06 05 18

Project Developer

↓

Client

Ref Book → System Analysis And Design

– V Rajaraman

* Information for Management

Operational. tactical. strategic. Statutory.

— x —

Session 1

07 05 18

Roles of Software Engineering:

→ Project manager

→ Team leader

→ Senior developer

→ Junior developers

Based on working procedure:

→ Project manager

→ Developers (back end)

↓

C#, Java, Database

→ Front end developers

→ Testing / Quality Assurance Team (QA) Team

Design pattern ⟶ Java

↓

Code project

Project Proposal

Project Title

Objective

Features

Team Member with roles, name, ID

1. PM
2. TL
3. Software manager
4. Developer

* next lab Print

hasan07cse@gmail.com ⟶ Pdf

Sub: Software Project Proposal

Mail body:

Team member's name

" " ID

——×——

09.05.18

* Management Hierarchy & information needs:

Organogram ⟶ Hierarchy.

* Functional Areas
* Quality of information
* Business Data Processing

OLTP (On-line Transaction Processing)

OLAP (On-line Analitical Process)

Batch processing

OLTP Vs Batch.

* Decision Support System.

* Big Data: (VVV)

→ Vast

→ Variety (Video, audio, image, file → Unstructured)

→ Velocity

—x—

13.05.18

* System Analysis And Design Life cycle:

Operation → day to day record

tactical → Monthly record

Strategic → Overall yearly record

Statutory → Govt. rules & regulation detecting.

* Steps involved in Analysis and Design:

1. Requirements Determinations
2. " Specifications
3. Feasibility Analysis

4. Final specification
5. Hardware study
6. System design
7. " Implementation
8. " Evaluation
9. " Modification.

——×——

"Sessional"

14.05.18

* Definition of Software Process:

→ A framework for the activities, actions, and tasks that are required to build high-quality software.

→ Software Process (SP) defines the approach that is taken as software is engineered.

* A Generic Process Model:

Software process

Process framework

ivities

activity #1

activity #n.

API → Application Program Interface.

```
Findresult (DIN, 1101, 11, 2010);
  {

  }
```

Stake holders

* Process Flow:

① Communication → ② Planning → ③ Modeling → ④ Construction → ⑤ Deployment

a). linear process flow

b). iterative " "

c). Evolutionary " "

d) Parallel " "

* The waterfall Model:

Print: Next Day:

Requirement Gathering

1. Requirement list (Description)
2. Feasibility Analysis ↙

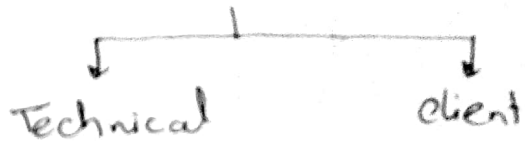
Not feasible (1-2 points)

16.05.18

System Life Cycle Diagram:

Babysedtting ~~Babysitting~~ period.

↳ System evaluation



Role of Systems Analyst:

Attributes of a systems analyst:

Knowledge of organization. [Jargon]

Knowledge of computers and software.

Good Interpersonnal Relations

Ability to communicate.

Analytical mind

Breadth of knowledge.

Tools used by systems Analyst

[Database Normalizing]

20.05.18

Tools used by systems Analyst:

ISO → International Standard organization.

CMM → Capacitive maturity Model.

↓

5 levels

CMMI 1

CMMI 5

Assignment:

ISO
CMMI } → next class

Title → Assignment on ISO and CMMI A.

Feasibility Analysis:

a) Technical Feasibility

b) Operational "

c) Economic "

CBA → Cost Benefit Analysis

SRS → System Requirement Specification.

Steps in Feasibility Analysis:

Guidelines for searching goals:

——×——

"Sessional"

21.05.18.

Information Gathering:

① Proposal

② Requirement Determination and feasibility analysis.

③ Information gathering /collection. (next day report).

Stakeholder → user, developers, those whose features we use, client.

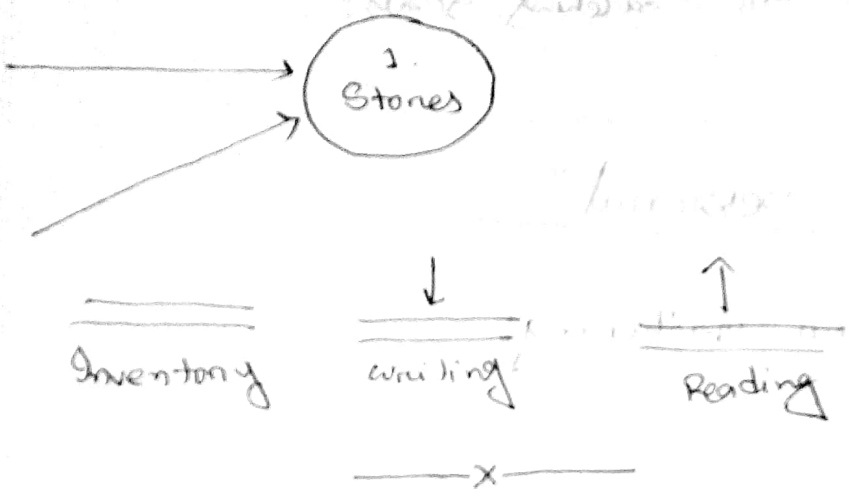
* Information gathering strategies:

List of collected informations. → for each requirement.

Information Sources.

Document Flow Diagram (DFD): (next day report)

Process



23.05.18

Resume → Summary

CV → Elaboration

Academic CV

Professional CV

Case-Study - hostel Information System:

Examine alternative solutions

(Solution B :

Solution C :

"Special class" 26.05.18

Ref. Book → Software Engineering - A practitioner's Approach - 6th Edition

- Roger S. Pressman

Software :→ Instructions
Data structure
Documents / Documentation

MS-DOS ⟶ OS (640KB)
↳ 1988
↓
2018

PC → IBM
↓
Now → Clone

GUI → Bill Gates

Windows → 1995
98
2000
XP

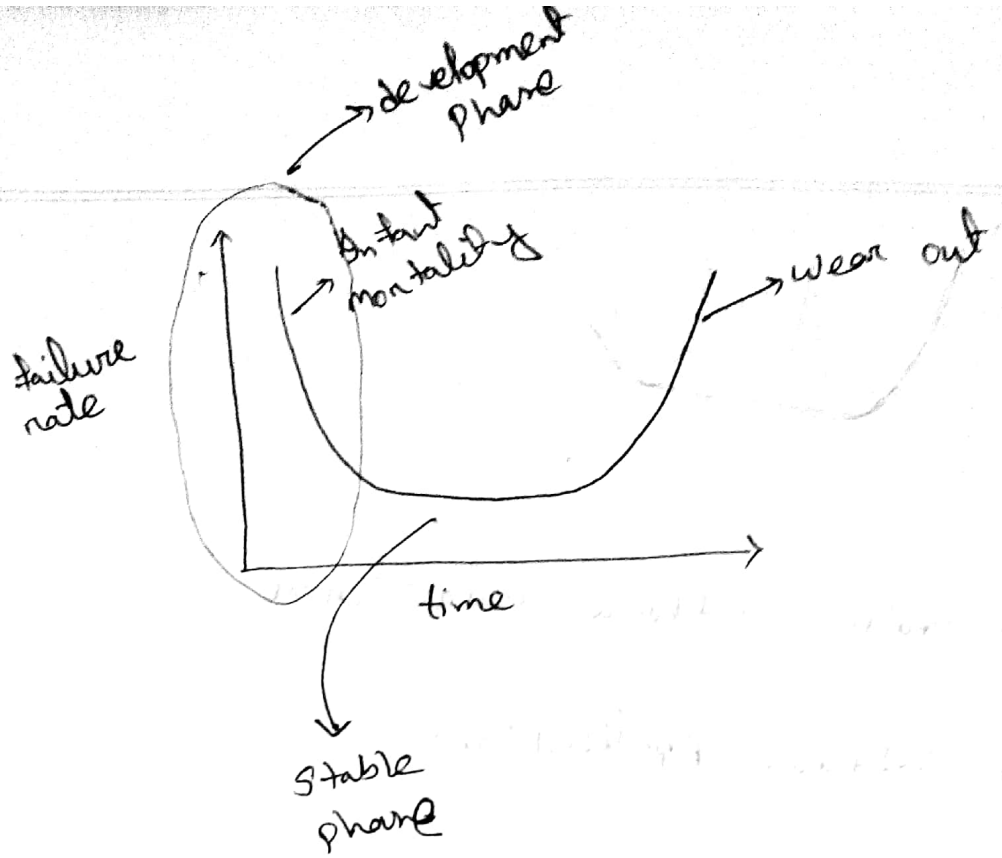


Fig: failure curve for Hardware.

Fig: failure curve for software.

(Idealized failure curve for software)

* In theory, software does not wear out. But
  1. Hardware upgrades
  2. Software upgrades.

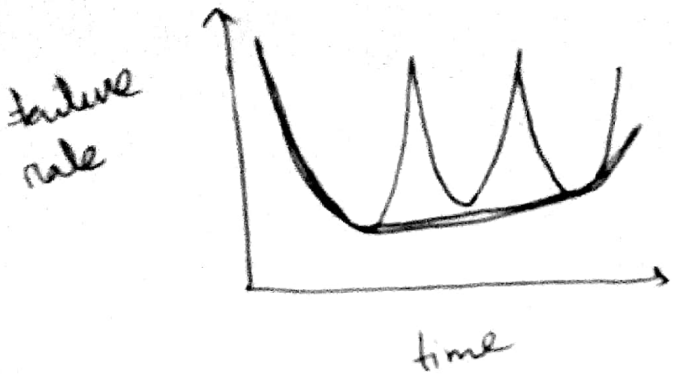


Fig: Realistic software failure curve

* Types of software Applications:
  1. System software
  2. Application "
  3. Engineering / Scientific "
  4. Embedded "
  5. Product-line "
  6. web-applications "
  7. AI "

* Ubiquitous Computing.
* Netsourcing.
* Open-source
* The new economy

* ~~Outsource~~ Outsourcing: → Next class

Legacy Software: (older programs)

The software must be adapted

must be enhanced

" " extended

re-architected

" "

Software Myths (Managerial myths)

(practitioner's myths)

* Software Products:

Generic Products

Customized products.

Software Costs:

Cost-effective software development.

↳ reliable

↳ trust worthy

↳ easy user interface

Project Management Concepts:

* Spectrum of Management Concepts:

  people → grow, motivate, deploy, retain.

  The product

  " process

  " project

PM-CMM

* Players:

  senior managers

  Project "

  Practitioners

  Customers

  End users.

* Technical leadership model: (Jerry Weinberg)

  Motivation

  Organization

  Ideas and Innovation.

  problem Solving

Managerial identity
Achievement
Influence and team building
Problem solving

} Project managers characteristics.

* Seven project factors impacting SW team:

Difficulty
Expected size
The time
The degree
" required quality & reliability of the system.
" rigidity
" degree of communication.

The Software Team:

marilyn manter model:

Democratic Decentralized (DD)
Controlled Decentralized (CD) [Conflicting term]
Controlled Centralized (CC)

CD Structure : Chief Programmers team (eaiest structure).

* Senior Engineer

Specialists
Support staff
Software librarian

* Orgnizational paradigm of sw team:

closed paradigm
Random "
Open "
Synchronous "

Co-achieveness → one team
Coupling → Different team.

* Project Coordination Techniques:

Formal, impersonal approaches
Formal, interpersonal procedures.

Informal, interpersonal procedures.

Electronic communication

Interpersonal network.

——X——

27.05.18

First browsers:

Netscape

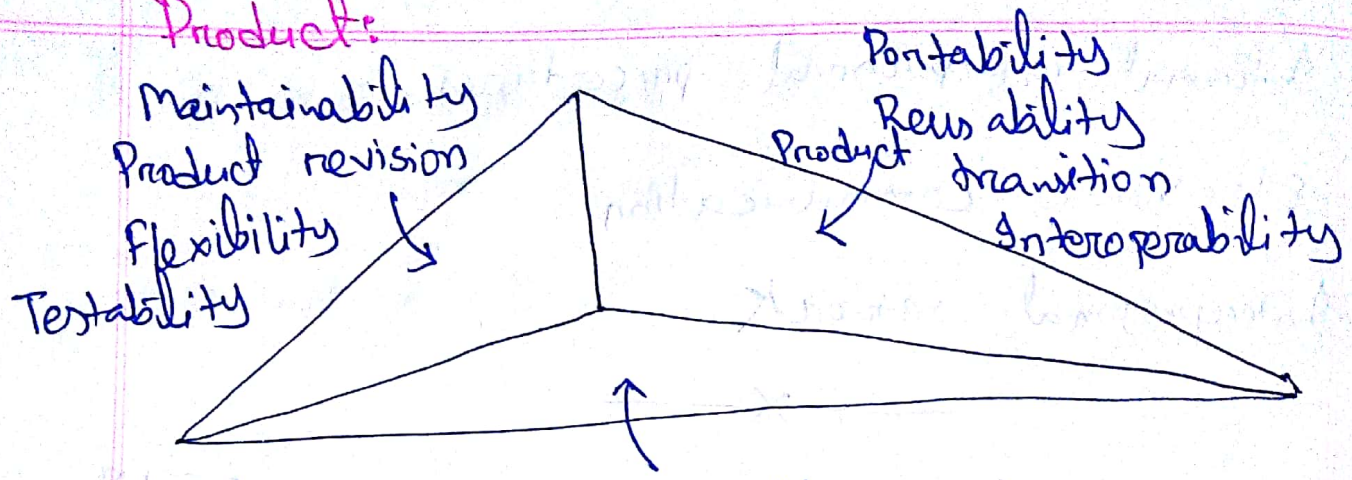
Communicator

explorer.

The Project:

* Understand what can go wrong
* Start on the right foot.
* Maintain momentum
* Make smart decisions.
* Conduct a potmortem analysis

Product:



Correctness, Integrity, usability, efficiency, Reliability.

McCall's triangle of Quality (1970's)

* Measure
  Metrices
  Software metrices
  Indicators.

Measure, Metrics :

Measurement principles:

Measurement Process:

Formulation

Collection

Analysis

Interpretation

Feedback

Effective Metrics Attributes:

simple and computable

Empirically and intuitively persuasive

Consistent and objective

Consistent

Programming language

Collection and Analysis Principles:

* In the Process and Project Domains

Process Indicators

Project Indicators

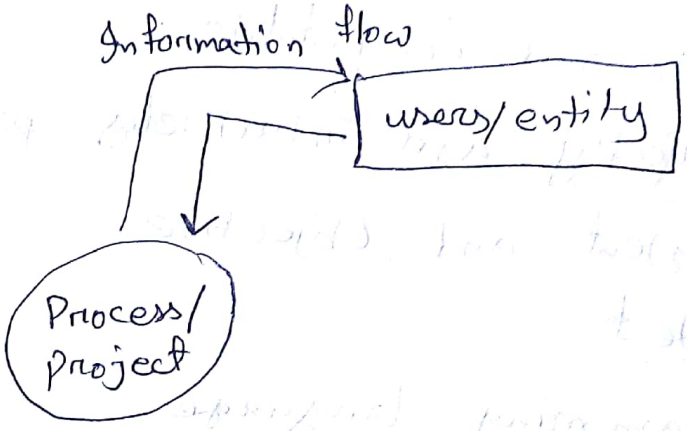
* Process involves steps of a program execution.
* Project involves scope, defined time, schedule, combination of process etc.
* Software metrics → useability.

—x—

"Sessional"

28.05.18

Title: Information gathering ~~approach~~. Strategies and Identify Sources DFD.



—x—

30.05.18.

Hostel Information System:

* Alternative solutions:

A: Improve manual system

B: Use PC based periodic update system

C: An on-line system with server and several clients.

CBA → Cost Benefit Analysis.

Direct Cost

Indirect Cost

Benefit:

Tangible.

Intangible.

Pay Back Period → Final (ROI)

↳ Return Of Investment

* Present Value Method:

$$\text{Present value} = \frac{x}{(1+r)^n}$$

n = no. of month

r = % per month interest

03.06.18

CT syllabus → Feasibility Analysis

lecture-3

Chapter-4:

Information Gathering:

Interviews are very important.

- Planning an interview
- Interviewing technique.
- use of Questionnaires

leadership	New technology
Management	Software modeling

System Requirements Specification:

- Ideal characteristics of SRS.

"Sessional" 04.06.18.

The RAD Model:

* Process Flow:
  - Communication
  - planning
  - Modeling
  - Construction
  - Deployment

=> The waterfall Model:

=> The Incremental Model:

=> The RAD Model:

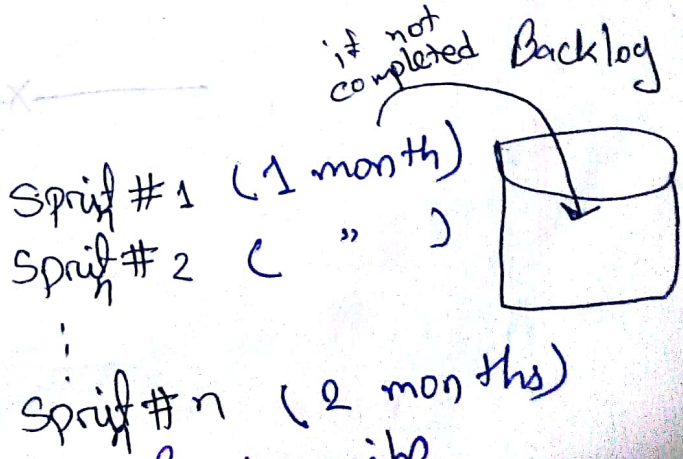
↳ Rapid Application Development.

=> Evolutionary Models: Prototyping

=> The concurred Model

~~SCRUM~~ SCRUM:

Agile



Next lab → Model choose & describe

06.06.18

Object Oriented System Modeling:

* Desirable properties of components:

Object oriented modeling

Object and their properties

object, entity

All objects have attributes

All objects have a state.

UML → Universal Modelling language.

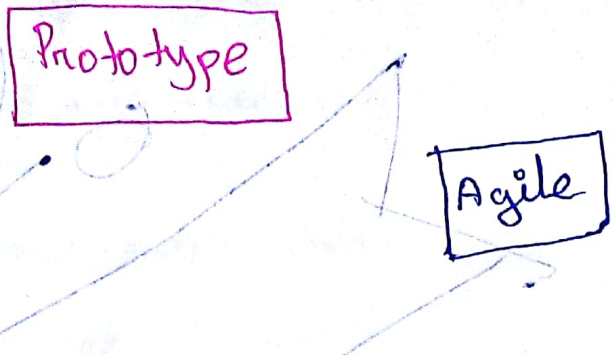
* class diagram – UML notation.

UML → Approach

Scanned by CamScanner

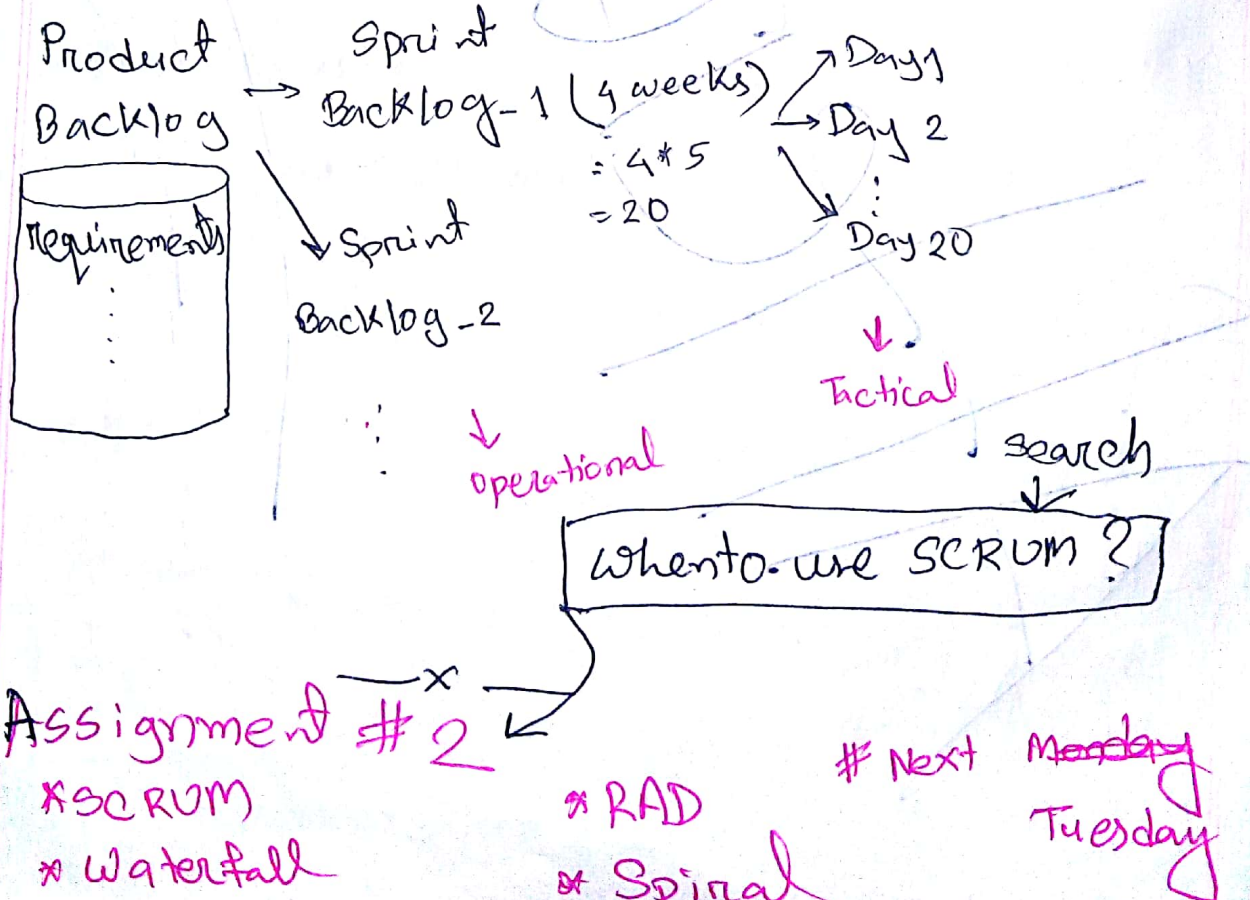
25.06.18.

* Process Flow:



Scrum → software development framework which follows agile process

Scrum:



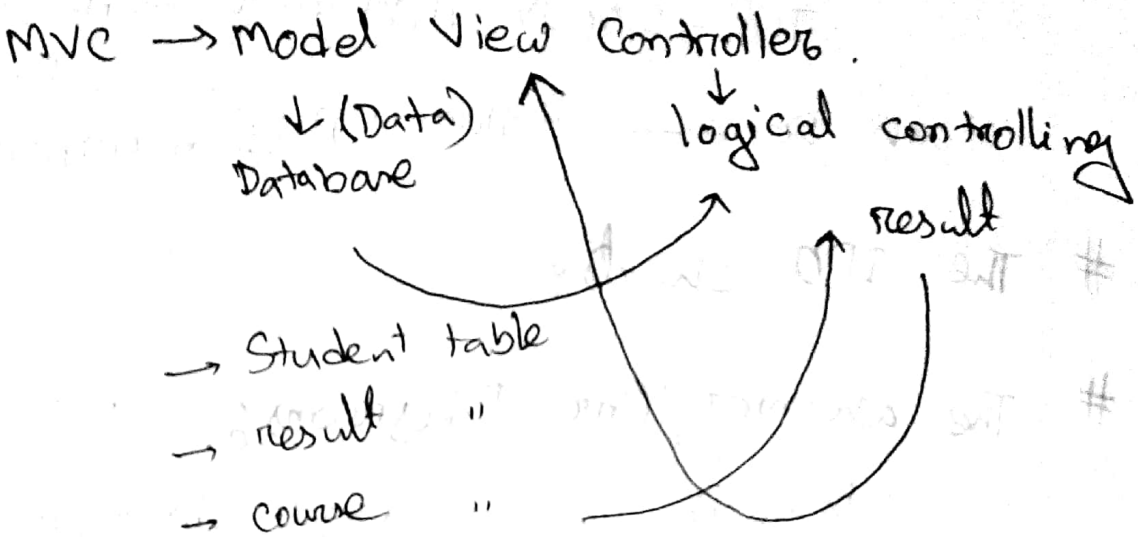
Assignment #2

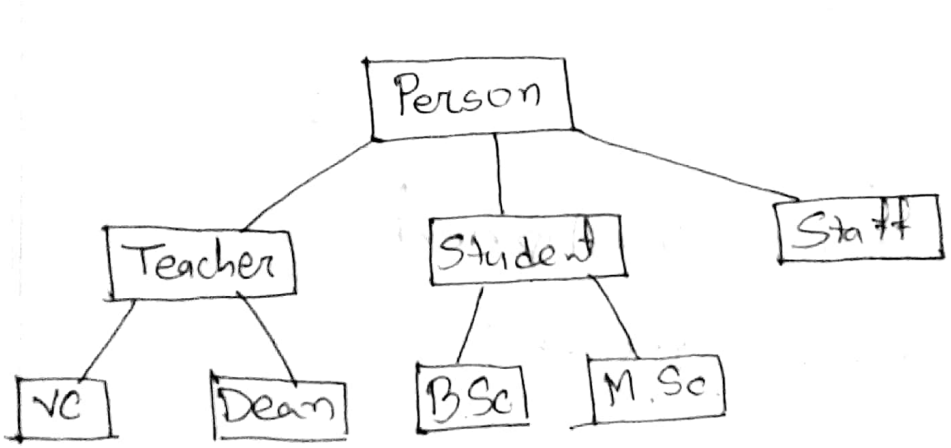
* SCRUM
* Waterfall
* RAD
* Spiral

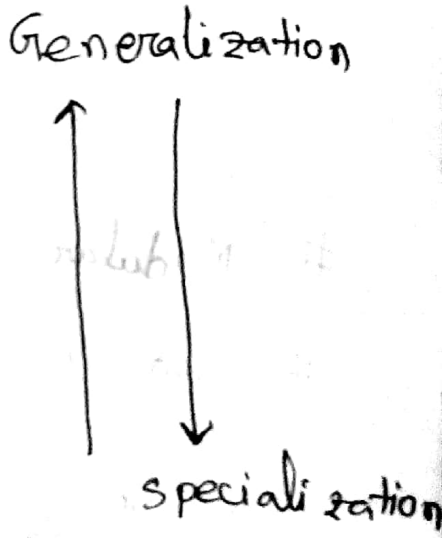
Next ~~Monday~~ Tuesday

"Lecture-06" 27.06.18.

Design Pattern:







Generalization
Specialization

Hierarchy Plus Input - Process - Output. (HIPO)

"lecture-07"

HIPO Diagram:

TOP-DOWN Approach. → Hierarchy of Tasks.

Work Done → Bottom-Up (Execution)

The IPO charts:

The warnier/Orr Diagrams:

Elements with example description.

——x——

01.07.18

Modular Software Development:

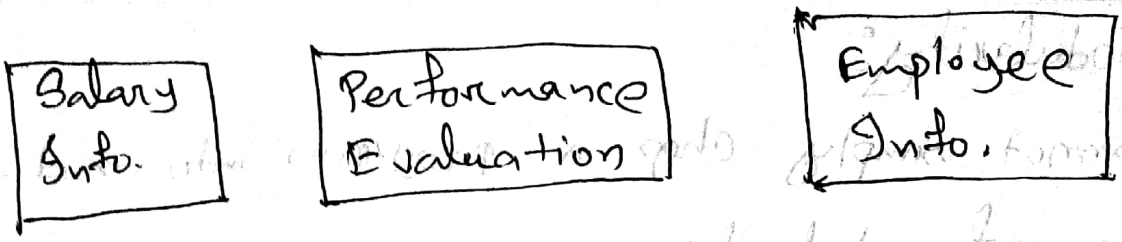
* What is modular approach?

"Project Status Presentation"

Next Lab Task:

→ Use case Diagram of your project,

→ Presentation on ⇒ "Selecting Soft. Dev. Model for your project."



CEO → All
→ Accessible (salary, performance, Employee info.)

line manager/
Team lead => (performance, Employee Info.)

Employee =>(Employee info.)

—×—

"Sessional"

02.07.18

Next lab:

→ Draw "Use case Diagram" of your project.

→ Selection of software Development Model for your project.

→ Which model?
→ Why?

04.07.18.

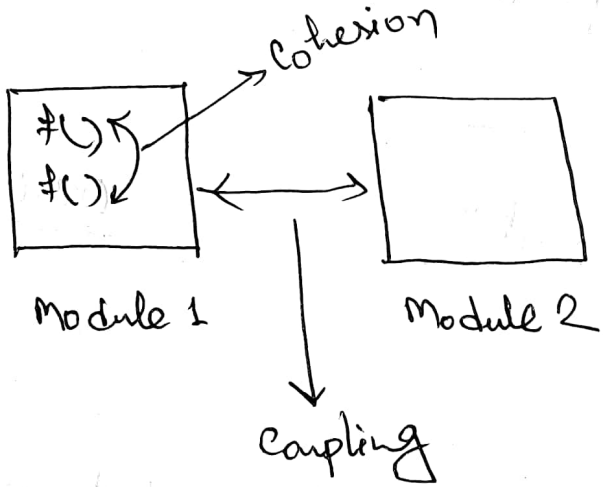
Modularity:

* Cannot simply chop a program into modules to get modularity.

* Cohesion, Coupling:

↓ maximize internal interaction

↓ minimize external interaction.



* Decreare dependency in different modules.

* Characteristics of Good Design:

- High Cohesion
- Low Coupling

* Coupling:

- Loosely coupled
- Highly coupled

i) Content Coupling
ii) Common "
iii) Control "
iv) Stamp "
v) Data "
vi) No "

lower the better

(~~not~~ ideal, but not practical)

* API

Common Coupling:

Global

→ Everyone ~~eyes~~ to read, write → Data inconsistency problem

→ Unauthorized access

→ More execution time needed

Control coupling:

*Component passes control parameters to coupled components.

—x—

"Sessional" 09.07.18

Weve choosen Waterfall Model.

Next lab: Nothing → H.W.

Tuesday → (10.07.18)

↓

Report

Case Diagram

Software Model

—x—

11.07.18.

Coupling:

Control Coupling

* Stamp Coupling:

Component passes a data structure to another component that does not have access to the entire structure.

* Data Coupling:

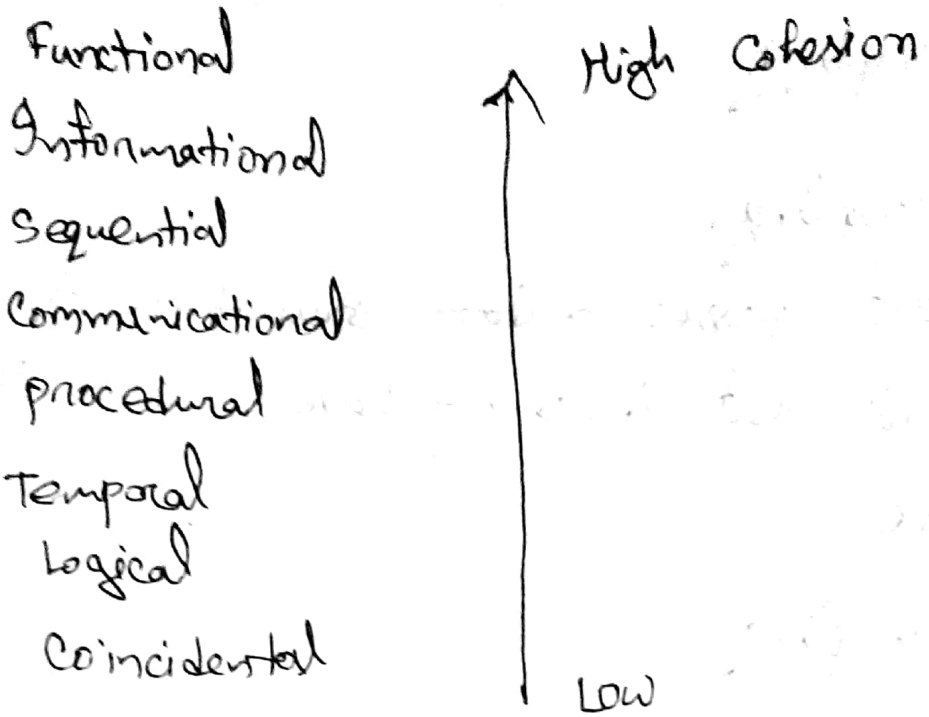
Two components are data coupled if there are homogeneous data items.

Cohesion:

The degree to which all elements of a component are directed towards a single task and all elements directed towards that task are contained in a single component.

degree of relatedness

* Range of Cohesion:



* Coincidental Cohesion:
Parts of the component performs multiple, completely unrelated actions.

* logical cohesion:
Elements of components are related logically and not functionally.

* Temporal:
Elements of a component are related by timing.

* Procedural:
Elements of a component are related only to ensure a particular order of execution.

* Communicational:
Module performs a series of actions related by a sequence of steps to be followed by the product and all actions are performed on the same data.

* Sequential:
Methods are together in a class ~~beacu~~ because the output from one part is the input to another part like an assembly line.

* Informational:
Module performs a number of actions, each with its own entry point, with independent code for each action, all performed on the same data.

* Functional:
Every essential element to a single computation is contained in the component.

"Special class" 14.07.12.

LOC

* Reconciling LOC and FP metrics:
* Object Oriented metrics:

UML → User Modeling language.

Object - Oriented metrics:

key class

Supporting class.

Use - Case Oriented metrics:

* Web Engineering Project Metrics: $C = \frac{6}{10+6}$

Customization index, $C = N_{ap} / (N_{dp} + N_{sp})$

* Metrics for software Quality:
* measuring Quality:

Mean-Time-To-change (MTTC)

Correctness → Defects per KLOC.

Maintainability

Integrity

Usability.

Integrity $= \sum [1 - (\text{threat} \times (1 - \text{security}))]$

NOP → No. of Object Point.

Defect Removal Efficiency (DRE):

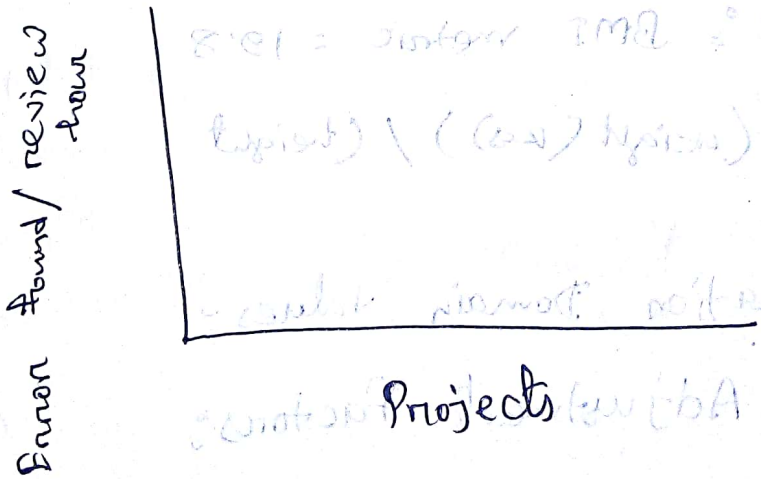
$$DRE_i = E_i / (E_i + E_{i+1})$$

E = Error
D = Defect
ith activity.

* Integrating metrics within the Software Process:

Managing Variation: Statistical Process Control:

1. The moving range control chart
2. The individual control chart.



*Moving average control chart.

mR bar → moving Range bar.

Zone Rules:

$$DRE = Echange / (Echange + Dchange)$$

Software Process and Project metrics:

* measure, Metrics, and Indicators.

measure: Height = 162 cm
weight = 51 kg

Metric: BMI metric = 19.8

(weight (kg)) / (height

* Information Domain values:

Value Adjustment Factors:

Function Point

Metrics for the Design Model:

* Hierarchical Architecture metrics:

  Structural complexity $(s(i) = f^2_{out}(i))$

  Data " $(D(i) = V(i)/[f_{out}(i)+1])$

  System " $(e(i) = s(i) + D(i))$

  Shape " (size = n + a

  Connectivity density $n = a/n$

* metrics for Object-Oriented Design:

  Size

  - population
  - volume
  - length
  - Functionality. (How many functions are delivered)

  Coupling

  Cohesion

  Primitiveness.

Specific class oriented Metrics:

Weighted methods per class (WMC)

Metrics for Source Code:

length N, $N = n_1 \log_2 n_1 + n_2 \log_2 n_2$

Volume V, =

Metrics for Maintenance:

Software Maturity index (SMI)

$SMI = [M_T - (F_a + F_c + F_d)] / M_T$

Project Scheduling:

* Basic Principles:

  Compartmentalization

  Interdependency

  Time allocation

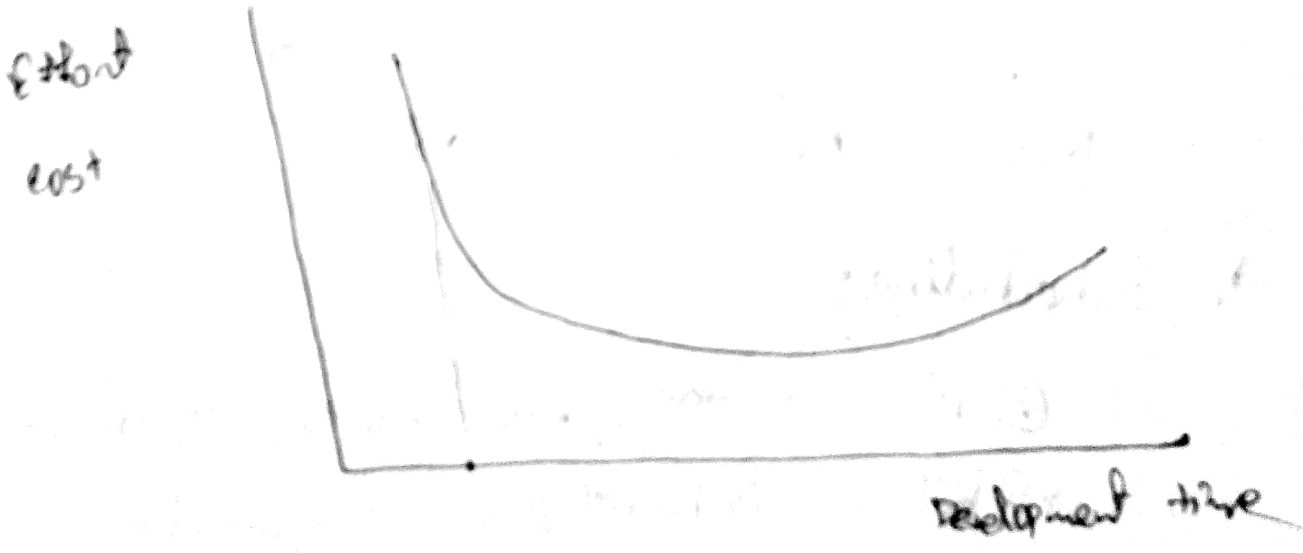
  Effort validation

  Defined responsibilities

  Defined outcomes

  Defined milestones.

* The relationship Between People & effort.

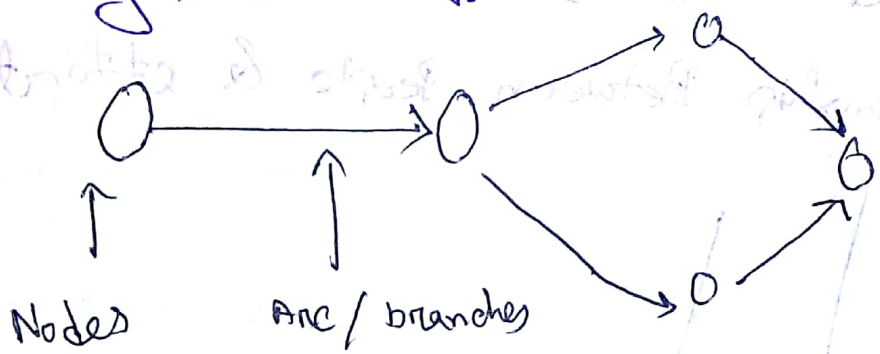


* Defining a task set for the SW Project:

Project → Uniquely defined.

1. Concept development project.
2. Application development
3. " Enhancement
4. " Maintenance
5. Reengineering projects → rebuilding.

* Activity Network:



* Scheduling:

CPM → Static/deterministic components.

PERT → Probabilistic components.

Time line chart / Gantt chart:

↓

Project Table.

Scheduling

Planning

Critical Path analysis: (Final)

As soon as the network is drawn for the given project, the time analysis is required for planning of various activities of the project. The main aim of time analysis is to prepare a planning schedule for the project.

→ Critical path analysis is a special method for time analysis to determine following:

i) Total duration for project completion

ii) Categorize the activities of the project in two types: Critical & Non-critical.

Critical Activity:

An activity in network diagram, those delay in beginning will further delay the project completion time.

Non-Critical Activity:

An activity which allows some scheduling slack, so that the start time of the activity may be delayed or advanced within some range without affecting the completion time of entire project.

* Critical Path calculations:

Notations:

E_i = Earliest occurrence time of event i

L_i = latest " " " "

(i,j) = Activity with head event j & tail event i

t_{ij} = Duration of activity (i,j)

ES_{ij} = Earliest starting time of activity (i,j)

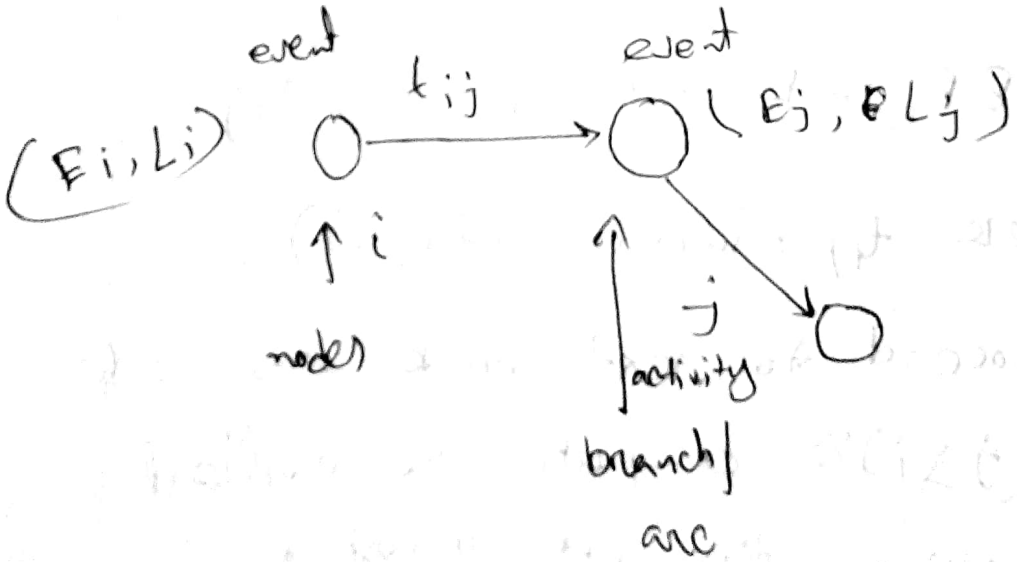
LS_{ij} = Latest starting time of activity (i, j)

EF_{ij} = Earliest finishing " " " "

LF_{ij} = Latest " " " " "

* Forward Path calculations:

Start from the initial node (i) with starting time of the project as zero. We move along the nodes with increasing orders of node numbers (serial no.) and end at final (terminal) node of the network.



* Algorith for forward Pass:

Step 1: initialize $i=1, E_1 = 0$ (First node)

" 2: Calculate earliest start time for each activity that begins at node i; as

$ES_{ij} = E_i$ for all (i,j) with starting node i.

" 3: Compute ~~earliest~~ earliest finishing time for each activity that begins at node i; as

$$EF_{ij} = ES_{ij} + t_{ij} = E_i + t_{ij}$$

where t_{ij} = duration of (i,j).

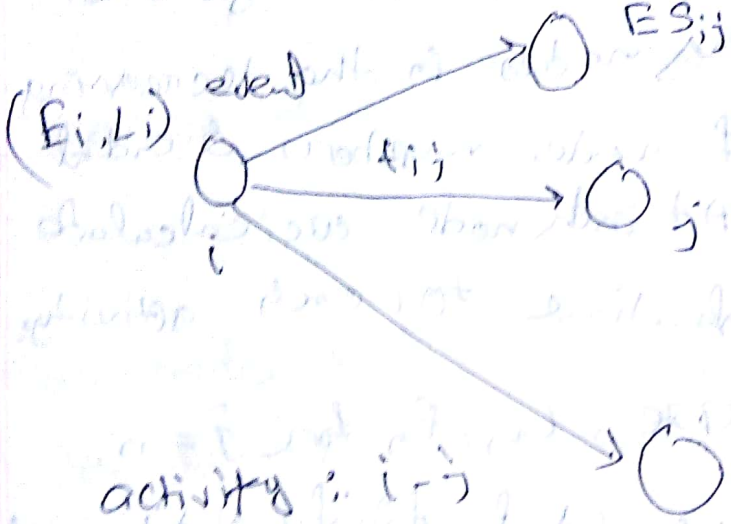
" 4: Proceed to next node, say node $j (j > i)$ & compute the earliest occurrence for node j using

$E_j = \max_i \{EF_{ij}\} = \max_i \{E_i + t_{ij}\}$

for all immediate predecessors i.

Step 5 : If $j = n$ (final node), then the earliest finish time for the project is given by,

$$E_n = \max \{EF_{ij}\} = \max \{E_{n-1} + t_{ij}\}$$



$$E_n = \max \{EF_{ij}\} = \max \{E_{n-1} + t_{ij}\}$$

$E_4 = \max\{13, 11\}$

$= 13$

Forward → max.
Backward → min.

* Backward Pass calculations:

Here, we begin from terminal (last) node of the network proceed through the network visiting nodes in the decreasing order of node numbers & end at the initial node. At each node we calculate the last finish time for each activity.

Step 1: initialize, $L_n = E_n$ for $j = n$.

2: set the latest finishing time for each activity (i,j) that end at node j.

$LF_{ij} = L_j$

3: Calculate latest starting time for each (i,j)

$LS_{ij} = LF_{ij} - t_{ij}$.

Step 4: Proceed backward to the node in the sequence that decrease by 1. latest occurrence time of node i (i<j)

$$L_i = \min_j \{LS_{ij}\} = \min_j \{L_j - t_{ij}\}$$

Step 5: If j = 1 (beginning node)

$$L_1 = \min\{LS_{1j}\} = \min(L_2 - t_{1j}),$$

* Mathematical Expression for Critical Activity:

→ Based on FP & BP calculations, an activity (i,j) is critical if it satisfy following conditions:

1. $E_i = L_i$ & $E_j = L_j$
2. $E_j - E_i = L_j - L_i = t_{ij}$

→ An activity (i,j) that does not satisfy the above conditions is termed as non-critical

* Critical Path:

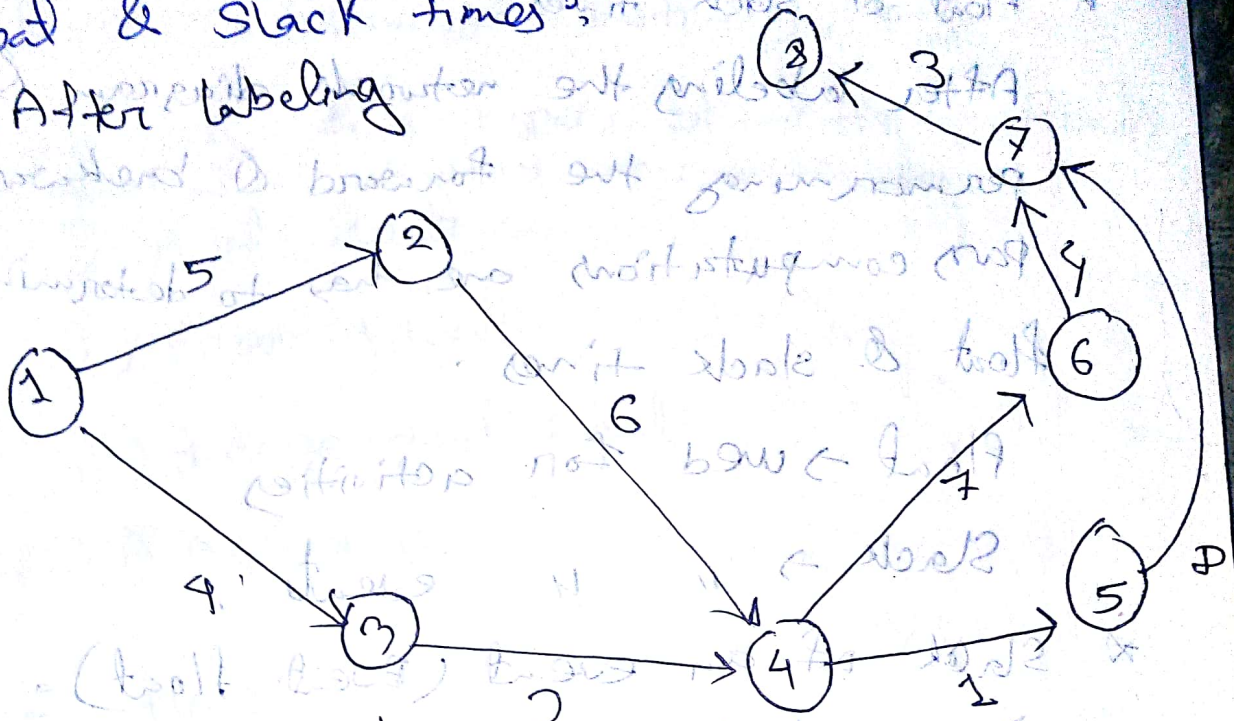
→ The sequence of critical activities (CA) in a network is called critical path. It is the longest path in the network from starting event to ending event & determines the min. time for project completion.

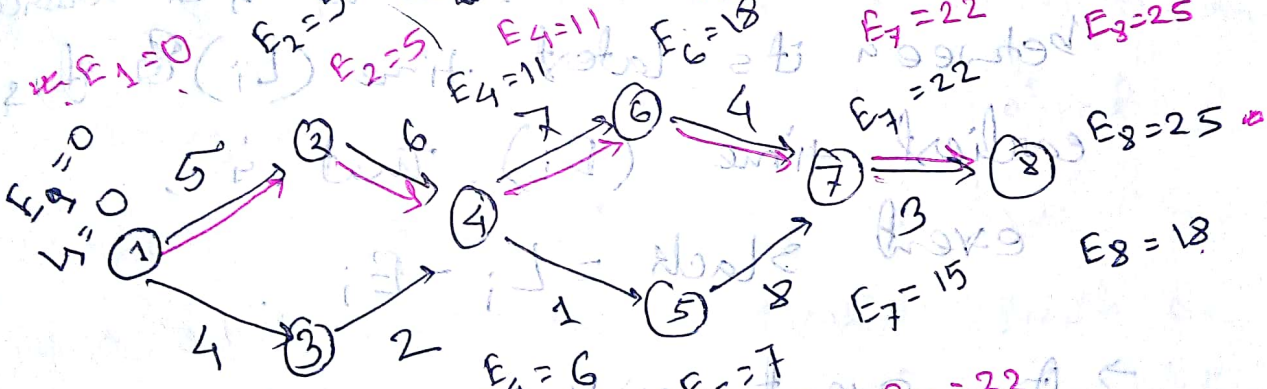
→ If an activity on a critical path is delayed by a day, the project would also be delayed by a day unless the duration of future (subsequent) activities are shortened by some other means.

→ To specify critical path on the network, double lines are used to distinguish from non-critical path.

Float & Slack times:

→ After labeling





$E_1 \rightarrow E_2 \rightarrow E_4 \rightarrow E_6 \rightarrow E_7 \rightarrow E_8$

Critical Path → 1 → 2 → 4 → 6 → 7 → 8

* Float & Slack times:

After labeling the network diagram & performing the forward & backward pass computations one has to determine float & slack times.

Float → used for activities

Slack → " " events.

* Slack of an event (Event float):

→ The slack of an event is difference between its latest time (L_i) & of is earliest time (E_i). That is,

event slack $= L_i - E_i$.

→ An event with zero slack time is called critical event.

CT → Project Scheduling

* Float of an activity (Activity float).

→ There are four types of activity floats which determine activity's time-estimates.

i) Total float

ii) Free float

iii) Independent float

iv) Interference "

i) Total float:

Amount of time by which an activity can be delayed without delay in project completion date. It implies the free time associated with the activity which can be used before, during or after completion of this activity.

Mathematically:

Total float = ~~last~~ latest finish time − Earliest finish time

or $= LS - ES$

i.e. $TF_{ij} = LS_{ij} - ES_{ij}$

→ An activity with zero total float is called critical activity.

* Free Float

A portion of the total float within which an activity can be manipulated without affecting the float of the subsequent activities, mathematically,

Free float of activity (i,j) FF_{ij}

$$= (E_j - E_i) - t_{ij}$$

Note that, $0 \leq FF_{ij} \leq TF_{ij}$

Free float is used to reschedule activities with min disruption of earlier plans.

"Design Pattern" 15.07.18.

* Evolution of design patterns:

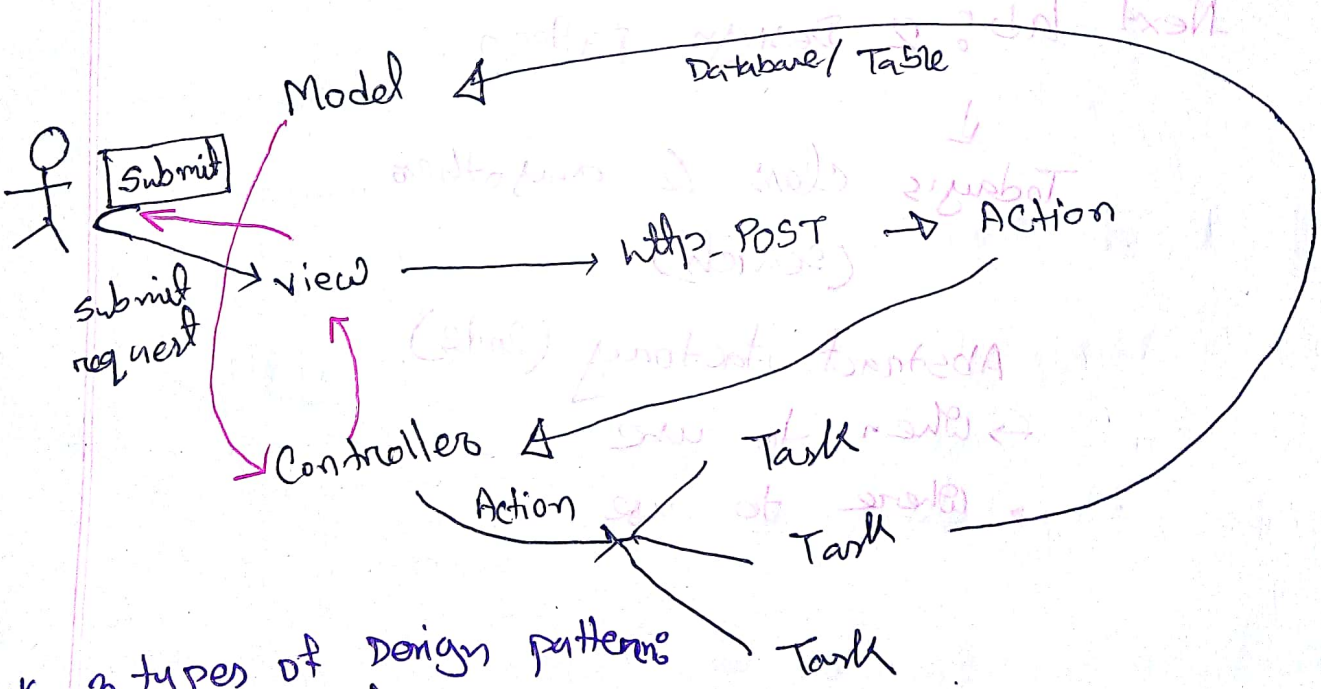
⇒ Usage:

1. Common platform for developers.
2. Best Practices.

Design Pattern: (Divided) → reuseable solution to specific problem.

Architectural Pattern (overall)

- MVC (model view controller)
- MVP (model view platform)



* 3 types of Design patterns:
  - → Creational
  - → structural and
  - → Behavioral

Abstract classes

16.07.18.

Abstract classes:

* Abstract factory Design pattern:

www.codeproject.com

CT-2

Modular Software development.

Next wednesday, class time.

Next lab: 2 Design pattern.

↓

Today's class & anyother.

(search)

Abstract factory (code).

↳ When to use

→ Where to use

———x———

"Chapter-14" 18 07 18.

Software Testing Techniques:

Unit Testing: Feature / module wise. developers do this on their own.

* Testability:

→ Operability

→ Observability

→ Controllability

→ Decomposability

→ simplicity

→ Stability

→ Understandability

—x—

"Chapter-14" 22.07.18.

What is a "Good" test?

* Internal and external views:

* Exhaustive Testing:

* Software Testing:

White-box methods — internal views

Black-box methods — external views

methods Strategies

BPT: Flow Graph Notation:

↳ Basis Path Testing

Flow chart Vs Flow Graph

Independent ways

Psycometric complexity

"Sessional" 23.07.18

Design Pattern:

Chain of responsibility.

Singleton

Factory method

Abstract factory

Chain of responsibility.

Prototype Design pattern.

Adapter " "

strategy

→

29.07.18

what is black box testing? How to do it?

User name [ ]

Password [ ]

* Username:

Test case - 1: Should not be filled with numeric

Test case - 2: 1st character should not be numeric

Test case - 3: should not contain space

Test case - 4: Minimum length 8 char.

* Password:

(i) Capital

(ii) Numeric

(iii) Special char.

(iv) Minimum length 8 char.

Final.

What type of field is given? How to test this field?

Guidelines:

UML → Unified Modeling language.

* What is UML Diagram?

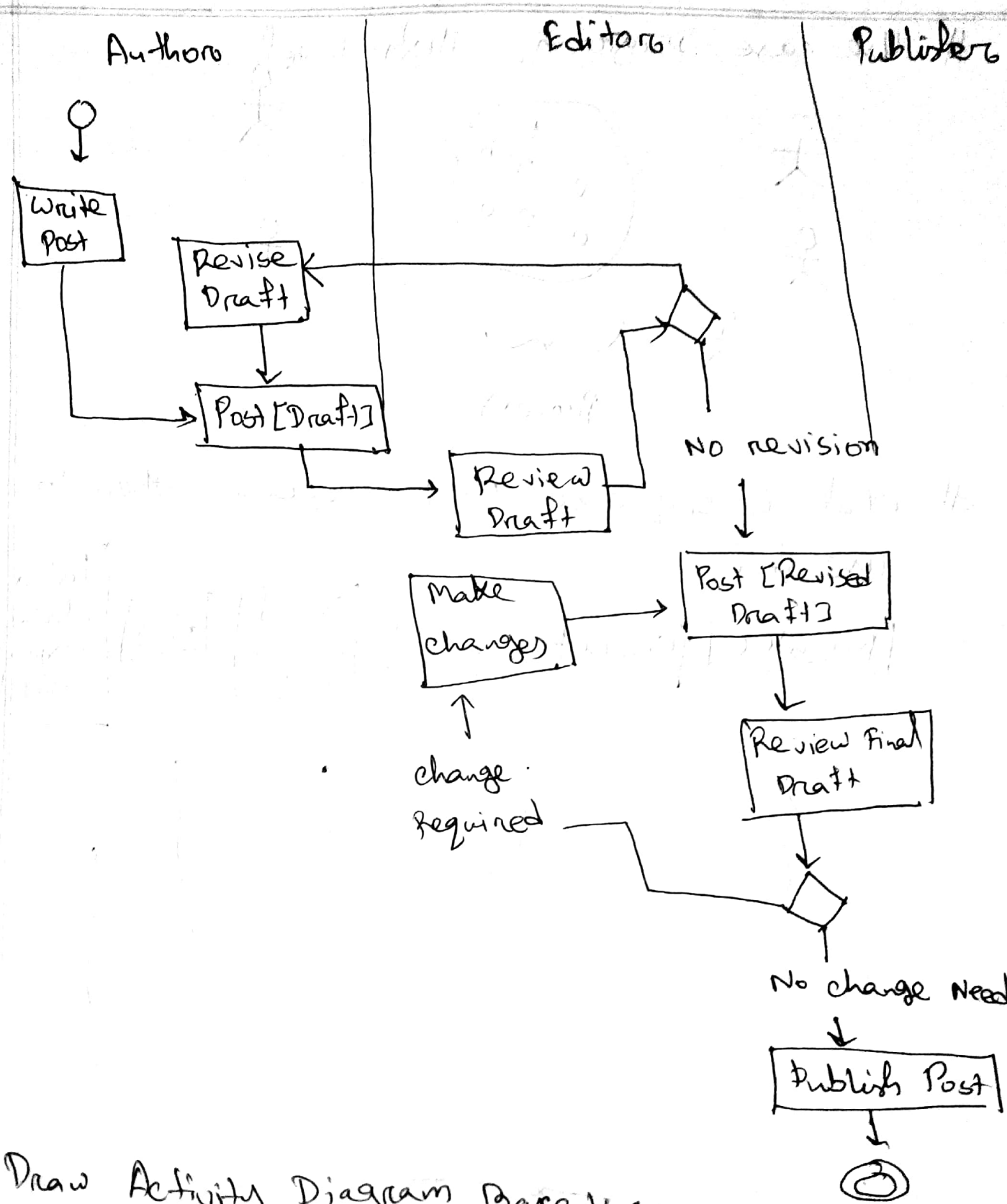
2 types:

(i) Behavioral UML Diagram (activity, use case, sequential diagram)

(ii) Structural " "

Activity Diagram:

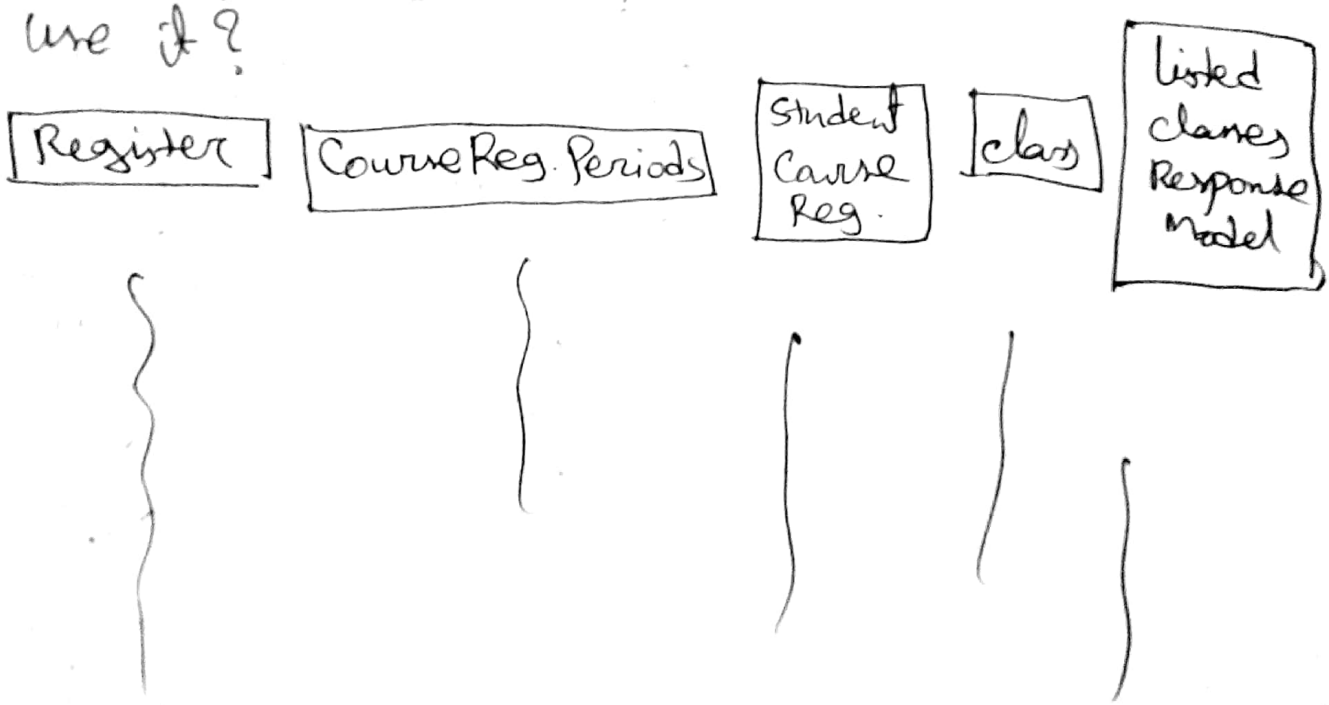
Definition.



Draw Activity Diagram Based on any given activity.

Use case Diagram: High level.

what is sequence UML Diagram? When to use it?



"Sessional" 30.07.18

UML Diagram:

(i) Behavioral UML Diagram

(ii) Structural " "

(a) Activity diagram

(b) Use case " "

(c) Sequential "

Simple Activity ⟶ Activity Diagram (few actor)

Complex & more actor ⟶ use case Diagram.

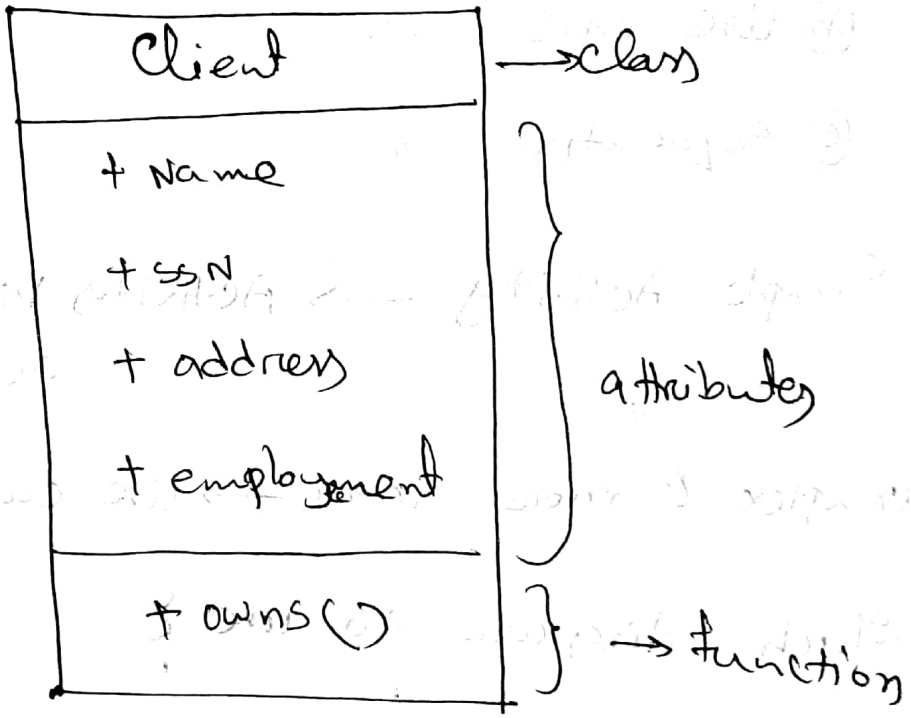
Which diagram to use?

Sequence Diagram:

Sequence of works

* Work according to function
* Use sequence diagram for functions.
* work scope period.

class Diagram:



Object Diagram :

* Com. What is component Diagram?

  * To find new data, go to master data at first.

Deployment Diagram:

→ Nodes (application server and database server)

→ Artifacts (application client and database schemas).

—— ∞ ——

01.08.18.

Software Quality Assurance: (SQA)

Consists of a means of monitoring the SE processes and methods used to ensure quality

* what is SQA engineering?
* what does a software quality engineer do?

Tester Vs SQA team

* Software Quality control.

what is Quality Management?

Quality Defined.
Quality Control
Cost of Quality.
Kinds of Quality Cost.
Purpose of Reviews.

—— ∞ ——

Last week → lab final
+
Report
+
Presentation

"Sessional"

"Solid Principles" 27.08.18

* Solid Acronym and Introduction.

Motivation behind the usage of Solid Principles:

Solid Introduction:

Solid Acronym:

SRP → S
OSP → O
LSP → L
ISP → I
DIP → D

Design Pattern Vs. Solid Principle : (v.v.I)

Design Principle
Design Pattern.

* A successful application development depends on.

Architecture
Design Principles
Design Patterns.

CT-3 → Till here.

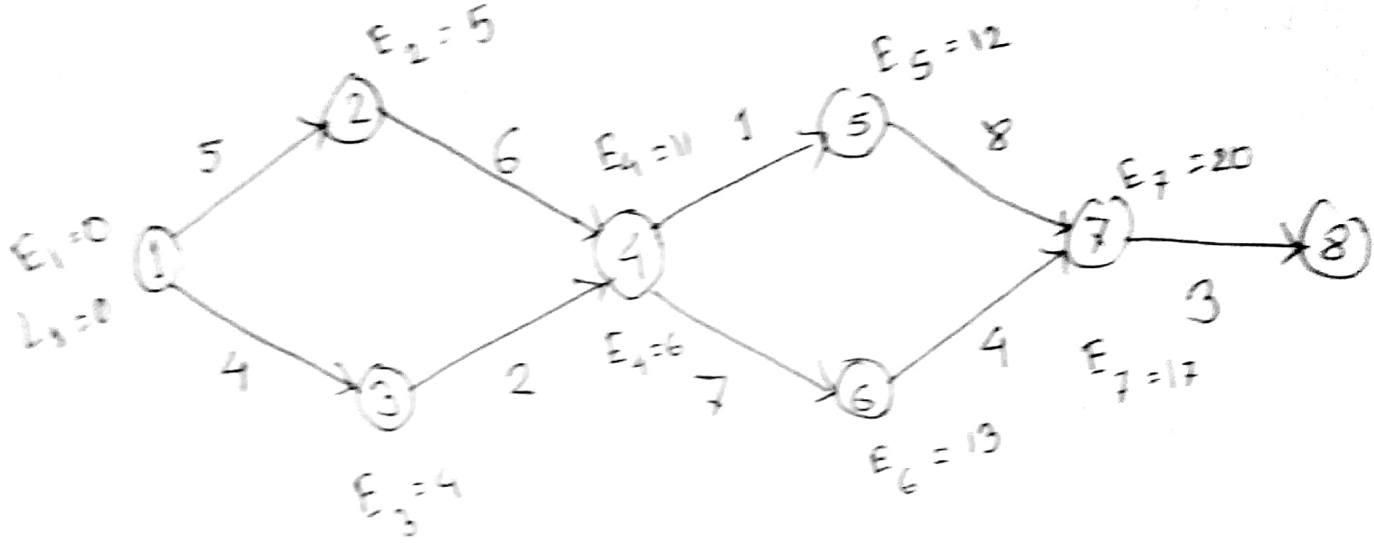
"Sessional" 03.09.18

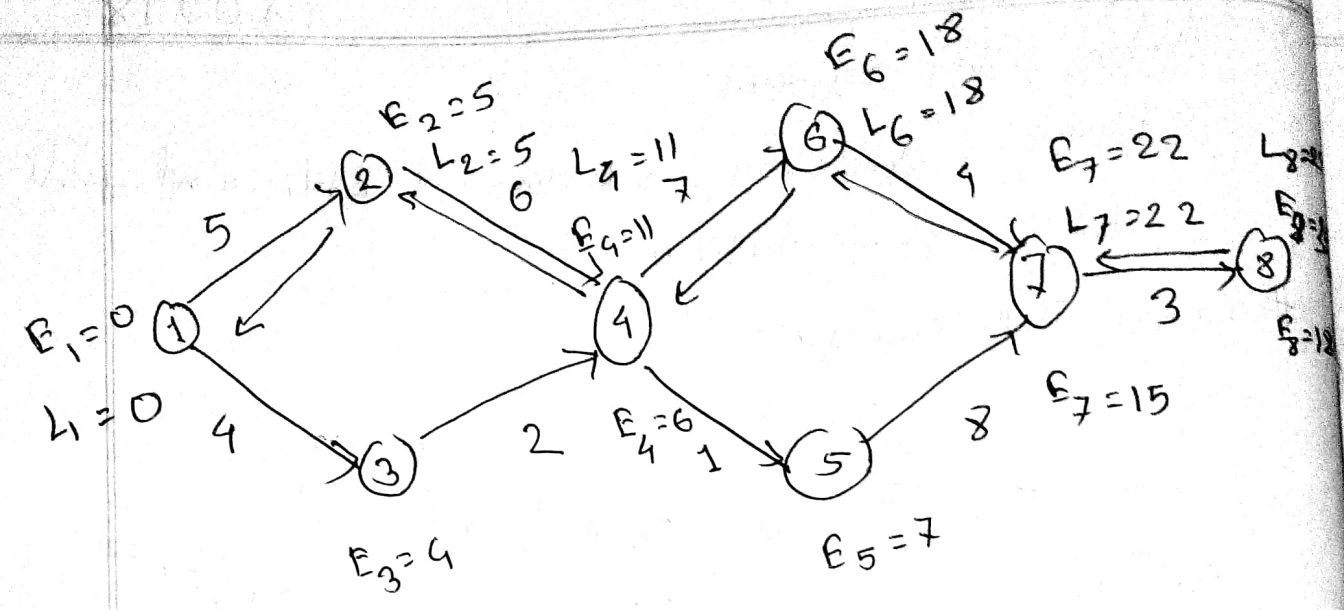
"Special class" 08.09.18

Information System

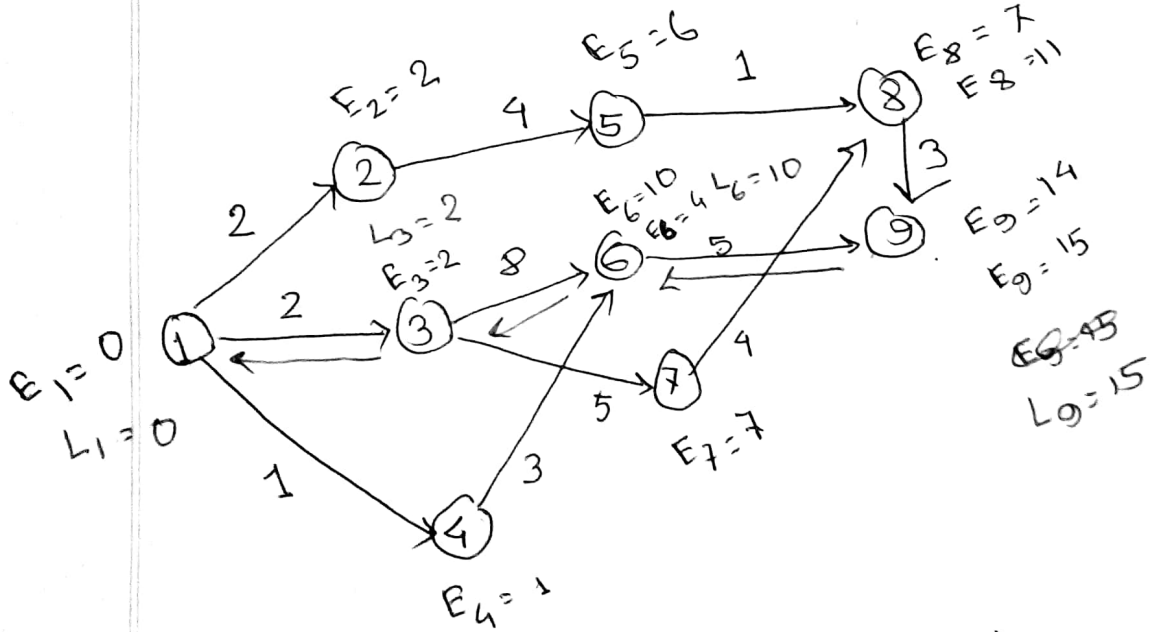
Software Engineering → Economical, Efficient, Real life implementation

Critical Path Analysis:



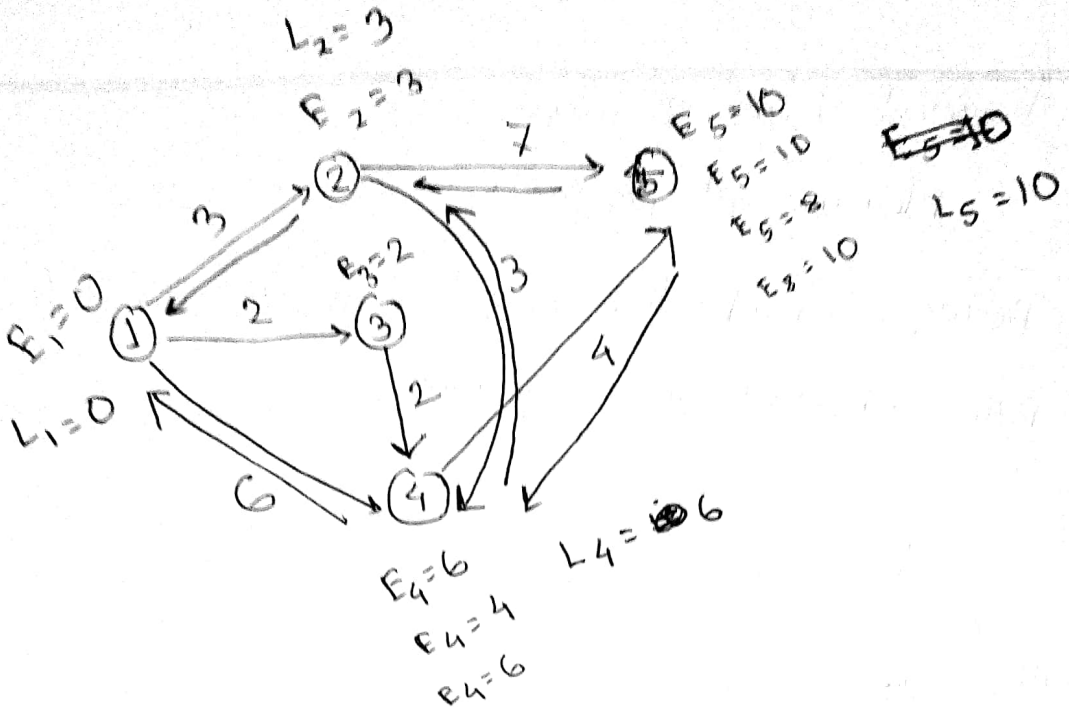


critical path: 1 → 2 → 4 → 6 → 7 → 8



Path duration = 15

Critical Path: 1 → 3 → 6 → 9



Critical Paths :

$1 \rightarrow 2 \rightarrow 5$

$1 \rightarrow 4 \rightarrow 5$

$1 \rightarrow 2 \rightarrow 4 \rightarrow 5$

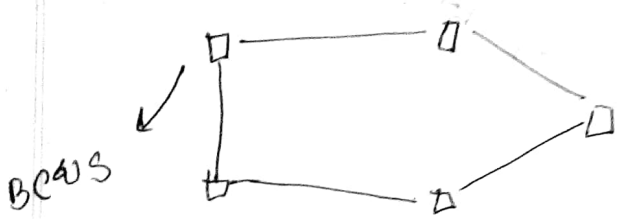
Path duration = 10

Earned Value Analysis:

Measure of progress

BCWS → Budgeted Cost of Work Scheduled.

BAC → Budget At Completion.



$$BAC = \sum_{i=1}^{n} BCWS_i$$

⇒ Time division → (PERT) PERT uses 3 Cases :
(Handles uncertain activity time)

(1) Most pessimistic Time "

(2) Most optimistic "

(3) Average time. (most likely durations)

→ It determines probability for each duration, whereas CPM considers most likely duration.

* Advantage of CPM:
  - → useful for large project
  - → straightforward
  - → Graphical networks
  - → Critical path pinpoints activity

* Optimistic → go well as per plan
* Most likely → most frequently
* pessimistic → when everything is assume opposite to the plan. The longest time the activity can take in worst case. This excludes major Catastrophes.

* The fundamental assumption of PERT is that these three time estimates from the end points and mode of beta-distribution.

initial ↙ mid ↓ Final ↘

t_e = expected time

V.G.

t_m = most likely time

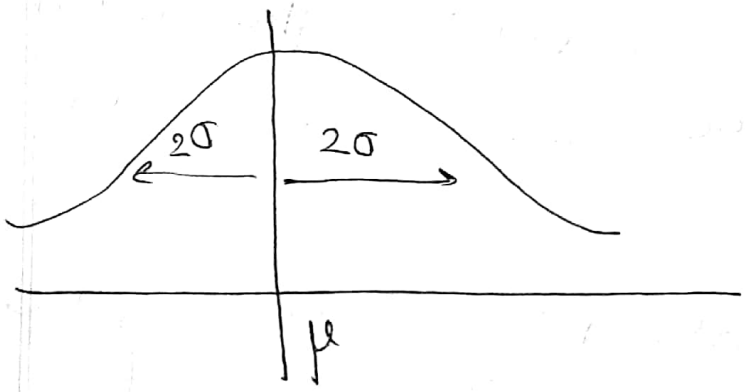
$$t_m = \frac{t_o + t_p}{2}$$

t_o = optimistic time

t_p = pessimistic "

$$t_e = [2t_t + (t_o + t_p)/2]/3$$

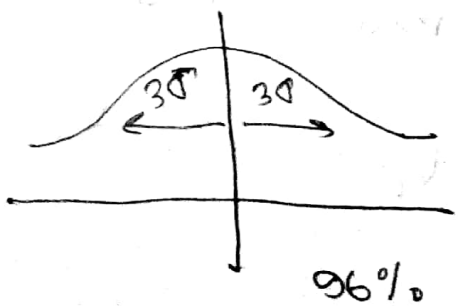
$$= \frac{(t_o + 4t_m + t_p)}{6}$$



95%

$$t_p - t_o = 6\sigma$$

σ = standard deviation



96%

σ^2 = Variance

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

$$\sigma^2 = \frac{(t_p - t_0)^2}{36}$$

Negative Float in a PERT network:

→ In a PERT network there is a scheduled date at which project is expected to be complete. This value of the scheduled completion time T_s is also called the time for the final event.

→ This value of T_s may be greater than or less than the mean expected time, 'T_e' of the project.

then, $T_s > T_e$ => A positive Float

$T_s < T_e$ => A Negative Float

$T_s = T_e$ => Zero Float.

* In such cases critical activities are not specified by the conditional of zero float along the path.
* The critical path is the path of least float for such kind of PERT network in which backward pass is based on the scheduled date.
* A negative float implies that activity must be reduced by that amount of float to keep the scheduled date unchanged.
* A negative float is arises in PERT, particularly when the mean expected project time of final event is less than the ~~sched~~ scheduled date.

PEST Algorithm:

Step 1: For each activity obtain estimates of optimistic time, pessimistic time & most likely time.

Step 2: Using these estimates, compute the mean & variance of each activity times by the formula

Step 3: Based on mean activity time, determine the critical paths by C.P.M.

Step 4: Once the activities are identified as critical activities, add the means & variances of critical activities to find the mean & variance of the project completion time.

Step 5: Total project time is normally distributed with the ~~mean~~ mean & variance determined in step 4.

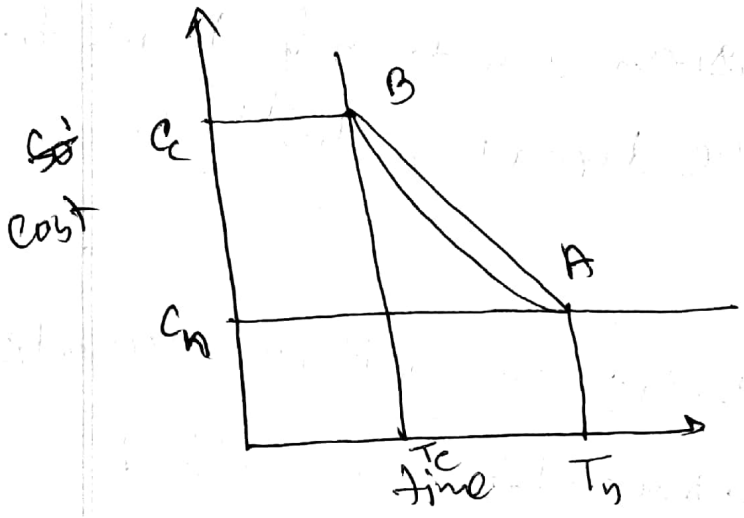
Time Cost - Trade off in PERT:

→ Project managers sometimes may have the option of crashing activities, spending extra money to compress an activity's duration by using overtime, subcontracting, expediting, materials ~~etc~~, etc.

Indirect time → can not crash

Direct time → can crash

$$\text{Cost slope} = \frac{\text{Crash cost} - \text{Normal cost}}{\text{Normal time} - \text{Crash time}}.$$



Probability of Meeting the Schedule time (deadline):

→ With PERT, it is possible to ~~determ~~ determine the probability of project completion on schedule.

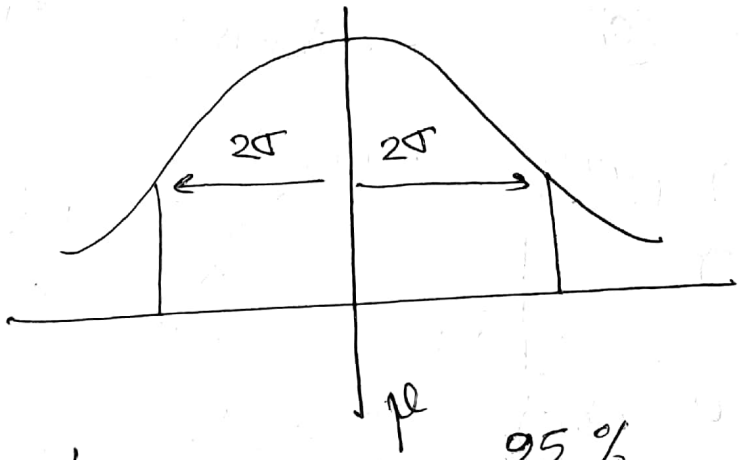
→ Initially there may be the latest time T_L for each event, but after the initiation of project.

$$\text{Prob}\left(Z < \frac{T_s - T_e}{\sigma_e}\right)$$

T_e = Expected project completion time

σ_e = standard deviation of schedule time

Z = Standard normal variable.



$\mu = 0$

$\sigma = 1$

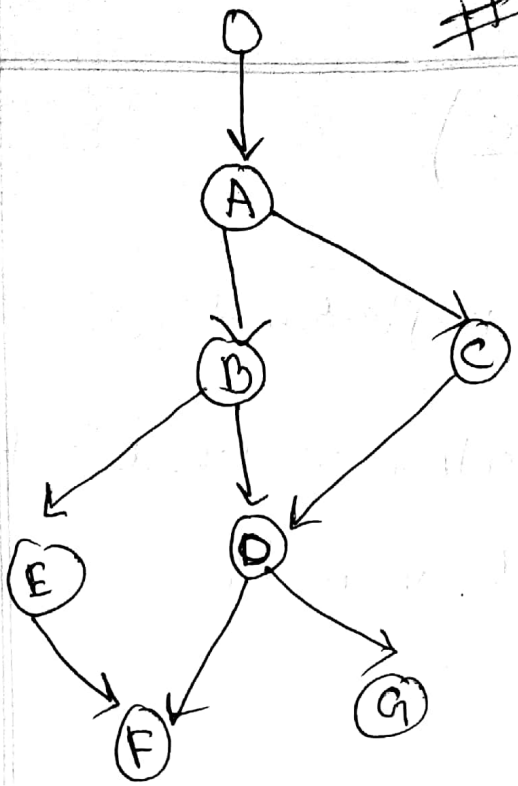
95 %

$$Z = \frac{x - \mu}{\sigma}$$

* (i) Draw network, find critical path, expected project completion time from following data table
(ii) What project duration will have 95% confidence of completion.

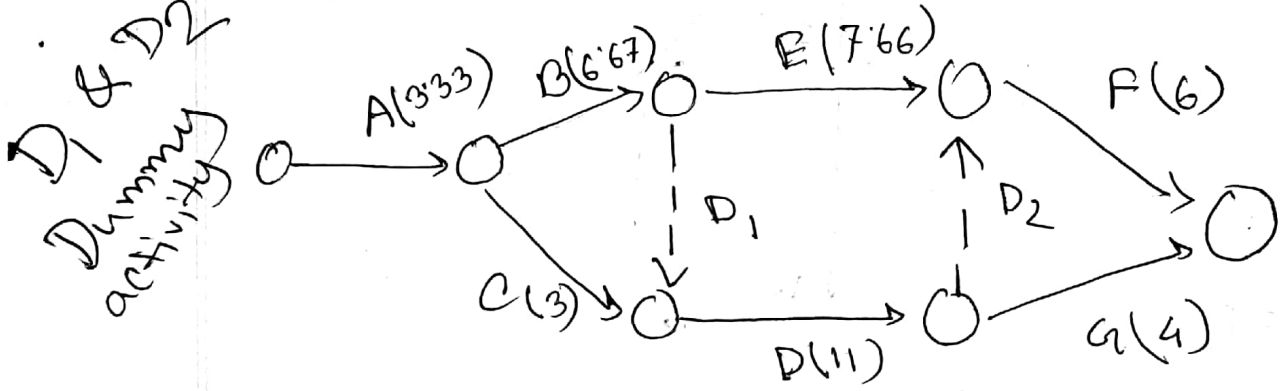
Earned Value Analysis.

Engineering
Estimation
scheduling



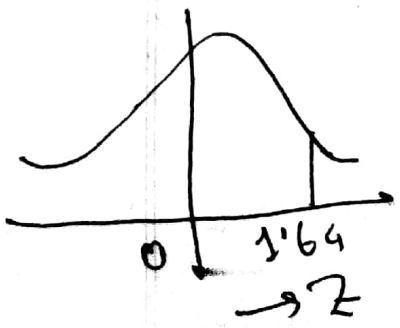
$$t_e = \left(\frac{t_o + 4t_m + t_p}{6}\right)$$

for A, $t_e = \frac{(3\times4)+1+7}{6} = 3.33$



Critical path : 1 → 2 → 3 → 4 → 5 → 6 → 7

$$\sigma_e^2 = \frac{(t_p - t_o)^2}{36}$$



$$Prob\left(2 < \frac{T_s - T_e}{\sigma_e}\right)$$

$$Pr(2 < 1.64)$$

V.V.9

09.09.18.

The Spectrum of Product management concepts;

V.9.

4 P

⇒ Software Process & Project Metrics

- ↳ measurement
- ↳ metrics Domain
- ↳ metrics for software Quality.

McCall's Triangle of Quality

Measure, metrics, Indicators.

Software measurement

- ↳ Direct
- ↳ Indirect. (LOC) line of Code.

Web Engineering Project metrics

Observations on estimation.

- ↳ Risk
- ↳ Uncertainty

Present value method

SW scope, First Activity of SW & PP

—x—

12.09.18

CPA → Critical Path Analysis → (V.V.I)

CPM → till

Cyclometric complexity → (V.V.I)

—x—

Speech. RecognitionListener

" . RecognizerIntent

" . SpeechRecognizer

telephony. Sms manager.

—x—

"Sessional"

17.09.18

⑥ Future Plan → Google API's location sending detecting local area, find nearest security service.

language independent.

—x—

17.09.18.

* learning goal.

lecture - 1

→ Data and information
→ Types of information

slide - 4
10
12
13
14
17

→ 4 types (Data manage in each level)

→ Information needs (Pyramid)

→ Needs for information system.

→ Management structure → (concepts)

→ 21 slide

→ 22 slide.

2

* System Analysis :

Slide - 4 → all steps

4

15 all

→ life cycle

→ 9 steps and system design and hardware (details)

→ System life cycle diagram (X)

→ Role of system analyst.

→ Attributes of a system analyst.

→ Tools used by a " "

→ DFD, VML, testing tools.

* Feasibility Analysis (3):

~~Ste~~ Slide – 5
6
7
8
9

math concept

→ Feasibility analysis

→ How to ~~~~ analysis a system.

→ learning goals.

↓ (a) Technical
(b) Operational
(c) Economic

→ steps in feasibility analysis

→ guidline for searching goal.

→ { case study
→ hotel

→ math

→ Present value method

→ Pay back period

→ Normal approach
present value } EBA

* Information gathering (4)

Slides → 5, 6, 7, 8, (9), 10, (12), 14

SRS - definition.

→ Information gathering strategies.

→ " source

→ " gathering method

→ Planning an interview → CEO (meeting)

→ Interview technique. } Data Store

→ Use of questionaries } DFD ?

→ Symbols of DFD.

* DFD (5):

→ Why do we need DFD? How?
  ↳ process & entity

→ Mess management system

→ DFD fig (Basic)
  ↳ Data store
  ↳ ~~Exter~~ Extened DFD
  ↳ 3 list, 1 Process, 1 entity, 1 data Store.