → Project Lead
   - Experienced people with good domain knowledge
   - They know the criticality of the project.
3. MANAGERS AND HIGHER MANAGEMENT
   → Useful Insight about project.
4. USERS OF S/W
   Most important → They will be using the SW max # times.

2. Brainstorming Sessions brainstorming
   ↳ Group Discussion Technique.
   ↳ Promotes Creative Thinking & New Ideas.
   ↳ Platform to express and share views, expectations & difficulties in implementation
   ↳ Facilitator
      ↳ ego clashes / conflicts (to avoid them)
      ↳ encourage people to participate as much as possible.
   • Work Product → Document
   * All ideas are documented to be visible to each participant (using White Boards, Projectors)
   * Detailed report, containing each idea in simple language, is prepared & reviewed by facilitator.
   * At end, a document is prepared with list of requirements & their priority.
3. Facilitated Application Specification Technique
4. Quality Function Deployment.

Scanned with CamScanner

software requirement specification

SOFTWARE ENGINEERING

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Desirable Characteristics of A SRS

1. Consistent
2. Correct
3. Complete
4. Understandable
5. Basis for design & testing
6. Acts as a contract
7. Modifiable
8. Verifiable
9. Traceable
10. Unambiguous

srs details next

Types of Requirements

→ SRS also contains CONSTRAINTS

(I) Functional And Non-Functional Reqmts
(II) User And System Requirements → complexity, language diff.
(III) Interface Specification

stakeholders

Stakeholder refers to anyone who may have some direct or indirect influence on the system requirements

- ↳ end-user who will interact w/ system
- ↳ anyone (except user) who will be affected by the S/W that is developed

FUNCTIONAL REQUIREMENTS (Product Features)

↳ Describe what the system has to do,
→ what are the expectations from the SW. } eg: features of a game.
→ what the SW should not do.

functional

NON-FUNCTIONAL REQUIREMENTS

↳ Mostly Quality requirements
→ Highlight how-well the SW performs its function
eg: For users: High performance, reliability, usability.

For developers:
↳ Maintainability, Testability, Portability.

non functional

→

USER REQUIREMENTS

user

- Written for users who are NOT experts of SW field
- Highlight the overview of system without design description
- Specifies → functional + non-functional.
  - External Behaviour
  - Quality.
  - Constraints.

Scanned with CamScanner

- What To Avoid?
  - ↳ Complex language
  - ↳ Design details.
  - ↳ Technical terms and values.

SYSTEM REQUIREMENTS

↳ Derived from user reqmts OR expanded form of User Requirements

↳ Used as input to designers so that they can prepare SDD.

Both User & System Reqmts are a part of SRS.

INTERFACE SPECIFICATION

↳ Application Programming Interfaces (API) are specified SRS.

↳ what kind of interfaces customer desires.

Feasibility Studies

- Determines if the project is workable or not.
- Work Product: Feasibility Report.
- Use → It helps the management/project team/customer to decide if the SW should be built

What factors are considered?

1. Current Practices vs Proposed System
2. Amount of resources needed.
3. Major risks that can occur
4. Cost and schedule
5. Technical skills reqd.

Benefits

1. 10% failure vs 80% failure
   - better.
2. More accurate estimates can be prepared.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

SRS

CHARACTERISTICS OF A GOOD SRS (DETAILED)

1. CONSISTENCY
- No conflicts between the requirements.
- every requirement must be specified using a standards terminology

2. CORRECT
- What is stated is exactly what is desired
- Expected functionality matches the reqmts present in SRS.

3. UNAMBIGUOS
- Every stated requirement has only (1) UNIQUE meaning
- Words with multiple meanings ?
  - these words should be specified with meaning.
- Software Requirement Specification Language

Adv: Language processors can be used exist which tell diff kinds of errors.

Disadv: Understanding of this language is not for everyone.

4. COMPLETE
- Includes all Functional + Non-Functional Reqmts + Constraints
- Specifies expected o/p from all kinds (valid/invalid) inputs from the user.

5. TRACEABLE
- Origin of each reqmnt should be clear.
- Important ∵ future referencing may be reqd for development/maintainence

6. VERIFIABLE
SRS verifiable (iff) each req. verifiable
- iff there is a process to check if the SW meets each of its reqmnts.
- "Good", "Fast" } 5 sec → 2 sec

Ambiguous reqmts can never be verified

7. MODIFIABLE
keep all cross-references.
- easy to make changes in SRS retaining its structure + CORRECTLY Modify (A → B1, B2)

8. RANKED FOR STABILITY/IMPORTANCE

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Software Size Estimation

software size estimation

1. Lines Of Code

What is a LOC? loc

- Declarations, Actual Code including logic and computation

What is NOT a LOC?

- BLANK Lines line of code
  - → Included to improve readability of code.
- COMMENTS
  - → Included to help in code understanding as well as during maintainence.
  - → Not a LOC because
    1. Do not contribute to any kind of functionality
    2. Misused by developers to give a false notion about productivity

Advantage of using LOC for size estimation
→ very easy to count and calculate from the developed code.

Disadvantage of using LOC for size estimation
→ (1) LOC is language + technology dependant
→ (2) What constitutes an LOC → varies from org^n to org^n.

eg:

```
for(int i=1; i<10; i++)
    cout << i;          (2)
```

Program 1

```
for(int i=1; i<10; i++)
{
    cout << i;          (4)
}
```

Program 2

```
int a;  // Declaration
```
code | comment

```
// Declaration X
int a;          (1)
```
comment | code

Scanned with CamScanner

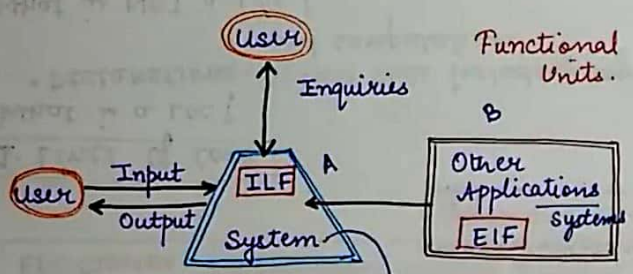
Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

functional points

Software Size Estimation

2. FUNCTION POINTS

* It measures functionality from user's point of view.
  - ↳ what the user recieves from SW + what the user requests from SW.
* Focuses on what functionality is being delivered.



A system has 5 types of functional units :-

1. **Internal Logical Files (ILF)**
   - The control info or logically related data that is present WITHIN the system.
2. **External Interface Files (EIF)**
   - The control data or other logical data i.e referenced by the system but present in another system

→ Data Function Types

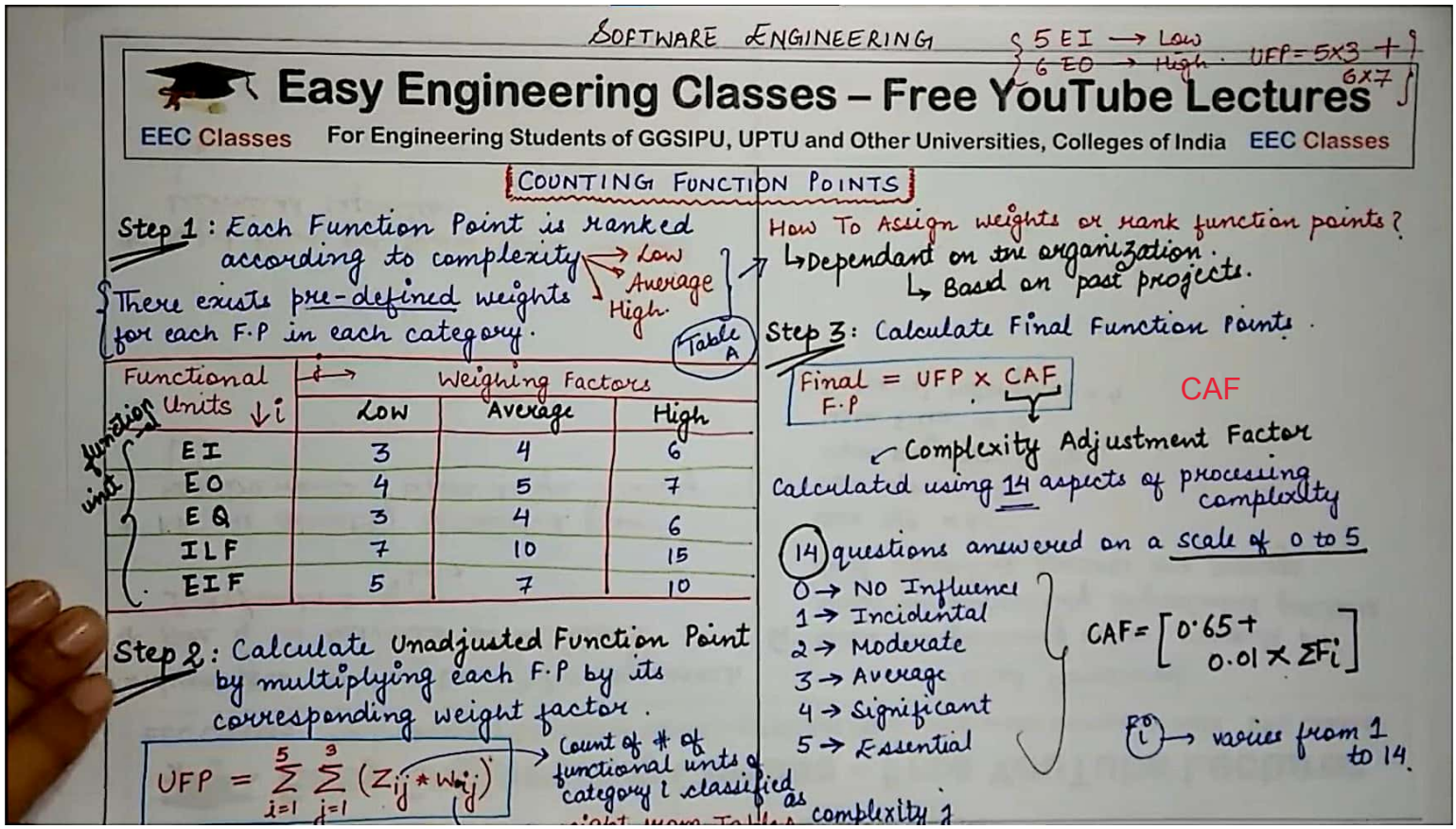
3. **External Inputs (EI)**
   data/control info that comes from outside our system
4. **External Outputs (EO)**
   data that goes out of the system after generation
5. **External Enquires (EQ)** → Combination of I/P - O/P resulting

→ Transaction Function Types.

Scanned with CamScanner

unadjusted function point

complexity adjustment factor



Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Advantages of Function Point Approach

* Size of SW delivered is measured independant of → language
  → technology & tools.
* FP are directly estimated from requirements, before design & coding
  → We get an estimate of SW size even before major design or coding happens (early phases).
  → Any change in reqmts can be easily reflected in FP count.
* Useful even for those users without technical expertise
  → Without F.P is based on external structure of the SW to developed.

Solved Numerical

Q. Given the following values, compute F.P when all complexity adjustment factors and weighting factors are (average) → CAF

User I/P = 50
User o/p = 40
User Enquiries = 35
User Files = 6
External Interfaces = 4

$\sum_{i=1}^{14} F_i \rightarrow 14 \times 3$

4,5,4,10,7

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} z_{ij} w_{ij}.$$

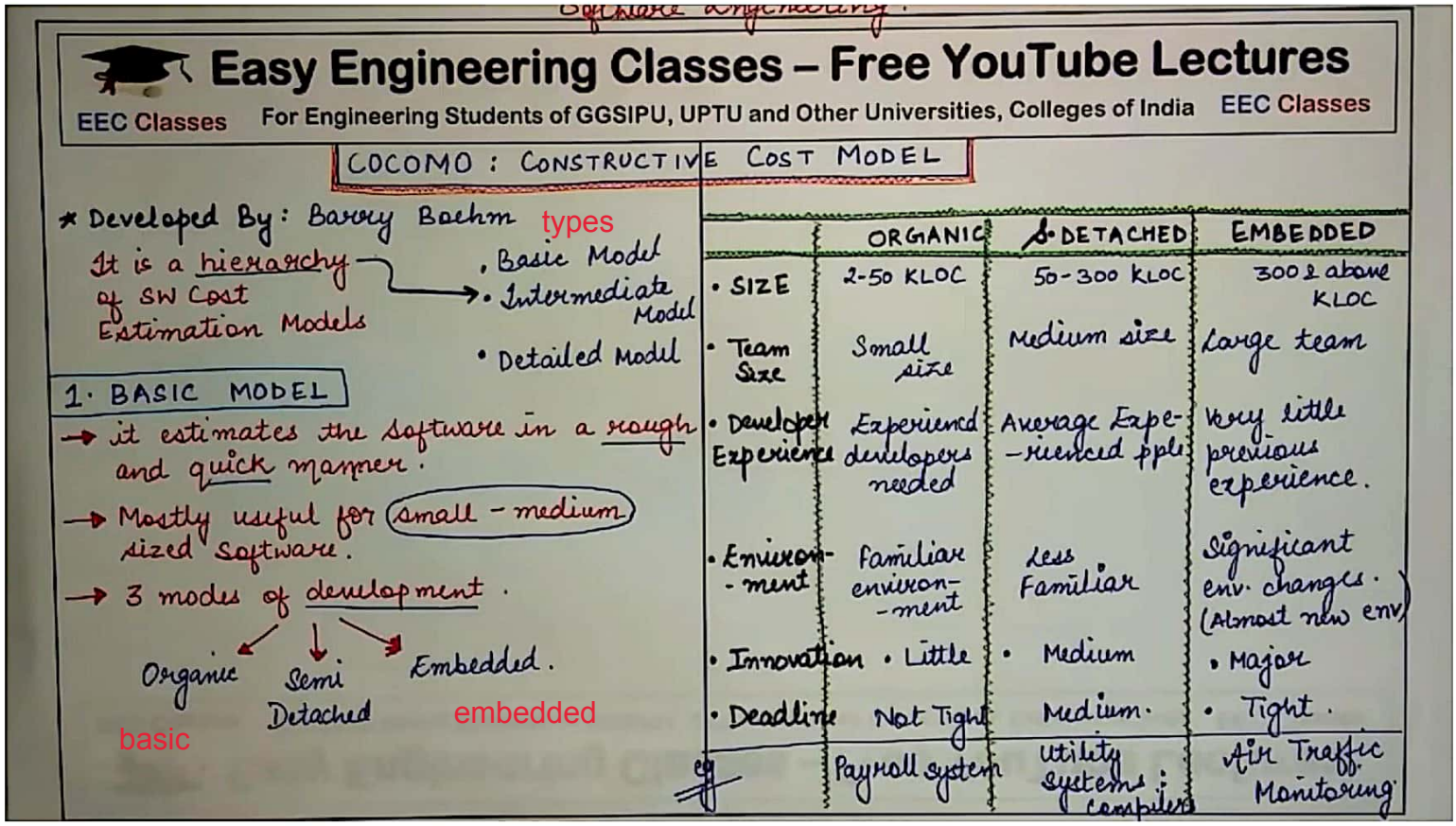
$UFP = 50 \times 4 + 40 \times 5 + 35 \times 4 + 6 \times 10 + 4 \times 7$
$= 200 + 200 + 140 + 60 + 28$
$= 628.$

$CAF = 0.65 + (0.01 \times \sum F_i) = 0.65 + \underbrace{0.01 (14 \times 3)}_{0.42}$
$= 0.65 + 0.42$
$= 1.07$

$F.P = UFP \times CAF$
$= 628 \times 1.07$

Scanned with CamScanner

cocomo



semidetacted

organic

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

basic model

BASIC MODEL EQUATIONS

1. Effort → $a\,(KLOC)^b$ person-months
2. Development Time → $c\,(Effort)^d$ months
3. Average staff size → $\frac{Effort}{Dev.\ Time}$ persons
4. Productivity → $\frac{KLOC}{Effort}$ KLOC/P.M

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

BASIC MODEL (CONTD)

Q). Suppose that a project (Basic) was estimated to be 400 KLOC. Calculate effort & time for each of 3 modes of development.

1. Organic :
   Effort $= 2.4 \times (400)^{1.05} \approx 1295$ PM
   Dev. Time $= 2.5 (1295)^{0.38} \approx$ 38 Months
2. Semi-detached
   Effort $= 3 \times (400)^{1.12} \approx 2462$ PM.
   Dev Time $= 2.5 \times (2462)^{0.35} \approx$ 38.4 Months
3. Embedded
   Effort $= 3.6 \times (400)^{1.2} \approx 4772$ PM
   Dev. Time $= 2.5 \times (4772)^{0.32} \approx$ 38. Months

Scanned with CamScanner

cocomo

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

COCOMO : The Intermediate Model

intermediate

★ **Difference between Basic & Intermediate Model**

1. Basic Model was quick but inaccurate and phase insensitive.
- Intermediate model includes a set of (15) additional Predictors (COST DRIVERS)
- It also takes development environment into account during cost estimation.

cost drivers

★ **Use of Cost Drivers**

Cost drivers adjust NOMINAL cost of project to actual project environment
↳ increasing the accuracy of estimation.

★ **15 COST DRIVERS**

I. Product Attributes
a) Required Software Reliability (RELY)
b) Database size (DATA)
c) Product Complexity (CPLX)

II Computer Attributes
a) Execution time constraint (TIME)
b) Main storage constraint (STOR)
c) Virtual Machine Volatility (VIRT)
d) Computer turnaround time (TURN)

III Personnel Attributes
a) Analyst Capability (ACAP)
b) Application experience (AEXP)
c) Programmer capability (PCAP)
d) Virtual Machine Experience (VEXP)
e) Programming Language experience (LEXP)

IV Project Attributes
a) Modern Programming Practices (MODP)
b) Use of software tools (TOOL)
c) Required development schedule (SCED)

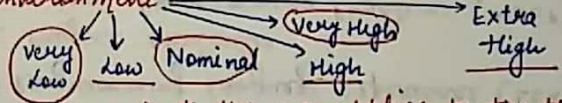
Scanned with CamScanner

effor adjustment factor eaf

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

COST DRIVER	VERY LOW	LOW	NOM-INAL	HIGH	VERY HIGH	EXTRA HIGH
Prod. Attr.						
RELY	0.75	0.88	1.00	1.15	1.40	–
DATA	–	0.94	1.00	1.08	1.16	–
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
Comp Attr.						
TIME	–	–	1.00	1.11	1.30	1.66
STOR	–	–	1.00	1.06	1.21	1.56
VIRT	–	0.87	1.00	1.15	1.30	–
TURN	–	0.87	1.00	1.07	1.15	–
Pers. Attr.						
ACAP	1.46	1.19	1.00	0.86	0.71	–
AEXP	1.29	1.13	1.00	0.91	0.82	–
PCAP	1.42	1.17	1.00	0.86	0.70	–
VEXP	1.21	1.10	1.00	0.90	–	–
LEXP	1.14	1.07	1.00	0.95	–	–
Project Attr.						
MODP	1.24	1.10	1.00	0.91	0.82	–
TOOL	1.24	1.10	1.00	0.91	0.83	–
SCED	1.23	1.08	1.00	1.04	1.10	–

* Each Cost Driver is rated for the given project environment



To what extent the C.D applies to the project

Effort Adjustment Factor (EAF)

↳ calculated by multiplying all the values that have been obtained after categorising each cost driver

Equations for COCOMO :

$$\text{Effort} = a_i (\text{KLOC})^{b_i} * \text{EAF} \quad \text{(person months)}$$

$$\text{Development Time} = c_i (\text{Effort})^{d_i}$$

↳ months.

MODE	a_i	b_i	c_i	d_i
Organic	3.2	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Scanned with CamScanner

COCOMO Intermediate Model - Numerical

Q A new project with estimated 400 KLOC embedded system has to be developed. Project Manager has a choice of hiring from 2 pools of developers: Very highly capable (with app.) with very little experience in programming language OR developers of low quality but a lot of progr. lang. experience. Which is better choice in terms of 2 pools?

$E = a(KLOC)^b \times EAF$

$E = 2.8(400)^{1.20} \times EAF$

Case I: $EAF = 0.82 \times 1.14 = 0.934$

$E = 2.8(400)^{1.20} \times 0.934$

$= 3470$ PM.

$D = 2.5(3470)^{0.32} = 33.9$ M

Case II

$EAF = 1.29 \times 0.95 = 1.22$

$E = 3412 \times 1.22 \approx 4528$ PM

$D = 2.5(4528)^{0.32} \approx 36.9$ M

Scanned with CamScanner

detailed development model

Universities, Colleges of India EEC Classes

COCOMO : Detailed Development Mode

Detailed Mode Is Phase Sensitive

it calculates the effect of Cost Drivers on each phase of SDLC.

* It uses phase-sensitive effort multipliers for each Cost driver to determine the amount of effort required to complete each phase of SDLC.

* It establishes Module - Subsystem - System Hierarchy
  ↳ The rating of cost driveners is done at that level only where the C.D is most susceptible to variable.

* Adjustment Factor (A)

$$A = 0.4(DD) + 0.3C + 0.3I$$

design code testing.

Size Equivalent → $(S \times A)/100$

Scanned with CamScanner

Mode and code size	Plan & Reqmt	System Design	Detail Design	Code & Test	Integ^n & Test
Life Cycle Phase value of μ_p					
Organic Small S=2	0.06	0.16	0.26	0.42	0.16
Org. Medium S≈32	0.06	0.16	0.24	0.38	.22
S.Det. Medium S≈32	0.07	0.17	0.25	0.33	.25
S.Det. Large S≈128	0.07	0.17	0.24	0.31	.28
Embed. Large S≈128	0.08	0.18	0.25	0.26	.31
Embd. XL S≈320	0.08	0.18	0.24	0.24	.34
Life Cycle Phase value of τ_p					
Organic Small S≈2	.10	.19	.24	.39	.18
Org. Medium S≈32	.12	.19	.21	.34	.26
S.Det Med S≈32	.20	.26	.21	.27	.26
S.Det Large S≈128	.22	.27	.19	.25	.29
Embed Large S≈128	.36	.36	.18	.18	.28
Embd XL S≈320	.40	.38	.16	.16	.30

✓✓ Effort = $\mu_p E$

Development Time = $\tau_p D$.

Scanned with CamScanner

Q Consider a project with the following main components (1) Screen Edit (2) CLI (3) File I/P and O/P (4) Cursor Movt (5) Screen Movement The sizes for these are estimated to be 4K, 2K, 1K, 2K, 3K LOC. Using COCOMO, determine

I. Overall cost and schedule estimates (assume values for cost drivers, with atleast 3 being different from 1.0)

II. Cost and schedule estimates for different phases.

$$Effort_i = a_i (KLOC)^{b_i} \times EAF$$

(a_i → 3.2, KLOC → 12, b_i → 1.05)

SW Reliability : High
Lang. Experience : Low
Product Complexity : High
Analyst Capability : High.

$$EAF = 1.15 \times 1.15 \times 0.86 \times 1.07 \approx 1.216$$

$$Effort = 3.2(12)^{1.05} \times 1.216$$
$$\approx 52.9 \text{ PM.}$$

$$\text{Dev. Time} = c_i (E)^{d_i}$$
$$= 2.5 (52.9)^{0.38}$$
$$= 11.29 \text{ Months}$$

Scanned with CamScanner

(II) $Effort_d = \mu_p \times E$

Development Time $= T_p D$

Effort

Plan & Reqmt	: 0.06×52.91	} Person Months.
Design	: 0.16×52.9	
Detail Design	: 0.26×52.9	
Code & Test	: 0.42×52.9	
Integ^n & Test	: 0.16×52.9.	

Development Time:

Plan & Reqmt	: 0.10×11.29	Months.
Design	: 0.19×11.29	"
Detail Design	: 0.24×11.29	"
Code & Test	: 0.39×11.29	"
Integ^n & Test	: 0.18×11.29.	"

Scanned with CamScanner

cocomo 2 not in syllabus

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

COCOMO - II MODEL

- Developed by Barry Boehm
- Revised version of original COCOMO.
- Includes 3 stages
  (1) Application Composition Estimation
  (2) Early Design Estimation
  (3) Post - Architechture Estimation

Application Composition Estimation Model

- Focuses on those applications which can be quickly developed using interoperable components.

eg: GUI, database manager, control packages for specific domains.

* Can also be used during prototyping phase of an application.
* Size is estimated using Object Points:
  1) Screens 2) Reports 3) 3GL component

Steps for Effort Estimation

1. Assess Object Counts

Estimate the number of screens, reports and 3GL components.

2. Classify complexity levels of each object

- Classify each object into Simple, Medium, Difficult.
- Screens on basis of # of views + sources.
  Reports on the basis of # sections + sources.

Screens

No. of views contained	Number & Sources of Data Tables: Total < 4 (< 2 server) (< 3 client)	Total < 8 (2-3 server 3-5 client)	Total 8+ (> 3 server, > 5 client)
< 3	SIMPLE	SIMPLE	MEDIUM
3-7	SIMPLE	MEDIUM	DIFFICULT
> 8	MEDIUM	DIFFICULT	DIFFICULT

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Number of sections (Reports)	Number and Sources of Data Tables		
	Total < 4 (< 2 server, < 3 client)	Total < 8 (2-3 server, 3-5 client)	Total 8+ (> 3 server, > 5 client)
0 or 1	SIMPLE	SIMPLE	MEDIUM
2 or 3	SIMPLE	MEDIUM	DIFFICULT
4+	MEDIUM	DIFFICULT	DIFFICULT

3. Assign Complexity weight to each object

depending on the category of the object

Object Type	COMPLEXITY WEIGHT		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL Component	-	-	10

4. Determine object points

Add all these complexity weights to get (Total) object points.

5. Compute new object points

↳ by considering reusable components

$$NOP = \frac{(\text{object points}) * (100 - \%\ \text{reuse})}{100}$$

6. Calculate productivity rate (PROD)

Developer Experience & Capability; ICASE Maturity + Capability	PROD (NOP/PM)
Very Low	4
Low	7
Nominal	13
High	25
Very High	50

$$PROD = \frac{NOP}{\text{Person-Month}}$$

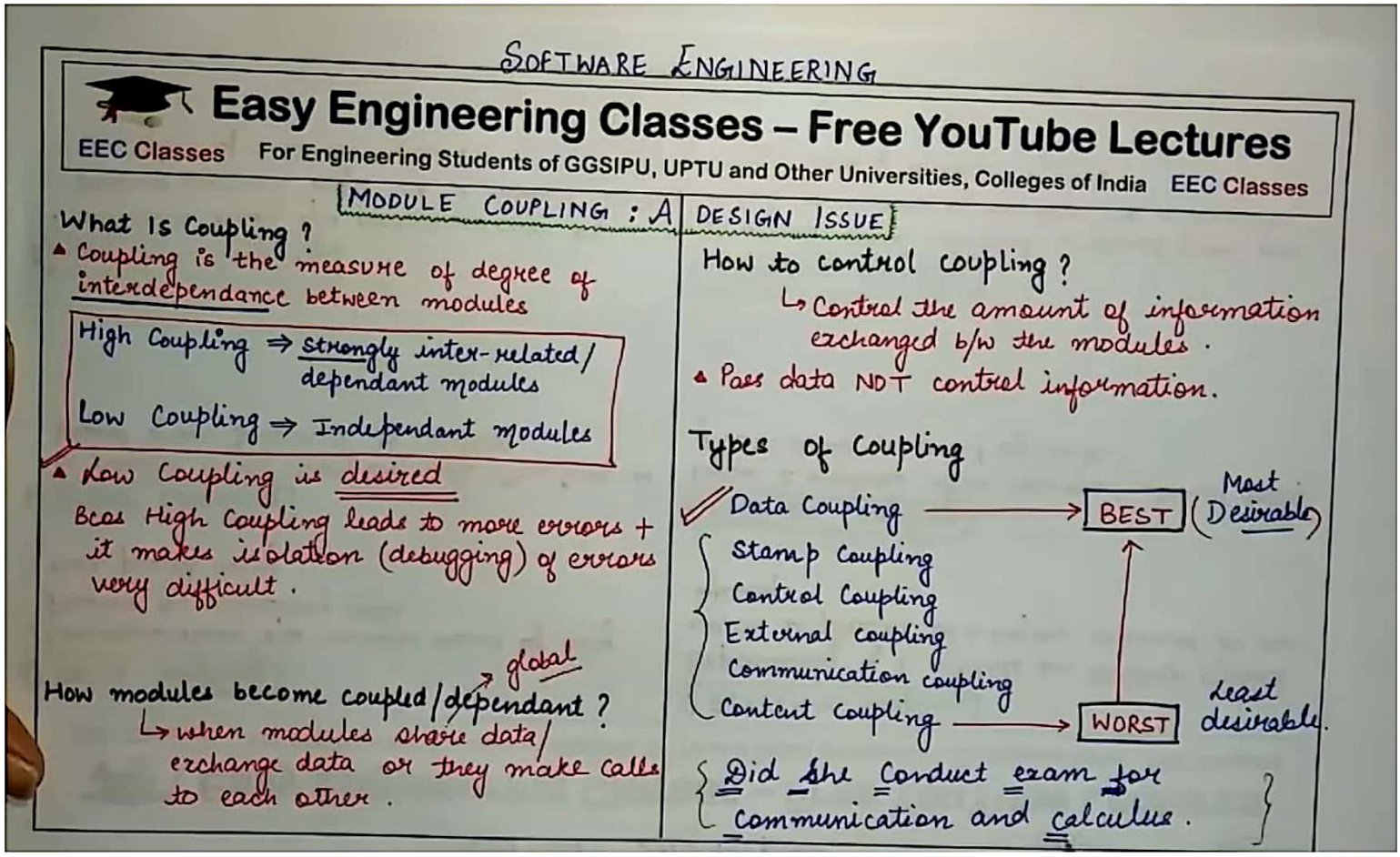
Calculated depending on the past project data

7. Compute Effort in Person-Months

$$Effort = \frac{NOP}{PROD} \text{ person-months.}$$

Scanned with CamScanner

coupling



Scanned with CamScanner

types of coupling

SOFTWARE ENGINEERING

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

1) DATA Coupling

data coupling

- Communication b/w modules occur by only passing the necessary data (No control info)
- Data passed using parameters.

2) STAMP Coupling

stamp

It occurs when complete data structure is passed from 1 module to another

Drawback : The entire data structure is not reqd by receiving module, only a part of d.s is needed.

3) CONTROL Coupling

contol

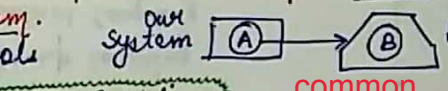
- Communication b/w modules occurs by passing control info (OR) a module controls the flow of another module.
- eg : flags, data that is set by 1 module & used by another module.

4) EXTERNAL Coupling

external

Dependency of a module on another module which is present in a SW/HW external to the system.

eg: tools



5) COMMON Coupling

common

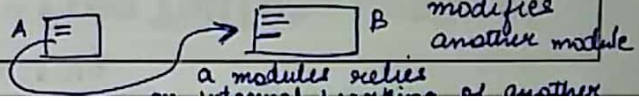
When 2 modules have common shared data/ global data (accessed by both)

▲ Shared data makes error isolation difficult. OR evaluation of change to shared data is difficult.

6) CONTENT Coupling

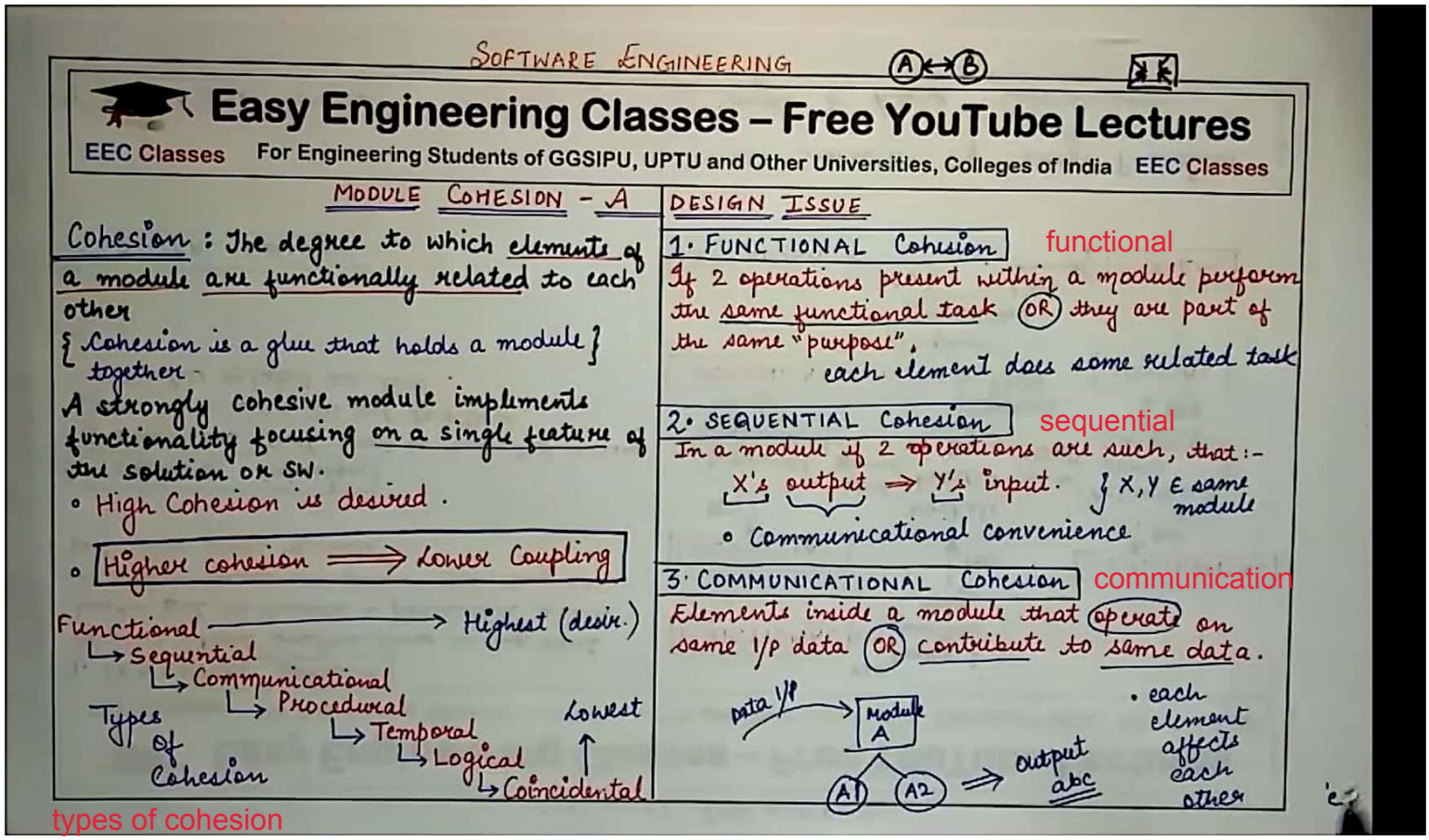
content

It occurs when control is passed from one module to middle of another OR a module changes data of another module.



Scanned with CamScanner

cohesion



SOFTWARE ENGINEERING

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

MODULE COHESION - A

Cohesion : The degree to which elements of a module are functionally related to each other

{ Cohesion is a glue that holds a module together }

A strongly cohesive module implements functionality focusing on a single feature of the solution or SW.

- High Cohesion is desired.
- Higher cohesion ⟹ Lower Coupling

Types of Cohesion
- Functional → Highest (desir.)
- Sequential
- Communicational
- Procedural
- Temporal
- Logical
- Coincidental ← Lowest

DESIGN ISSUE

1. FUNCTIONAL Cohesion

functional

If 2 operations present within a module perform the same functional task (OR) they are part of the same "purpose", each element does some related task

2. SEQUENTIAL Cohesion

sequential

In a module if 2 operations are such, that :-

X's output ⇒ Y's input. { X, Y ∈ same module

- Communicational convenience

3. COMMUNICATIONAL Cohesion

communication

Elements inside a module that operate on same I/p data (OR) contribute to same data.

- each element affects each other

types of cohesion

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

4. PROCEDURAL Cohesion

procedural

Modules whose instructions do different tasks, but to ensure a particular order in which tasks are performed, they are put into same module

(A1) → (A2) - - - - - > (A9)

• Poor Maintenance

5. TEMPORAL Cohesion

temporal

• Instructions that must be executed during same time span are put together.
(Activities related in time).

Eg: Boot set up ← A1, B1, C1 → Booting

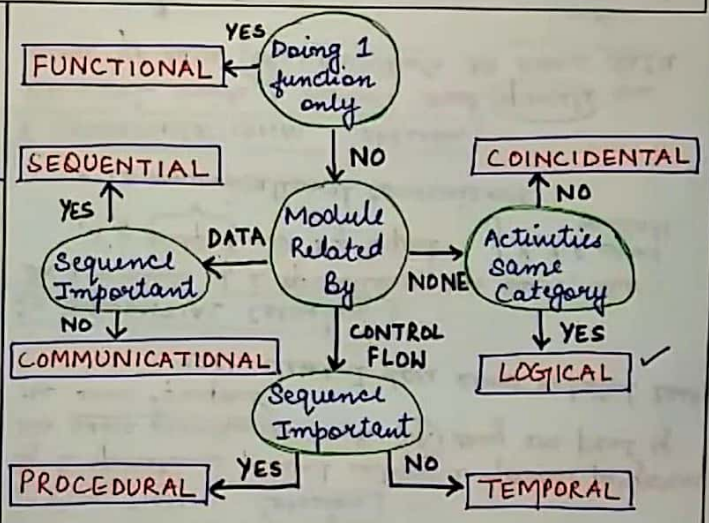
6. LOGICAL Cohesion

logical ✗

↳ When modules have logically similar instructions/elements

7. COINCIDENTAL Cohesion

→ Instructions are mostly not related to each other.



{ FIRST Semester Of College Provided Tough & Long Curriculum. }

Scanned with CamScanner

Evolutionary Process Models

Evolutionary process model resembles iterative enhancement model. The same phases as defined for the waterfall model occur here in a cyclical fashion. This model differs from iterative enhancement model in the sense that this does not require a useable product at the end of each cycle. In evolutionary development, requirements are implemented by category rather than by priority.

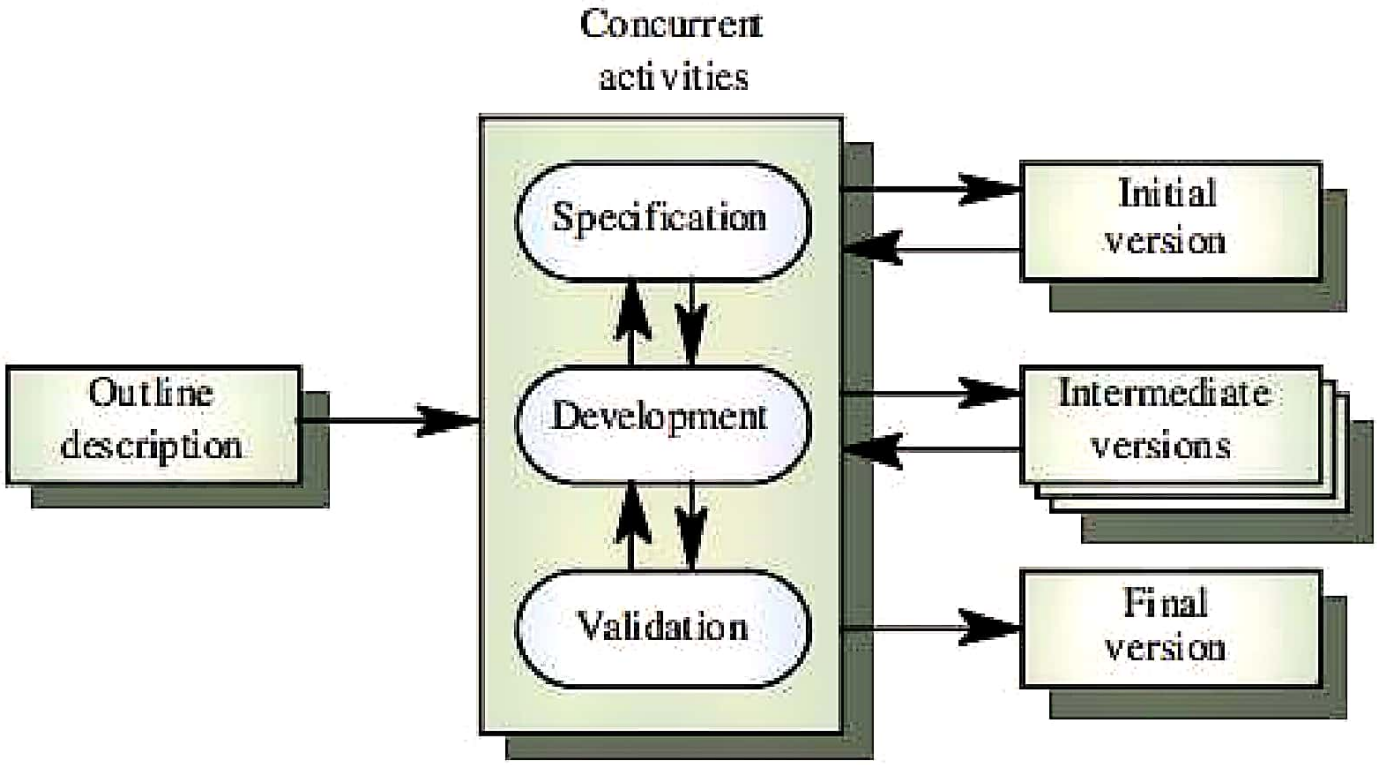
evolutionary process models

This model is useful for projects using new technology that is not well understood. This is also used for complex projects where all functionality must be delivered at one time, but the requirements are unstable or not well understood at the beginning.

scroll

Scanned with CamScanner

Evolutionary Process Model



Scanned with CamScanner

Prototyping Model

- The prototype may be a usable program but is not suitable as the final software product.
- The code for the prototype is thrown away. However experience gathered helps in developing the actual system.
- The development of a prototype might involve extra cost, but overall cost might turnout to be lower than that of an equivalent system developed using the waterfall model.

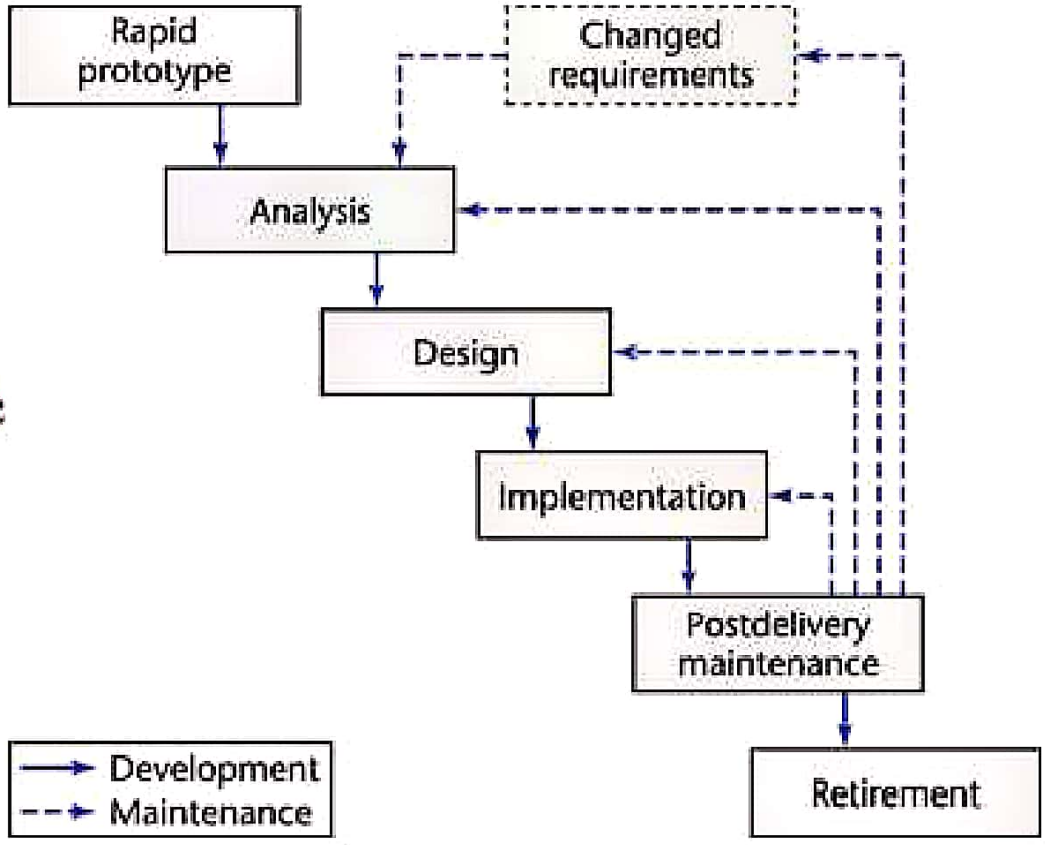
Scanned with CamScanner

scroll up

prototyping model

Prototyping Model

- Linear mode
- "Rapid"



Scanned with CamScanner

Incremental Process Models

They are effective in the situations where requirements are defined precisely and there is no confusion about the functionality of the final product.

incremental

After every cycle a useable product is given to the customer.

Popular particularly when we have to quickly deliver a limited functionality system.

Scanned with CamScanner

Iterative Enhancement Model

This model has the same phases as the waterfall model, but with fewer restrictions. Generally the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles. Useable product is released at the end of the each cycle, with each release providing additional functionality.

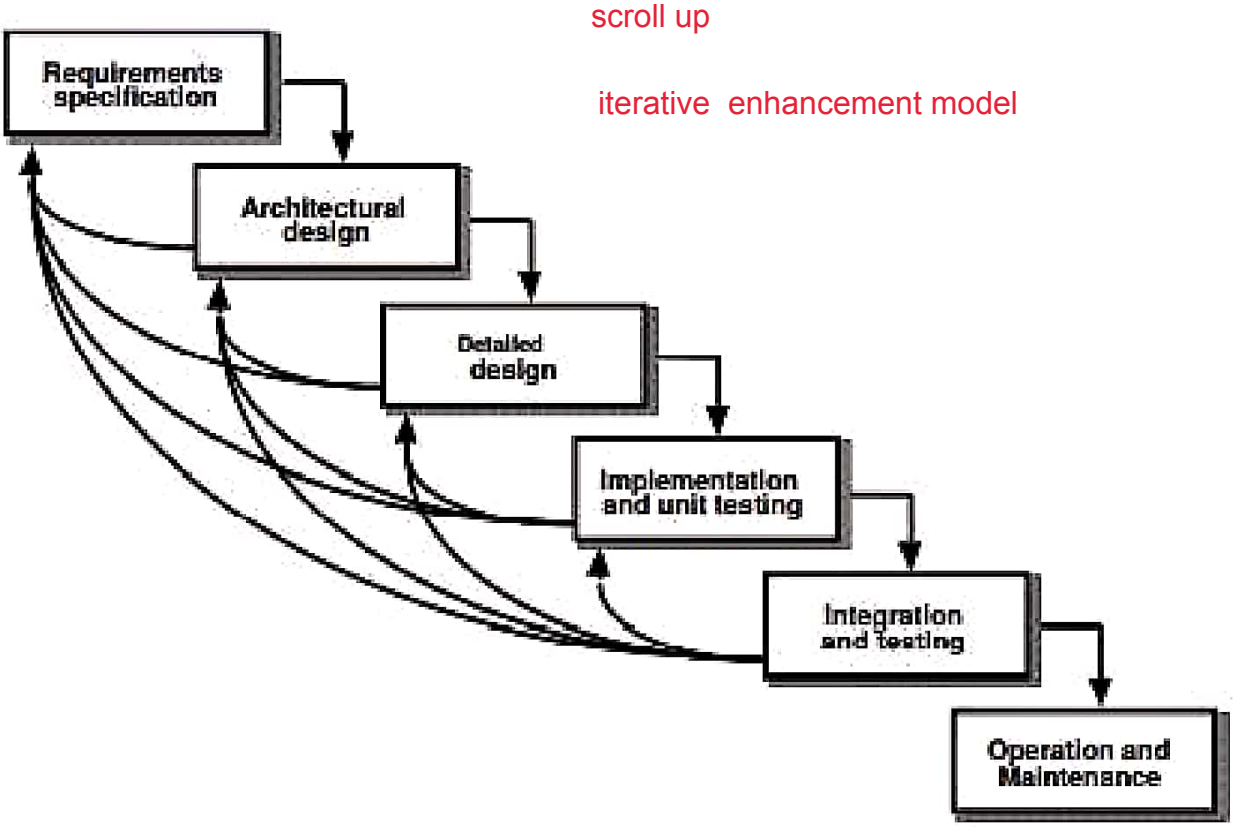
- ✓ Customers and developers specify as many requirements as possible and prepare a SRS document.
- ✓ Developers and customers then prioritize these requirements
- ✓ Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

Scanned with CamScanner

Iterative Enhancement Model

scroll up

iterative enhancement model



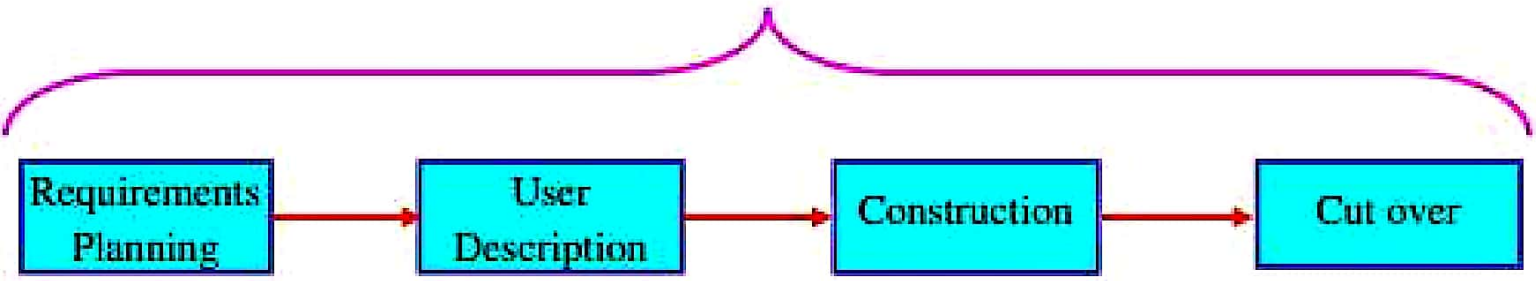
Scanned with CamScanner

The Rapid Application Development (RAD) Model

- Build a rapid prototype
- Give it to user for evaluation & obtain feedback
- Prototype is refined

RAD

With active participation of users



Scanned with CamScanner

The Rapid Application Development (RAD) Model

Not an appropriate model in the absence of user participation.

Reusable components are required to reduce development time.

Highly specialized & skilled developers are required and such developers are not easily available.

Scanned with CamScanner

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**.

The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

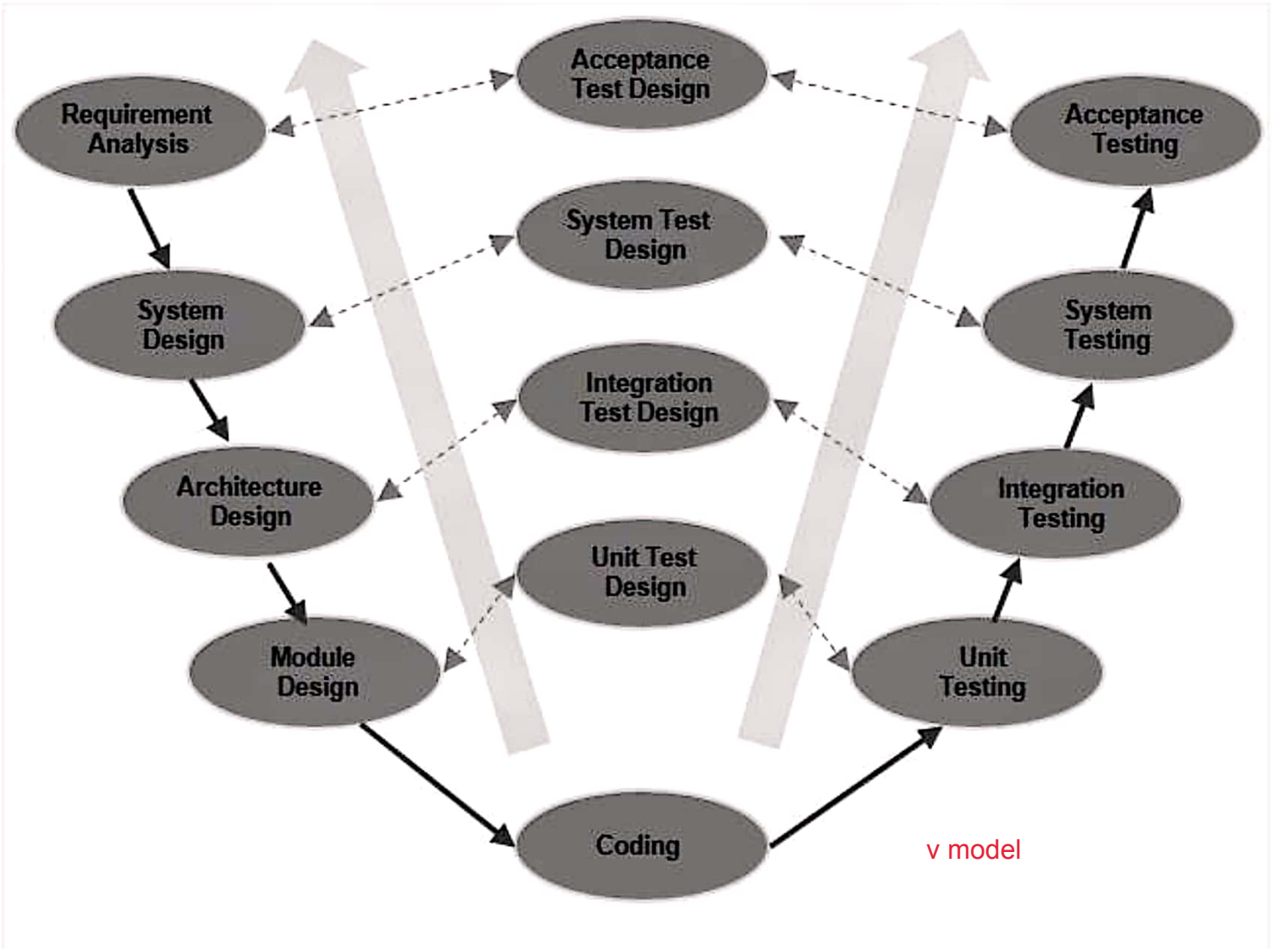
V-Model - Design

V model

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.

Scanned with CamScanner



v model

Scanned with CamScanner

function oriented design

Software Engineering

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Function Oriented Design

It is an approach to software design where the design is decomposed into a set of interacting units where each unit has a clearly defined function.

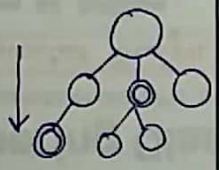
↳ System is designed from functional viewpoint.

A Generic Procedure

- Start with a high level description of what the SW / program does.
- Refine each part of the description one by one by specifying in greater detail the functionality of each part

⇓

Top Down Structure



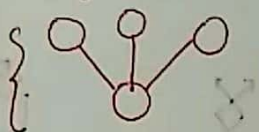
PROBLEM: Mostly each module is used by almost 1 other module ⇒ parent.

SOLUTION:

↳ Designing of reusabable modules.

modules use several other modules to do their required functions.

Reusable design.



For a function - oriented design, design can be represented mathematically or graphically by using :-

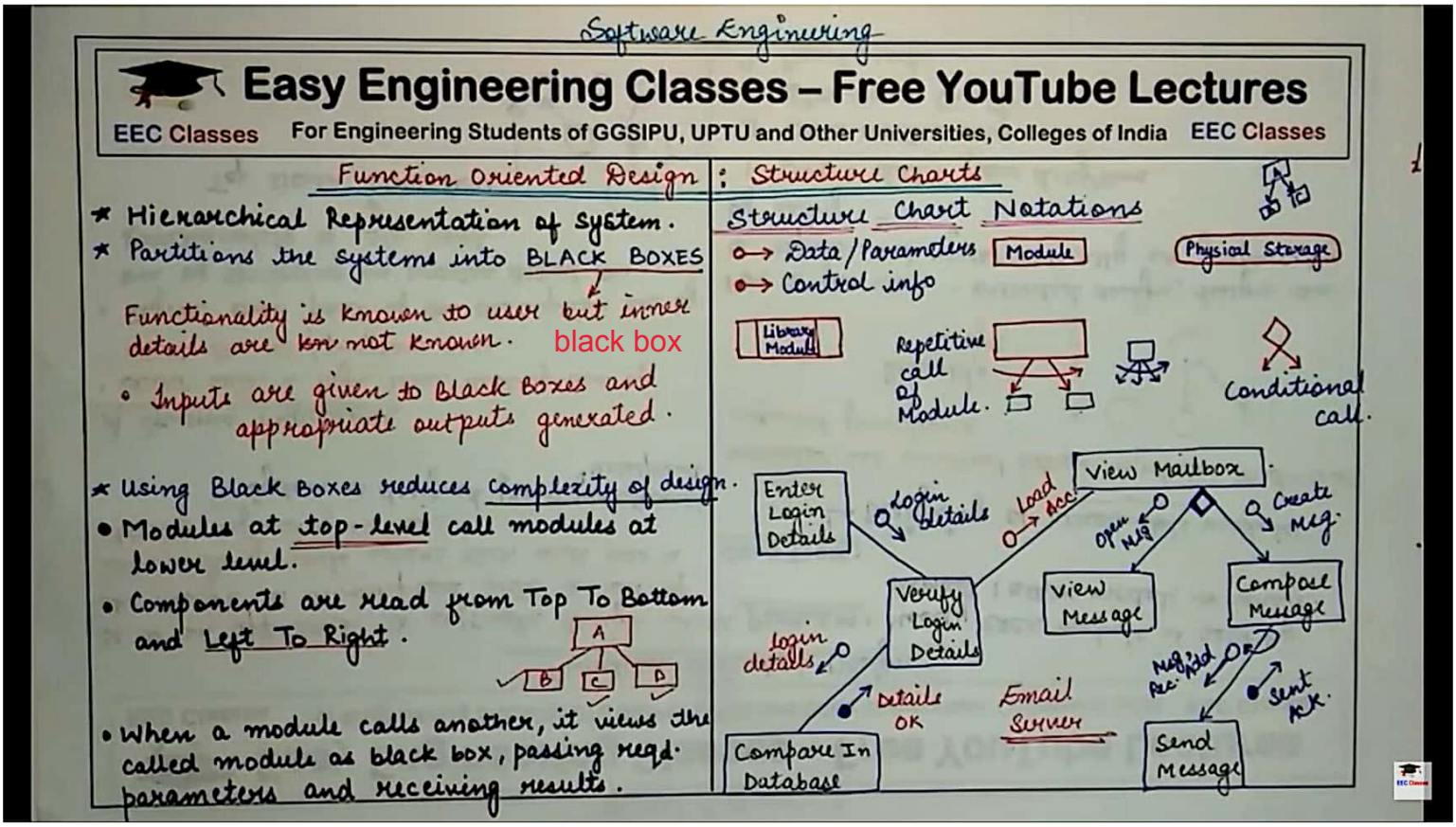
1. DFD : data flow diagram
2. Data dictionary
3. Structure Charts.
4. Pseudocode.

types

ashish

Scanned with CamScanner

structure charts



Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Types of Structure Charts

Transform - Centred Structures

structures

These type of structure charts are designed for systems that receive an input which is transformed by a sequence of operations, with each operation being carried out by one module.

Transaction - Centred Structures

These structures describe a system that processes a number of different types of transactions.

Pseudo Code

pseudocode

* Useful in both preliminary and detailed design phases.
* System description using short English like phrases describing the function.

- Keywords and Id Indentation: describes the flow of control.
- English phrases: processing actions performed.

Advantage

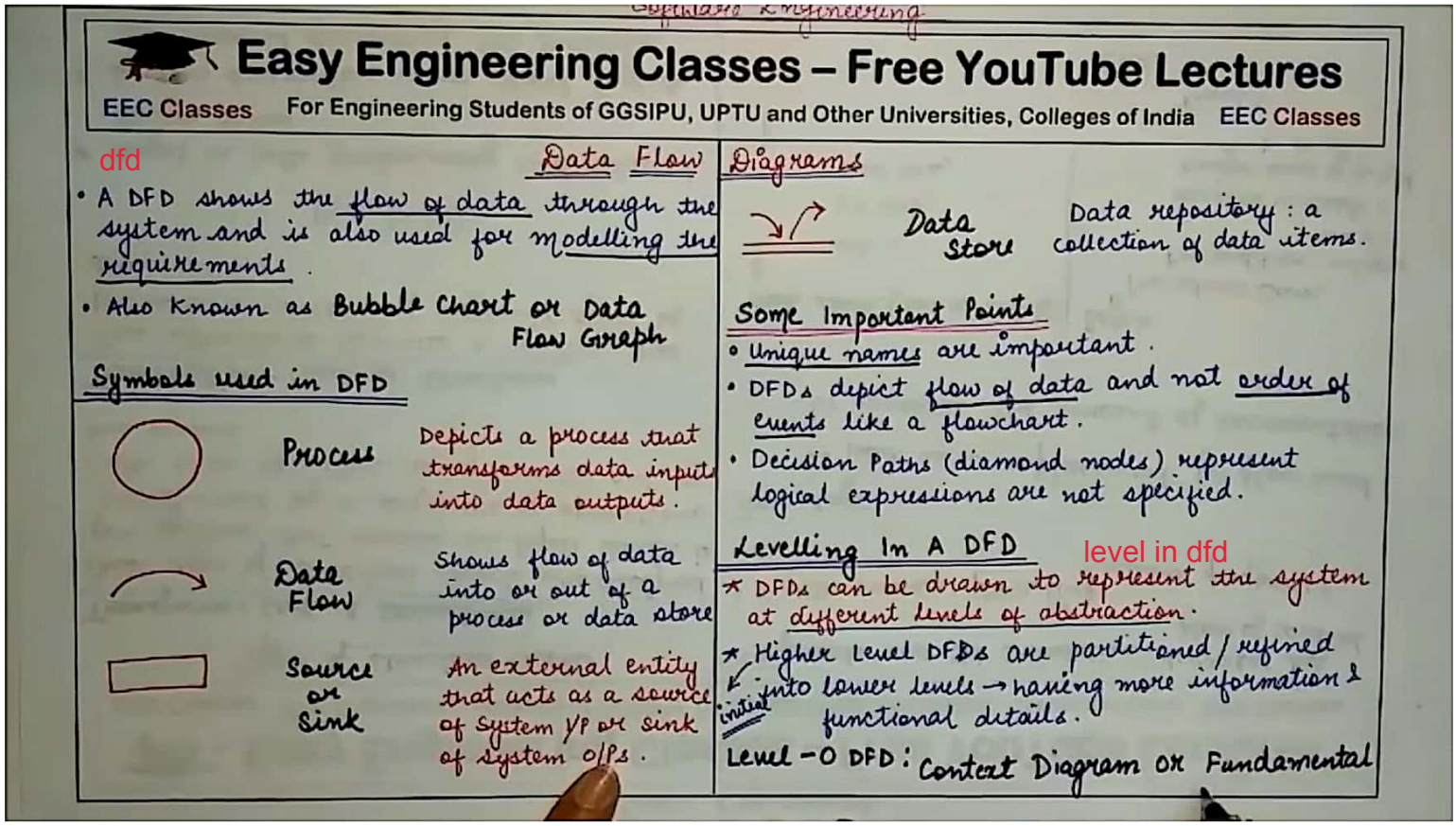
↳ used as a replacement for flow chart
↳ decrease the amount of documentation required.

```
void sum (int a, int b)
{
    int c;
    c = a+b;
    cout << c;
}
```

```
Begin:
  Procedure: Sum
  Precondition: integers
                a & b.
  declare variable c
  Assign sum of a and
         b to c.
  Print c
End.
```

Scanned with CamScanner

data flow diagram



Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

dfd

Data Flow Diagrams

- A DFD shows the flow of data through the system and is also used for modelling the requirements.
- Also known as Bubble Chart or Data Flow Graph

Symbols used in DFD

Process	Depicts a process that transforms data inputs into data outputs.
Data Flow	Shows flow of data into or out of a process or data store
Source or Sink	An external entity that acts as a source of system I/P or sink of system O/Ps.

Data Store — Data repository : a collection of data items.

Some Important Points

- Unique names are important.
- DFDs depict flow of data and not order of events like a flowchart.
- Decision Paths (diamond nodes) represent logical expressions are not specified.

Levelling In A DFD

level in dfd

* DFDs can be drawn to represent the system at different levels of abstraction.
* Higher Level DFDs are partitioned / refined into lower levels → having more information & functional details.

Level – 0 DFD : Context Diagram or Fundamental

Scanned with CamScanner

Software Engineering

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Level-0 DFD represents the entire system as a single bubble with Input and Output data indicated by incoming & outgoing arrows.

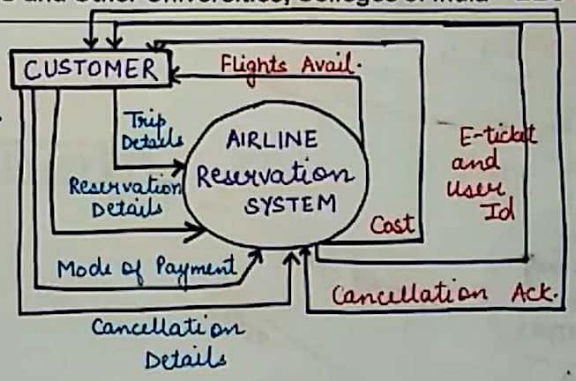
⇓

Decompose this DFD into multiple bubbles.

⇓

Each bubble is then decomposed into more detailed DFDs.

✓ Preserve the * inputs and outputs between different levels of DFD



Level-0 - DFD

Inputs Outputs

level 0

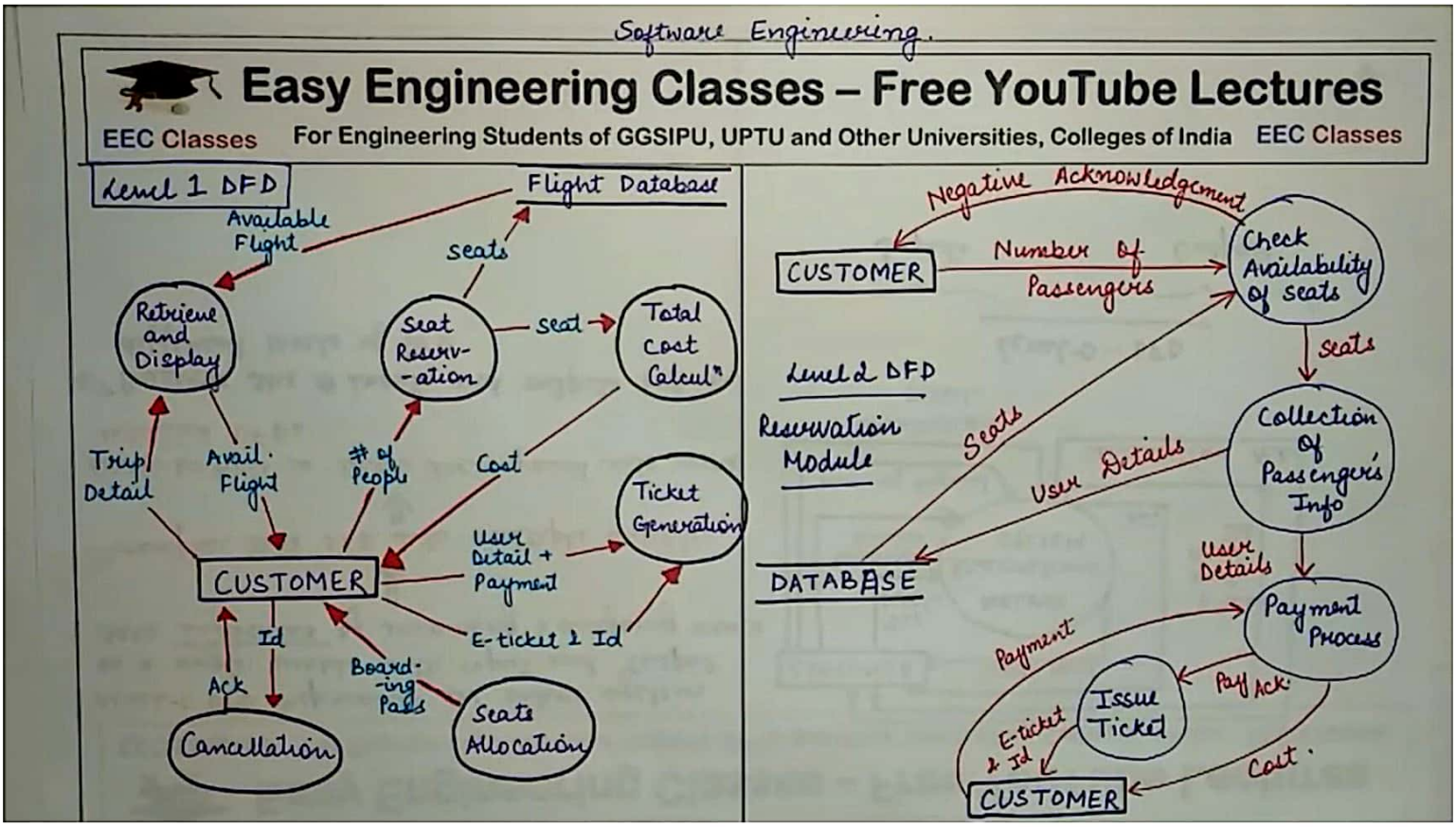
Scanned with CamScanner

level 1

level 2

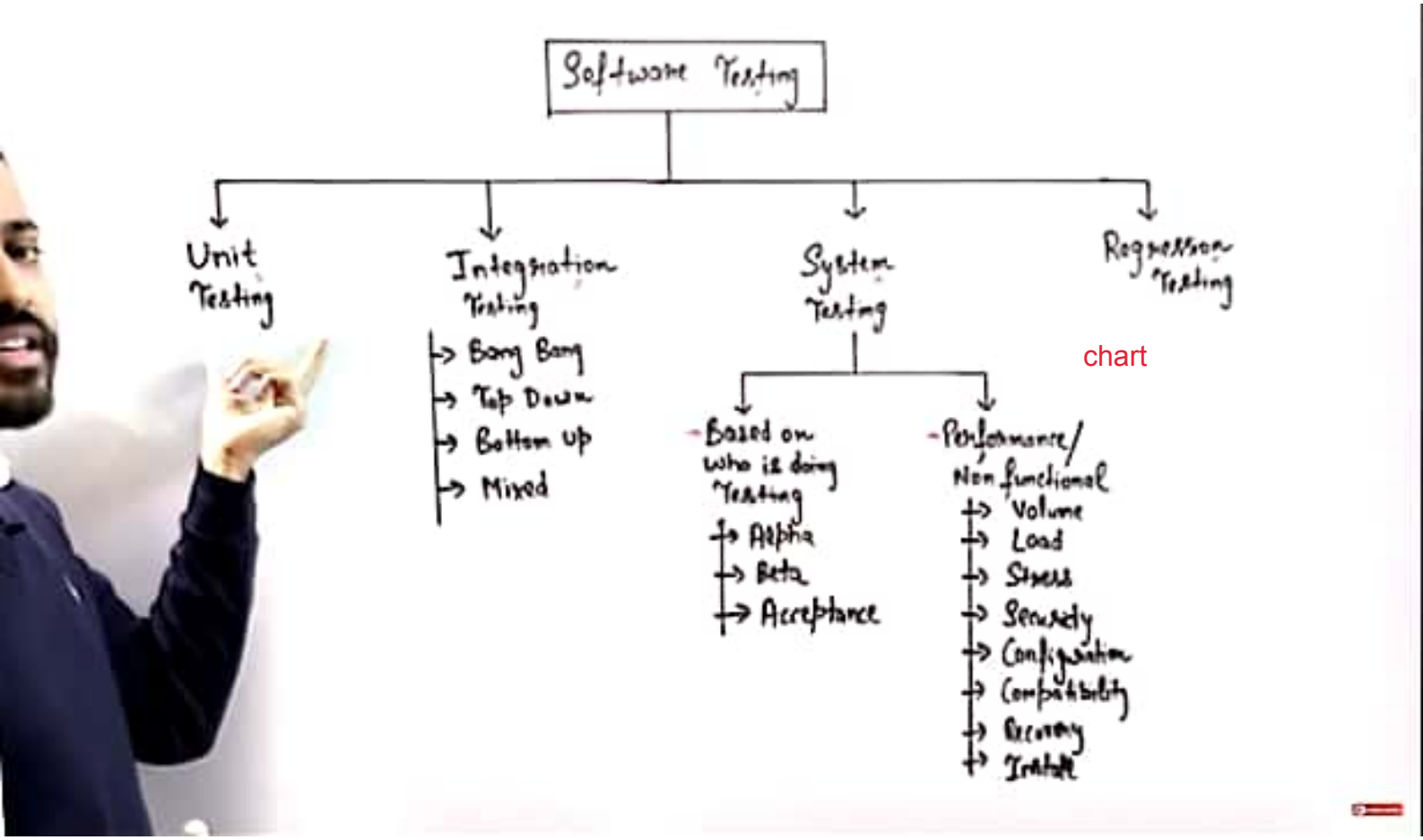
flight

resevation module



Scanned with CamScanner

software testing

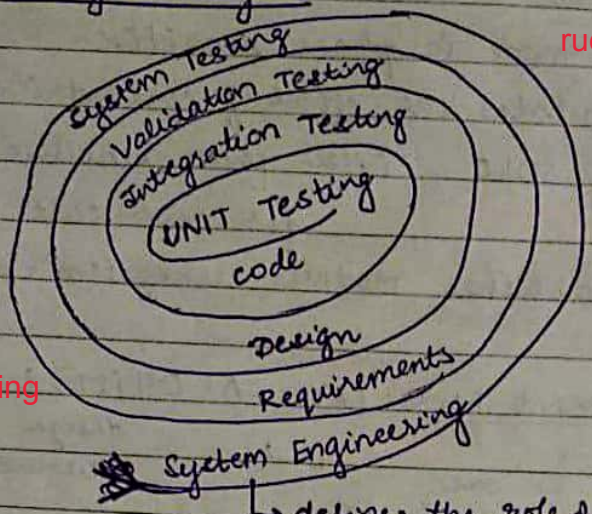


chart

orz

Scanned with CamScanner

Software Testing Strategies



rudr

s/w is 'developed' moving inwards in the spiral

'Tested' outwards

software testing

↳ defines the role of the s/w. (performance constraints, Validation criterion, functions of s/w)

Unit Testing

Tests each module/ component individually (ensures complete coverage of a single unit)

* If there is an interactn b/w 2 modules, a dummy interactn is rather taken.

types

Integration Testing

components are integrated together step by step to form a complete s/w.

Validation Testing

Requirements are validated against the developed s/w

System Testing

s/w is tested with other system elements as a whole

unit testing

UNIT TESTING : Process of taking a module & testing it in isolation from the rest of the s/w & comparing the actual results with the results defined in the specifications & design of module

Why isolatn? → Fault isolatn & debugging is easier
→ Divide & Conquer, exhaustive approach for each module

What is tested?
→ Module Interfaces : to check correct flow of info in & out of module (results, parameters)

विकार उत्पन्न करें ऐसे वस्त्रों और शृंगार से बचो।

* Drivers

Scanned with CamScanner

→ Local Data Structure : to ensure intermediate data is stored properly.

→ Independent / Basis paths : to ensure all statements are executed atleast once.

→ Boundary conditions : O/P / computation at boundary values is correct.

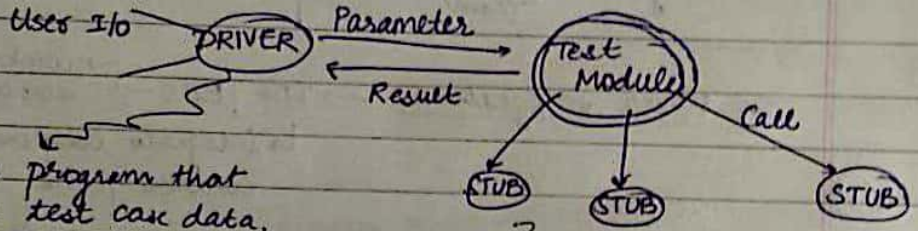
→ Internal logic

→ Error Handling Paths

PROBLEMS: Nobody is calling it, it is calling nobody.

↳ solution: DRIVER & STUB

driver



Main program that accepts test case data, passes data, & prints relevant results.

Subordinate Modules, called by module to be tested

→ A dummy sub-program that does min data manipulatn, provides verificatn of entry & returns the control to module under testing.

stub

Scaffolding : the overhead code

↳ code in drivers & stubs

↳ Test Harness

↳ used to auto generate scaffolding.

scaffolding

INTEGRATION TESTING

Determine the correctness of the interface.

Why interface check? → Data may be lost

→ Global data synchronizn

→ Subfunctns malfunctn when combined together.

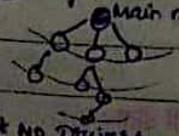
integration testing

Method- unit comp. taken one by one, incrementally.

↳ debugging, fault location

Types of Integration- types

① Top Down ② Bottom Up ③ Sandwich

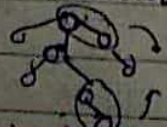


* No Drivers
* Stubs

top down

* No Stubs
* Drivers

bottom up



sandwich

विकार उत्पन्न करे ऐसे वस्त्रों और शृंगार से बचो।

Knowledge is wealth

Scanned with CamScanner

verfication and validation

Verification and Validation

at end of development phase

Verification – Are we building the system right?
Validation – Are we building the right system?

Manual Testing (look & review the documents)	Program execution
Reviews, Meetings, (Activities) inspections	Black Box & White Box Testing
carried out by Internal Quality Assessment Team	carried out by testing team

at end of development process

Top Down Integration → DFS / BFS

- combines all comp. directly subordinate to a given module
- integrate components on a major control path

top down

Steps

① Stubs are substituted for all subordinates
② Stubs are replaced one at a time with actual components
most BFS/DFS

Overhead

Bottom Up Integration

bottom up

Steps ① Low-level components are combined into CLUSTER/ BUILDS
② Driver coordinates the test case I/P-O/P. Cluster is tested. Driver is removed & clusters are combined with new modules.

regression

REGRESSION TESTING : Addn of a new module in integratn may cause problems with the fns that previously worked correctly.

↳ Re-execution of some subset of test cases that have already been conducted.

Can be done manually / Automated
↳ capture/playback tools
(capture test cases & results for playback)

What tests are conducted?
- check all modules
- check those components that have changed
- check those most likely to get affected by integration

जो कभी निराश नहीं होता।

Knowledge is wealth

Scanned with CamScanner

VALIDATION TESTING: validation testing

focuses on 'user'-visible actions & 'user' recognizable output.

Criteria
- Validatn criterion is a section in SRS, describes desired functionality that forms basis of validatn-testing.

How? Through tests which ensure
- → functionality is achieved
- → correct behaviour is achieved
- → performance constraints met

A deficiency list is prepared after testing, & required changes are done.

SYSTEM TESTING:

Incorporates the s/w with other system elements. system testing
H/W, other libraries/ S/W modules.

Types- recovery

⇒ **RECOVERY TESTING**: system must recover from faults & resume processing with little downtime. (Fault Tolerance)

types

① Entire system not to be affected
② Less recovery time

How?
- Force system to fail in diff ways.
  - Manual Testing → check mean-time to recover
  - Automated Testing → check correct re-initialisns

security

⇒ **SECURITY TESTING**: verify protection mechanism

How? Tester's Responsibility
- acts as intruder/attacker
- ensures that cost of the attack is higher than the info achieved. (Low efficiency of attacking)

stress

→ **STRESS TESTING**: verify situations where system demands resources in abnormal quantity.
- Excessive memory requirement
- DB requests
- simultaneous login

sensitivity

⇒ **SENSITIVITY TESTING**: variatn of stress testing
- check for data combinns in valid I/P domain, but may cause improper functioning.

यही अच्छा साहसी है जो कभी निराश नहीं होता।

Knowledge is wealth

Scanned with CamScanner

performance

→ PERFORMANCE TESTING : Test run-time performance of S/W. Happens from UNIT testing itself.

deployment

→ DEPLOYMENT TESTING: Configuration Testing
(Diff. environments, all types of installation procedures, documentations req.

acceptance

* ACCEPTANCE TESTING : Conducted when S/W is developed for a specific customer & not for a large public. Allows customer to validate (check wrt SRS) all requirements.

How? Customer tests S/W → Feedback → -ve → changes done → Final delivery

* ALPHA & BETA TESTING — alpha beta testing

→ Customer is anonymous.
But, → carried out by the customer.

ALPHA TESTING	BETA TESTING
① Done by customer at developer's site	① Done by customer at user's site. (β-version of an app)
② conducted in a controlled environment (Hence, wrong I/Ps will be in check)	② Real-time environment, not under developer's control.
③ Carried out before release of product to customer	③ after
④ Error/failures are recorded	④ Reported
⑤ White + Black Box Testing → Internal structure is also tested.	⑤ Black Box Testing

विचार उत्पन्न करें ऐसी चर्चा और शृंगार से बचो।

Scanned with CamScanner

MODULE COUPLING (Design Issue)

Coupling - measure of degree of interdependence b/w modules.

High coupling - strongly dependent & vice versa

Low coupling is desired.
(fault isolation / easier debugging)

Reason for coupling ⇒ when modules share data or make calls to each other

Controlling coupling ⇒ control the amt. of info exchanged.
Pass the data, not Control information.

Types of Coupling-

Data coupling (most desirable) ['passing only' the necessary 'data' using parameters]

Stamp → complete DS is passed (even if complete DS not req.)

Control → a module controls diff. & another module eg. flags

External

Communication

Content coupling (worst)

Module depends upon a module external to the system ex. tools (ext.)

control is passed from one module to middle & another (changes data & another module)

2 modules share common/shared data / global data (not as parameters, shared!)

MODULE COHESION

Degree to which elements of (same) module are functionally related to each other. (cohesion is like a glue to hold a module)

Higher the cohesion, lower the coupling. → Desired.

High cohesion module focuses on a single feature.

Types of Cohesion-

(best) Functional → each element does some related task.

Sequential → O/P of one opern is I/P to another. In same module

Communicational ← Elements that operate on same I/P data or contribute to same O/P data. (communicational convenience)

Procedural → instructions do diff tasks, but to ensure a particular order, put into same module.

↳ Poor maintenance

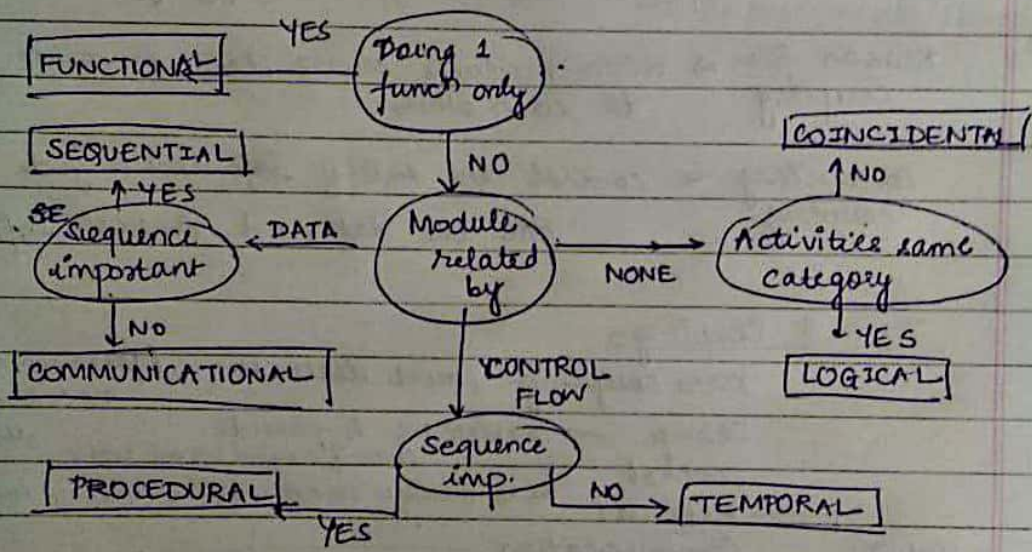
यही सच्चा साहसी है जो कभी निराश नहीं होता

Scanned with CamScanner

Temporal → Instructns that must be executed during same time span / phase / stage.

Logical → When modules have logically similar instructions / elements.

Coincidental → Instructns are not related.



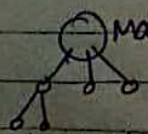
FUNCTION ORIENTED DESIGN

After Requirement phase, we've design phase.
↓
what'll be the modules, their functionalities & how'll they be connected

Funct^n oriented design is decomposed into a set of interacting units where each unit has a clearly defined function.

How? ① Start with high level description of what the S/W program does.

② Refine each part in detail one by one. } TOP DOWN

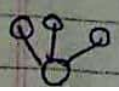


Main Module, has major 3 funct^nality

↳ Needs more modules to achieve the functionity

↳ Issue
Each module used by almost 1 other module (parent)

Sol^n Design reusable modules.



Scanned with CamScanner

functional testing

FUNCTIONAL TESTING

→ Test the functionality of the program.
→ Observation of O/P for certain I/P. No code Analysis
→ BLACK BOX Testing

BAV

* Boundary Value Analysis — boundary value analysis

bcz I/P values near boundary have higher chances of error

For each variable,
test cases developed are-

- Min value
- Just above min.
- Nominal
- Max
- Just below max.

single fault assumption

* Single fault assumption- Only one variable can assume fault / boundary at a time

For n variables → (4n+1) test cases.

roburstness testing

* Robustness Testing (stronger / robust version of BVA)
extended version of BVA : BVA + just above + just below
max min

For n variables → (6n+1) test cases

worst case testing

* Worst Case Testing (Extension of BVA)
Single fault assumption is not ~~taken~~ considered.

For n variables → 5^n test cases.

equivalence class testing

* Equivalence Class Testing

I/p & O/p domain is partitioned into mutually exclusive parts.

Any one sample from a class is representative of the entire class.

Steps to make test case-

1. Identify eq. classes based on I/P & O/P conditions & partitioning into valid & invalid classes
2. Generate test cases using eq classes

Ex. $1 \le x \le 100$

Valid → Invalid

वही सच्चा साहसी है जो कभी निराश नहीं होता।

विकार उत्पन्न करें ऐसे वस्त्रों और शृंगार से बचो।

Scanned with CamScanner

* DECISION TABLE BASED TESTING

decision table based testing

Useful when a variety of actions exist & we need to take a particular action depending upon existing condⁿ.

Decision table consist of -

- → condition stub
- → Action Stub
- → Condition Entries
- → Action Entries

condition stub

If entries ⇒ binary
↳ Limited entry decision table

action stub

CONDITION STUB		ENTRIES							
	C1	TRUE				FALSE			
	C2	T		F		T		F	
	C3	T	F	T	F	T	F	T	F
ACTION STUB	A1	X	Y						
	A2	X							
	A3								
	A4	X these action hold							

* CAUSE EFFECT BOUNDARY TESTING

cause effect boundary testing

Technique to systematically select a high yield set of test cases.

विकार उत्पन्न करें ऐसे वस्त्रों और शृंगार से बचो।

विकार उत्पन्न करें ऐसे वस्त्रों और शृंगार से बचो।

Scanned with CamScanner

software maintainence

maintainence

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Software Maintenance

Software Maintenance includes :
- error correction
- enhancement of capabilities
- deletion of obsolete capabilities
- Optimization.

Any work done to change the SW after it is in operation is considered to be maintenance work.

Purpose : It preserves the value of the SW over time.

4 Categories of Maintenance

(1) CORRECTIVE MAINTENANCE corrective

↳ Initiated as a result of defects in software any kind of coding, design, logic error.

→ Appropriate actions are taken to restore the correct functionality of the SW system.

patching

Patching : Emergency fixes (mainly due to pressure from management)

↳ gives rise to unforeseen future-errors due to lack of proper impact analysis.

(2) ADAPTIVE MAINTENANCE adaptive

↳ Modifying the software to match changes in ever-changing environment

↓

any outside factors : platform changes, govt policies, business rules.

(3) PERFECTIVE MAINTENANCE perfective

↳ Improving the processing efficiency OR performance OR restructuring the SW to improve changeability.

- Expansion in Requirements ⟹ Enhancing existing system functionality

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

- Better, faster, cleaner.

preventive

(4) PREVENTIVE MAINTENANCE

↳ Making the program easier to understand
↳ helps us in future maintenance work.

eg: optimization, documentation updation.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

reliability

RELIABILITY : Introduction

Reliability of a Software specifies probability of failure free operation for a given time duration.

* It is a dynamic system characteristic
  - presence of faults → a function of number of SW failures.
  - it is an execution event where the SW behaves unexpectedly.

✓ Not all SW faults have equal probability of manifestation / execution.
  - ↳ Removing SW faults from those SW parts which are rarely used makes little improvement in Reliability.

Operational Profile of SW : the way in which SW is used by the users
- The kind of I/P that are supplied to the SW.

Failure of a SW depends on 2 factors :
1. No. of faults being evaluated in SW.
2. Operational profile of execution of SW.

Types of Time

1. Execution Time : The actual CPU time that the SW takes during its execution
2. Calendar Time → Normal/Regular time that we use on a daily basis.
3. Clock Time : Actual time that is elapsed while the SW is executing
   ↓
   including the SW time that the SW spends while waiting in system.

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

reliability metrics

Reliability Metrics

1. Probability of failure on demand

- It is a measure of the likelihood that the system will behave in an unexpected way when some demand is made on it

eg: Safety-critical system.

rocof

2. Rate of Occurrence of Failure (ROCOF)

A measure of frequency of occurrence with which unexpected behaviour is likely to be observed.

eg ROCOF = $\frac{9}{100}$ → SW will fail 9 times out of 100 operational unit times.

3. Mean Time To Failure (MTTF)

mttf

- A measure of time interval between observed failures.
- Useful when system is stable and no changes are being made to it.
  ↳ Indication of how long the system will be operational before failure occurrence.

4. Availability : Measure of how likely the system is to be available for use.

Availability = $\frac{10}{100}$

formula

MTBF = MTTF + MTTR

- MTBF: Mean Time Between Failure
- MTTF: Mean Time To Failure.
- MTTR: Mean Time To Repair

$$\text{Availability} = \left(\frac{MTTF}{MTTF + MTTR}\right) \times 100\%$$

Scanned with CamScanner

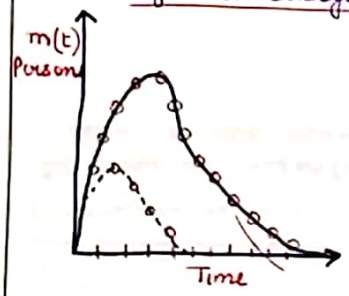
not in syllabus

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

putnam resource allocation

The Putnam Resource Allocation Model

- Based on Rayleigh curve for approximating Hardware development projects
  - → Norden of IBM.
- Putnam observed that Rayleigh curve was a close approximation for SW project and Software subsystem development.



Rayleigh Manpower Loading Curve.

—— Overall curve
---- Design & Coding

It measures Manpower in terms of person per unit time as a function of time
→ Person-year/year

The Rayleigh curve is given by differential equation

$$m(t) = \frac{dy}{dt} = 2Kat\,e^{-at^2} \quad —(1)$$

dy/dt : Manpower utilization rate per unit time
t : elapsed time
a : parameter affecting shape of curve
K : area under curve in interval $[0,\infty]$

Integrating (1) on interval $[0,t]$ we get

$$y(t) = K\left[1 - e^{-at^2}\right] \quad —(2)$$

↓
Cumulative Manpower used upto time t.

$y(0) = 0$ $\quad$ $y(\infty) = K$

The cumulative manpower is null at the start of the project & grows monotonically towards the total effort K

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Importance of 'a'

Dimension of a : $1/time^2$

It plays an important role in determination of peak manpower.

↳ Larger the value of 'a', earlier will be the occurrence of peak & steeper is the person-profile.

$\frac{d^2y}{dt^2} = 2Kae^{-at^2}[1-2at^2] = 0 \Rightarrow \boxed{t_d^2 = \frac{1}{2a}}$

t_d : time where maximum effort rate occurs

t_d represents the total project development time.

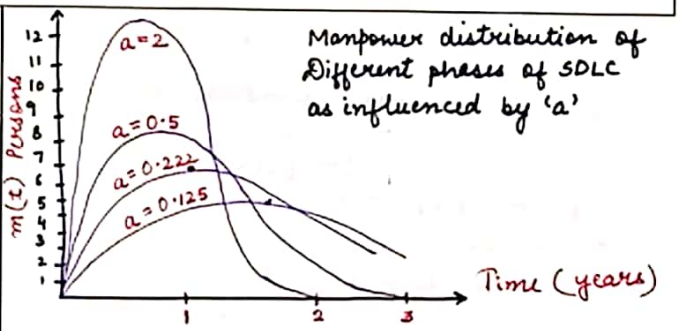
$\boxed{a = \frac{1}{t_d^2}}$

Put t_d inplace of t in (2)

Estimate of develop. time

$E = y(t) = K\left(1 - e^{-\frac{t_d^2}{2t_d^2}}\right)$

$= K(1 - e^{-0.5}) = 0.3935K$



Manpower distribution of Different phases of SDLC as influenced by 'a'

→ Put this value in ① eqn.

$m(t) = \frac{2K}{2t_d^2} te^{-\frac{t^2}{2t_d^2}} = \boxed{\frac{K}{t_d^2} te^{-\frac{t^2}{2t_d^2}}}$

people involved in project at the peak time.

At time $t = t_d$, peak manpower is ⇒ $m(t_d) = \boxed{m_0 = \frac{K}{t_d\sqrt{e}}}$ (total proj cost/effort)

Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

Putnam Resource Allocation Model

Q A software project is planned to cost 95 PY in a period of 1 year 9 months. Calculate the peak manning and average rate of SW team build up.

Software Project cost = K = 95 PY.

Peak development time = $1 + \frac{9}{12} = \underbrace{1.75 \text{ yr.}}_{t_d}$

Peak Manning: $m_0 = \frac{K}{t_d \sqrt{e}}$

$= \frac{95}{1.75 \times 1.648} \approx 33$ persons

Avg rate of SW team build-up $= \frac{m_0}{t_d} = \frac{33}{1.75} \approx 18.8$ person/year

Q Consider a large-scale project for which manpower requirement is K = 600 PY and development time is 3 year 6 months

a) Calculate the peak manning & peak time

b) what is the manpower cost after 1yr 2m?

a) $t_d = 3 + \frac{6}{12}$ yr = 3.5 year.

$m_0 = \frac{K}{t_d \sqrt{e}} = \frac{600}{3.5 \times 1.648} \approx 104$ persons.

(b) $y(t) = K\left[1 - e^{-at^2}\right]$

$t = 1 + \frac{2}{12}$ yr = 1.17 year.

$a = \frac{1}{2t_d^2} = \frac{1}{2 \times (3.5)^2} = 0.041$

$y(1.17) = 600\left[1 - e^{-(0.041 \times 1.17 \times 1.17)}\right]$ (e to the power)

32.6 PY

Scanned with CamScanner

reverse engineering

Software Engineering

Easy Engineering Classes – Free YouTube Lectures

EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

REVERSE ENGINEERING

It is the process followed so as to find unknown and difficult information about a software.

Why Is It Important ?

It is important in case when software lacks proper documentation, is highly unstructured or its structure has degraded through maintenance works.

Purpose: Recovering information from the existing code/any intermediate documents
↳ program understanding at any level is included in R.E.

Uses of Reverse Engineering

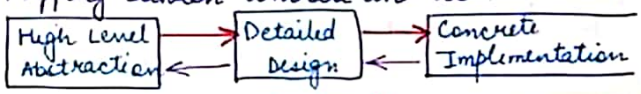
- ▲ Program Understanding
- ▼ Re-documentation or document generation
- ▲ Recovery of design details
- ▼ Identify Reusable components.
- ▲ Business Rules implied on SW.
- ▼ Understanding high level system description
- ▲ Identify components that require restructuring.

What is not in scope of Reverse Engineering ?

* Re-design * Re-structuring
* Enhancement of system functionality.

Major Tasks

1. Mapping the program to the (problem it represents in) application domain.
2. Mapping between concrete and abstract levels



3. Rediscover high level structures
↳ Understanding implementation details.

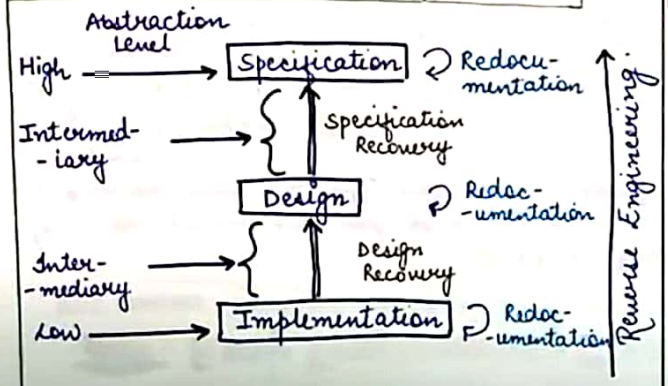
Scanned with CamScanner

Easy Engineering Classes – Free YouTube Lectures
EEC Classes For Engineering Students of GGSIPU, UPTU and Other Universities, Colleges of India EEC Classes

4. Find missing links between program syntax and semantics.

5. Extract Re-usable component
   ↳ to find the number of components that can be reused from this code.

LEVELS OF REVERSE ENGINEERING



REDOCUMENTATION

It is the recreation of a representation of a system which is equivalent to the existing system and at the same level of abstraction

Goals: To create alternate views of system to improve understanding.

② Improve existing documentation
③ Create documentation for a program that has been recently modified.

DESIGN RECOVERY

Identify and extract higher-level abstractions.
↳

The obtained design is used as a basis for future system modification or development of similar application.

reverse engineering

Scanned with CamScanner

er diagram

Entity-Relationship Diagrams

ER-modeling is a data modeling method used in software engineering to produce a conceptual data model of an information system. Diagrams created using this ER-modeling method are called Entity-Relationship Diagrams or ER diagrams or ERDs.

Purpose of ERD

- The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD.
- The ERD serves as a documentation tool.
- Finally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.

Components:

components

Entity, attribute, relationships

Entity:

entity

- An entity can be a real-world object, either animate or inanimate, that can be merely identifiable.
- An entity is denoted as a rectangle in an ER diagram.

Attributes

atrributes

- Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.
- Represented by circle

Scanned with CamScanner

There are four types of Attributes:

1. Key attribute
2. Composite attribute
3. Single-valued attribute
4. Multi-valued attribute
5. Derived attribute

key

composite

single valued

multivalued

derived

Relationships:

relationships

i_love_amy

- The association among entities is known as relationship. Relationships are represented by the diamond-shaped box. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.
- By diamond

Degree of relationship: no. of participants, unary, binary, ternary

cardinality

Cardinality: no. of entities in 1 set which can be associated with those of another set.

- One to one
- Many to one
- One to many
- Many to many

Scanned with CamScanner

State diagram: state diagram

A state diagram is used to represent the condition of the system or part of the system at finite instances of time.

It's a **behavioral** diagram and it represents the behavior using finite state transitions.

State diagrams are also referred to as **State machines** and **State-chart Diagrams**.

Uses of statechart diagram –

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behavior of the system .
- To understand the reaction of objects/classes to internal or external stimuli.

Two types state diagram

- **Structure Diagrams –** Used to model the static structure of a system, for example- class diagram, package diagram, object diagram, deployment diagram etc.
- **Behavior diagram –** Used to model the dynamic change in the system over time. They are used to model and construct the functionality of a system. So, a behavior diagram simply guides us through the functionality of the system using Use case diagrams, Interaction diagrams, Activity diagrams and State diagrams.

Scanned with CamScanner

state diagram state chart

Basic components of a statechart diagram –

- **Initial state –** We use a black filled circle represent the initial state of a System or a class.
- **Transition –** We use a solid arrow to represent the transition or change of control from one state to another. The arrow is labelled with the event which causes the change in state.
- **State –** We use a rounded rectangle to represent a state. A state represents the conditions or circumstances of an object of a class at an instant of time.
- **Fork –** We use a rounded solid rectangular bar to represent a Fork notation with incoming arrow from the parent state and outgoing arrows towards the newly created states. We use the fork notation to represent a state splitting into two or more concurrent states. fork
- **Join –** We use a rounded solid rectangular bar to represent a Join notation with incoming arrows from the joining states and outgoing arrow towards the common goal state. We use the join notation when two or more states concurrently converge into one on the occurrence of an event or events. join
- **Self transition –** We use a solid arrow pointing back to the state itself to represent a self transition. There might be scenarios when the state of the object does not change upon the occurrence of an event.
- **Composite state –** We use a rounded rectangle to represent a composite state also.We represent a state with internal activities using a composite state.
- **Final state –** We use a filled circle within a circle notation to represent the final state in a state machine diagram.

Scanned with CamScanner

Project scheduling

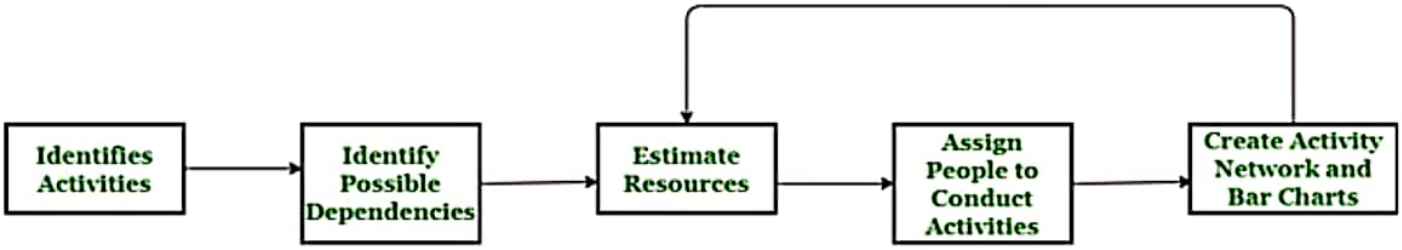
project scheduling

Scheduling in project management means to list out activities, deliverables, and milestones within a project that are delivered. It contains more notes than your average weekly planner notes. The most common and important form of project schedule is Gantt chart.

Advantages of Project Scheduling :

There are several advantages provided by project schedule in our project management:

- It simply ensures that everyone remains on same page as far as tasks get completed, dependencies, and deadlines.
- It helps in identifying issues early and concerns such as lack or unavailability of resources.
- It also helps to identify relationships and to monitor process.
- It provides effective budget management and risk mitigation.



Project Scheduling Process

Scanned with CamScanner

critical path

Critical path

In project management, a critical path is the sequence of dependent tasks that form the longest duration, allowing you to determine the most efficient timeline possible to complete a project.

Software Configuration Management(SCM)

SCM is the discipline which

scm

software configuration management

- Identify change
- Monitor and control change
- Ensure the proper implementation of change made to the item.
- Auditing and reporting on the change made.

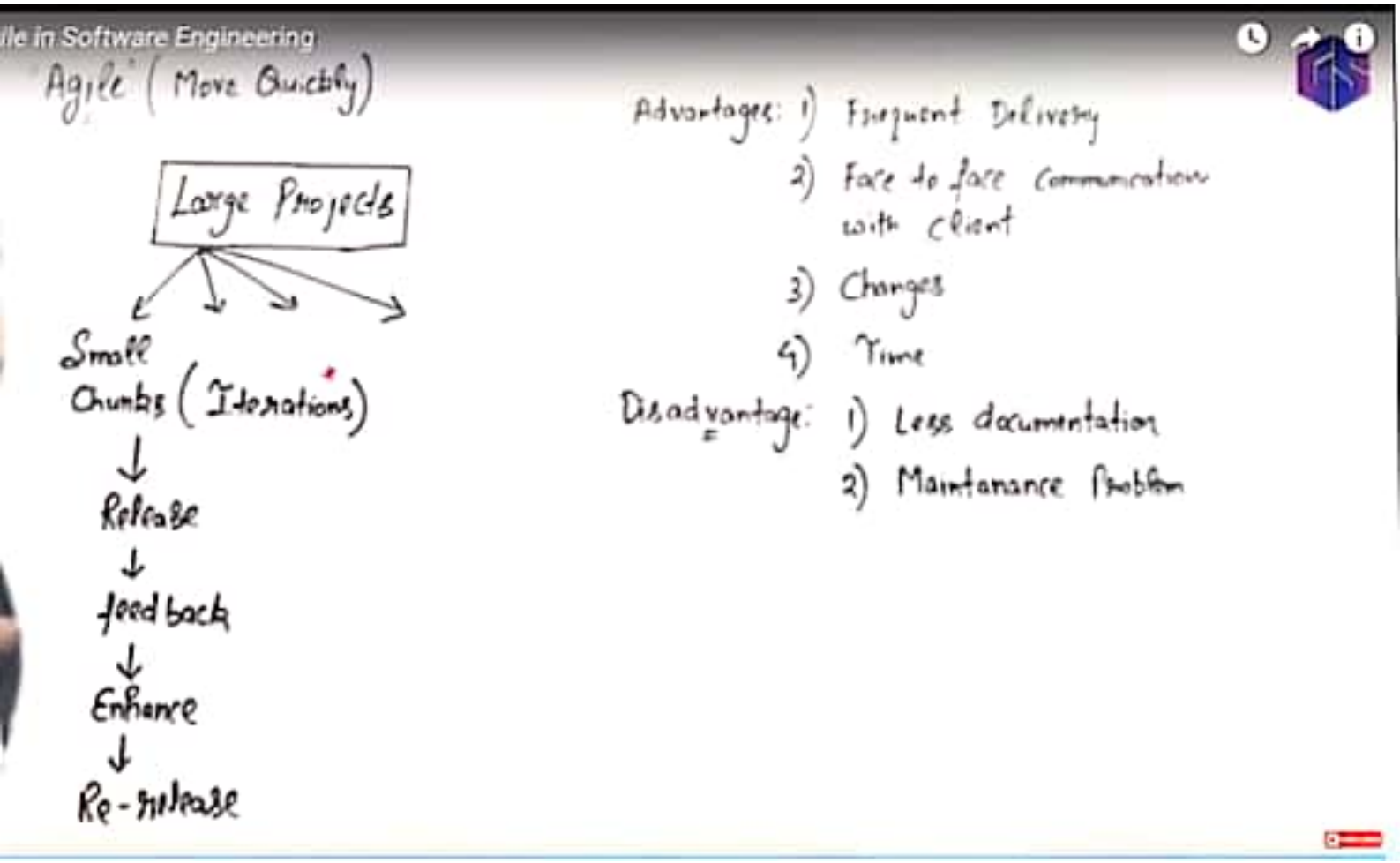
Configuration Management (CM) is a technic of identifying, organizing, and controlling modification to software being built by a programming team.

The objective is to maximize productivity by minimizing mistakes (errors).

It provides the tool to ensure that changes are being properly implemented.

Scanned with CamScanner

agile model

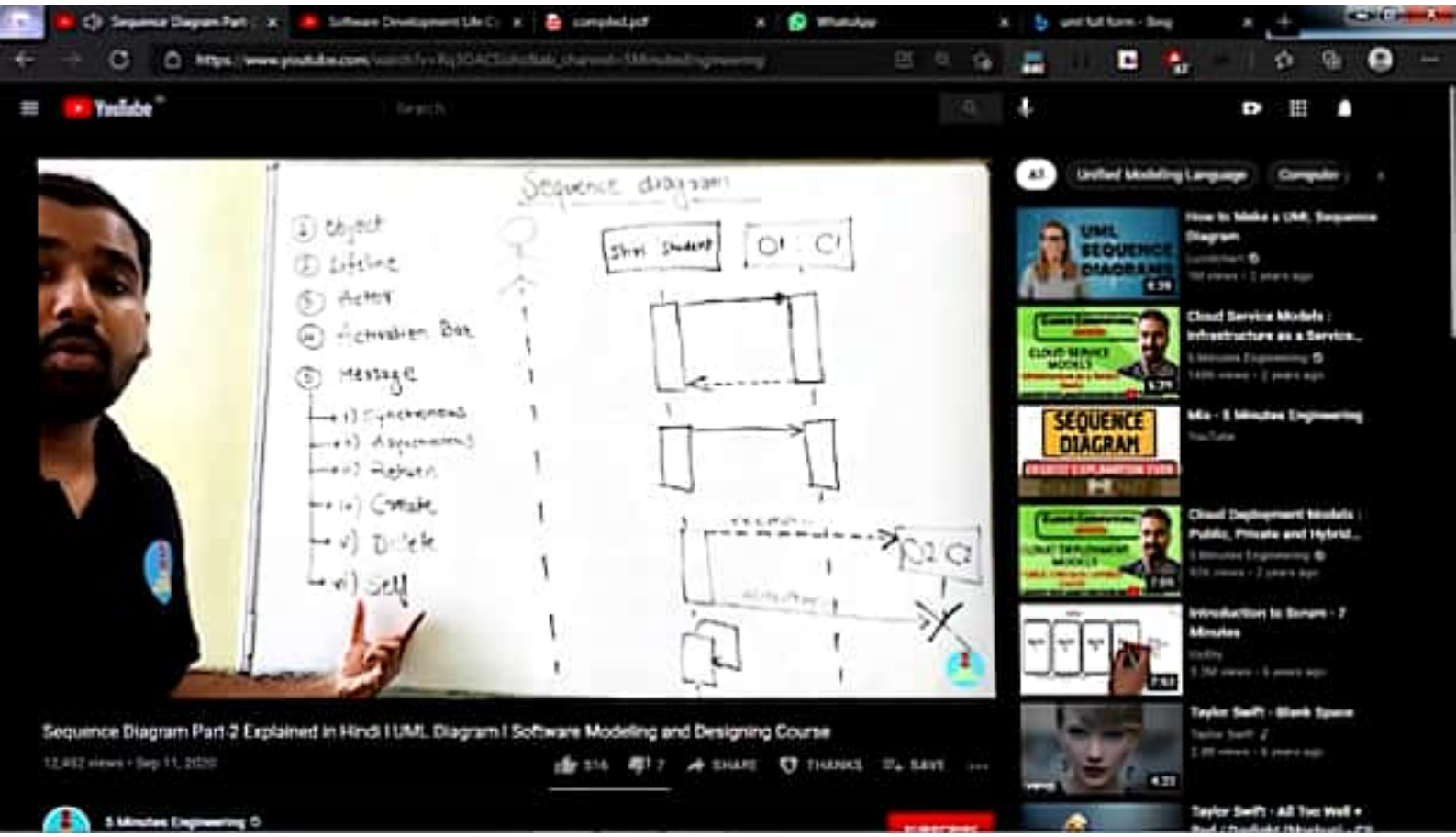


Scanned with CamScanner

object_lifeline_actor_activation-bar_message

sync_async_returns_create_delete_self

sequence diagram



Scanned with CamScanner