

SCD

Software Construction

Implementation

< Logic Building

SDLC

- ① Requirement
- ② Design
- ③ Implementation
- ④ Testing
- ⑤ Deployment
- ⑥ Maintenance

L#1

AI

5-9

→ What is Intelligence

→ What is AI?

- Human like
- Create machines that can think and act like human
- ML: train the machine to make the decisions

• Rational - Like

- Make decisions according to situation
- Human Like decision making policy

• Reflexive Action

- Actions that are performed suddenly
on a special condition

e.g.: Touching a very hot thing
you immediately pull off your hand.

eye blinking

• History of AI

- 1960s → start

• AI Applications

- Robotics

SCD

Software Construction

- ① Software Engineering → Design, Documentation
- ② Coding

→ ⚡ Nature of Problem

Functional Requirement

- ① Understand Problem Domain

- ② Requirement Gathering (what is required?)

• ① User requirement (User language) [what?]

• ② System requirement [How?] (How to implement)

Non-Functional Requirement

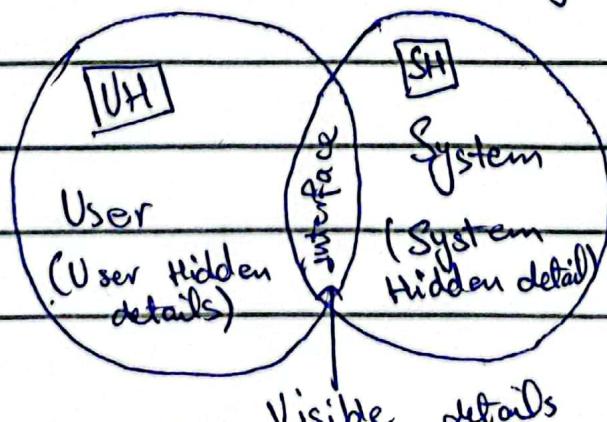
- ① Product Requirement (Hardware Req., Product Deployment)

- ② Organization (Organization Proof) (Security)

- ③ External Requirement (All factors that are imported from external system)

↳ Python old version.

↳ Ecommerce purchase payment method failed



WRSPM

W → World (Domain)

R → Requirement

S → Specification

P → Programming

M → Machine

③ Architecture

→ Decompose an enterprise system into independent sub-system that have value in the system.

→ How these sub system interact

→ Principles and guidance for the design evolution over time.

Lecture #3 AI

12 - 9 - 2023

→ Agent

↳ Intelligence is measured by

① Situated in a environment

② Autonomous (Independent)

③ Proactive ④ Social (Interactive)

→ Task environment

Types / Structure of agents

1- Simple Reflex Agent → Have no history

2- Model based

3-

4-

SCD

→ Module depends on other

Persons or companies may own cars.

The car owner ID is the ID either

the person or company, that owns

the car. A car may have only one owner (person or company). A car may

have a loan on multiple loans.

A bank provides a loan to a person

or a company for the purchase of a car. Only the loan owner may

obtain a loan on the car. The car

owner type and the loan customer

type indicate whether the car owner

loan holder is a person or company.

Make a list of nouns and verbs

classes

① Person

② Company

③ Car

④ Loan

⑤ Bank

SCD

Domain Modelling

→ Understand the problem

WRSPM

↓
World (Domain) → Understand your domain

Requirement Elicitation

↳ User requirement (what the user need?)

User → Between user & environment

WR(S)PM

Environment

Software Architecture

→ Software Components (Sub-system)] that have their own value

→ How they interact

(sub-system)

→ Independent fracture and how they

provide facility and interact with each other

SA Model

Module → Dependant (Can be many many)

① Pipe & filters (Same input, output) (Compiler)

Data is present in high Integrity

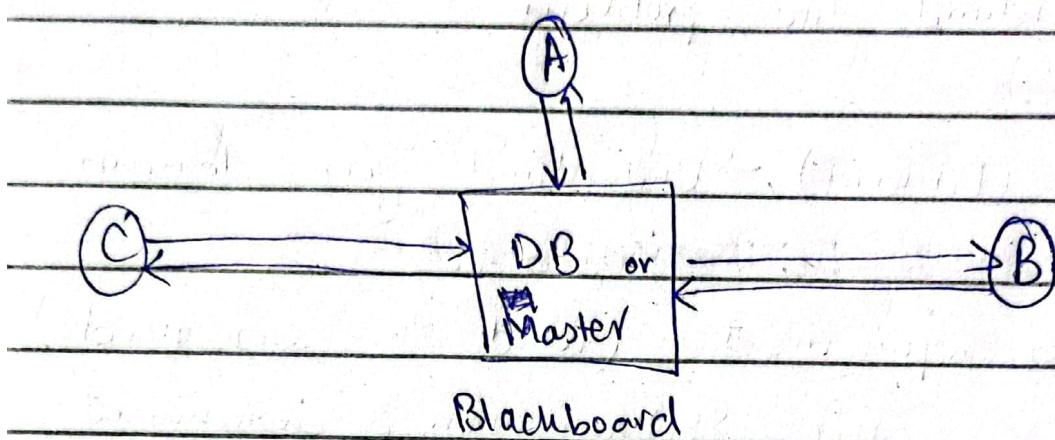
② Blackboard (Central database, Different Modules)

③ Layered (MVC, MVM) (Implement in different layers and don't interact)

④ Client Server (Request, Response or Served)

⑤ Event based (ASP .NET) Output on specific event

→ Architectural Models that can be follow together or separately



Event based ⇒ Occurs on specific event

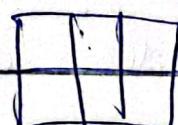
Software Design

System Structure (Decompose into Components)

Interaction of components (sub-systems)

→ Modular Decomposition → (Software Design)

↓ Design classes



Software Design

→ Design of a specific (particular) module

Modularity

Breaking down of a component and interaction

A module must have these four

things:

- ① Coupled - Coupling
- ② Cohesion
- ③ Information Hiding
- ④ Data encapsulation

AI

Problem Solving by Searching

Measuring problem-solving performance

→ Completeness: Always returns the optimal solution (Tell if solution exists or not)

→ Optimality: Optimal Solution

→ Time Complexity: How much time it takes

→ Space Complexity: How much space it takes

Branch factor: Maximum no. of nodes

Search strategies

→ No information about the path or the path cost and do not have additional information.

→ Often called blind search

→ e.g. Breadth-first, Depth first, Uniform cost, Bidirectional search

Frontier

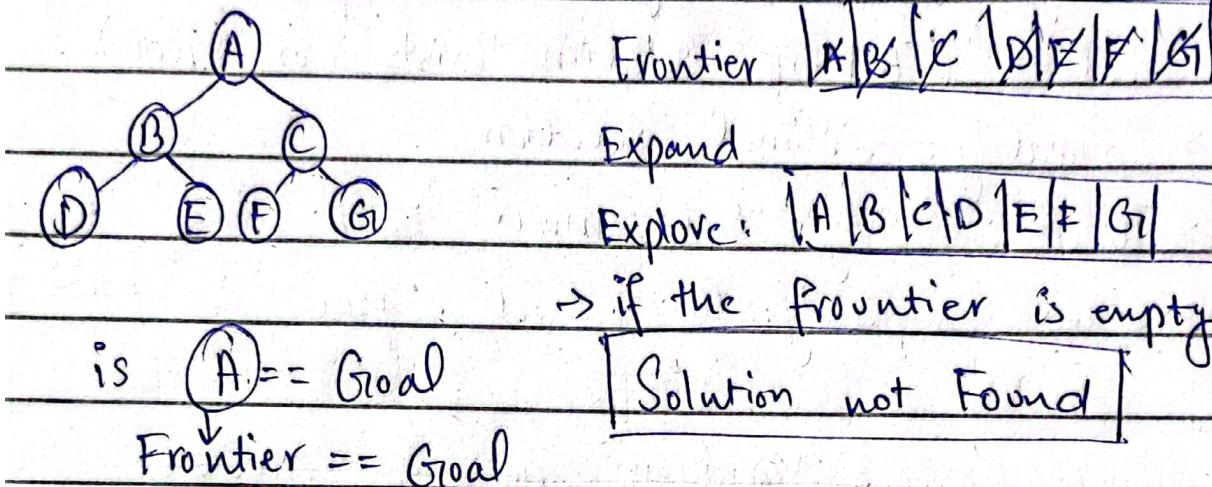
Node that may or may not have child node but they are not expanded or searched.

Breadth - first Search

Implementation:

assumes that every cost is 1

- Fringe is FIFO (Queue)
- Goal = M



BFS:

- ① Complete : Yes
- ② Time : $O(b^d)$ → exponential in d
- ③ Space : $O(b^d)$ → keeps every no. in memory (problem)
- ④ Optimal : Yes (if cost = 1 per step)

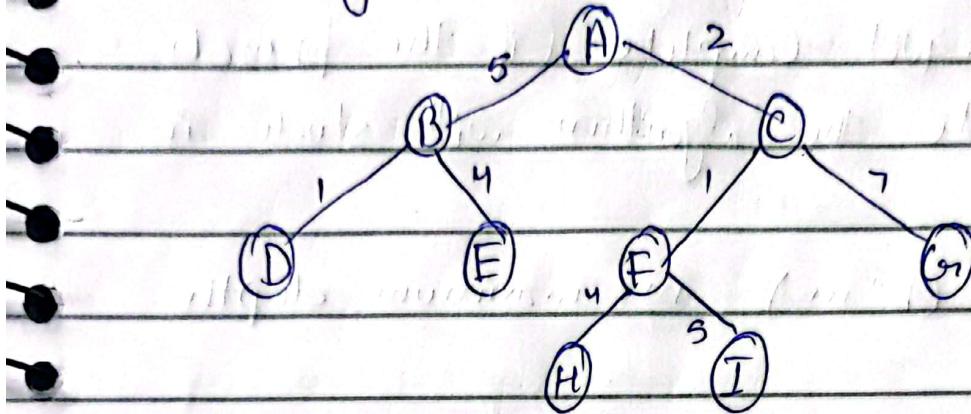
⇒ Time and Space complexity in exponential is not considered good.

Uniform-cost Search

- If the cost is equal of all nodes then BFS is optimal
- Extension of BFS, depends on costs

Implementation : (using Priority Queue)

- Priority queue
- $[X] = g(n)$



- Elements are arranged according to costs of nodes (ascending order)
- If the cost of 2 nodes are same then we follow BFS (Queue)

UCS:

- ① Complete: Yes (if b is finite)
- ② Time: $O(b^{c^*})$
- ③ Space: $O(b^{c^*})$
- ④ Optimal: Yes

Depth First Search

- Always expand one of the deepest level and then back track
- Backtracking search
- Implementation:

LIFO (Queue) (Stack)

DFS

- ① Complete : Not Complete (if the branch is infinite the algorithm will stuck in loop)
- ② Time: $O(b^m)$ $m = \text{maximum depth}$
- ③ Space:
- ④ Optimal : doesn't guarantee best solution

CN

21-9

- | | | |
|----------------|-----|------------------------|
| ① Access Link | { } | Packets = $2L$ |
| ② Edge devices | | Link Speed = R (bps) |
| ③ Network Core | | $2R$ |
- No. of bits in each packet = L

Packet Transmission delay = $\frac{L}{R}$ (bits)

→ RJ11 - ethernet

→ RJ45 - Telephone (landline)

→ Routing Algorithm

→ Store & Forward

→ Arrival > Transmission Rate then

this cause queue

BPE

→ Activities (Streamline) $A_1 \rightarrow A_2 \rightarrow A_3$

→ Sequence

(Left - to - right) Workflow → Order

ERP → Enterprise Resource

SCD

25-9

Tightly Coupling: (Not Good) Content coupling

① Module A directly access Module B's data member.

② Common coupling.

③ Module A and B are relied on some global data.

④ Module A relying on externally imported imposed format (protocol / interface).
→ XML and Jason

External coupling

Loosely Coupling: (Highly)

Data coupling

① Module A only pass parameter for requesting functionality of Module B

② Module A sends message to Module B,

Message coupling

Medium Coupling: (Somewhat accepted)

① Module A controls the logical flow of module B by passing information or by using flags. (Control Coupling)

② Module A and B rely on some composite data structure changing data structure directly affects the other module.

(Data Structure Coupling)

Cohesion:

Weak Cohesion:

• Coincidental Cohesion

① Different parts of module are together just because they are in a file.

Temporal Cohesion

② Different parts/code / functions are activated at the same time.

Procedural Cohesion

③ One part follows the other in time

Communicational Cohesion

④ Similar parts/functions are grouped. They are similar but perform different thing.

Medium

communicational
↓

- ① All elements operate on same inputs and produces same output.
- ② One part output serves as input to other part. → Sequential

Strong

- ① Each operation in module can manipulate object attributes → Object Cohesion
- ② Every part of the function is necessary for execution of single well-defined function

↓ Functional cohesion

AJ

26-9

→ DFS is better in space complexity than BFS

Space Complexity = $b \times m$

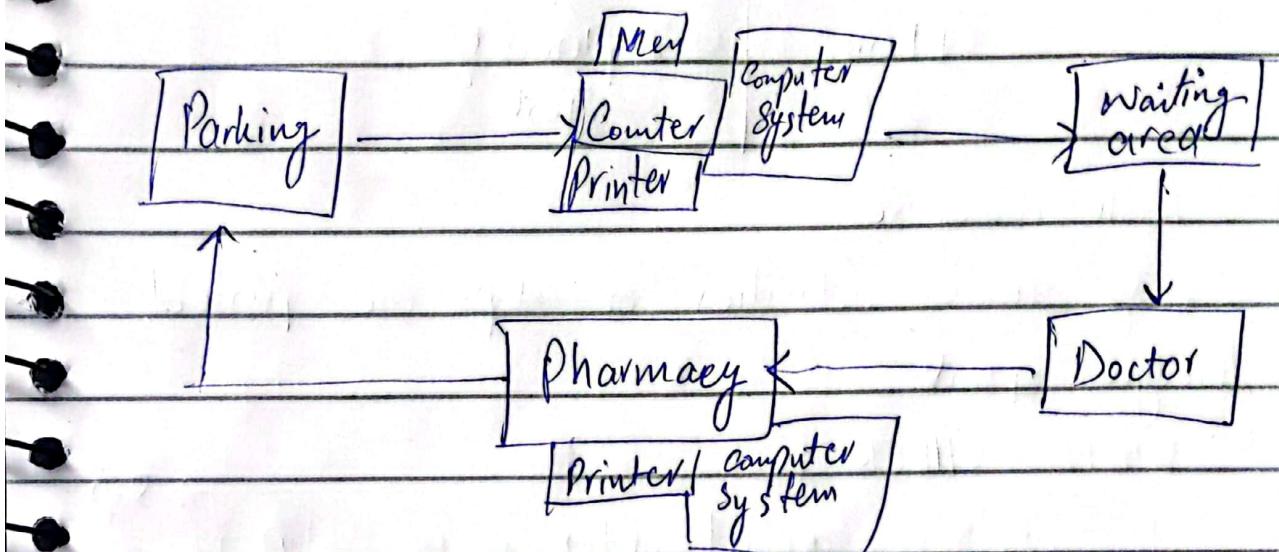
→ Depth Limit Search

↳ The Solution for the depth limit search, fix the limit of the depth.

→ Termination guaranteed

→ Failure Mode and Effect Analysis

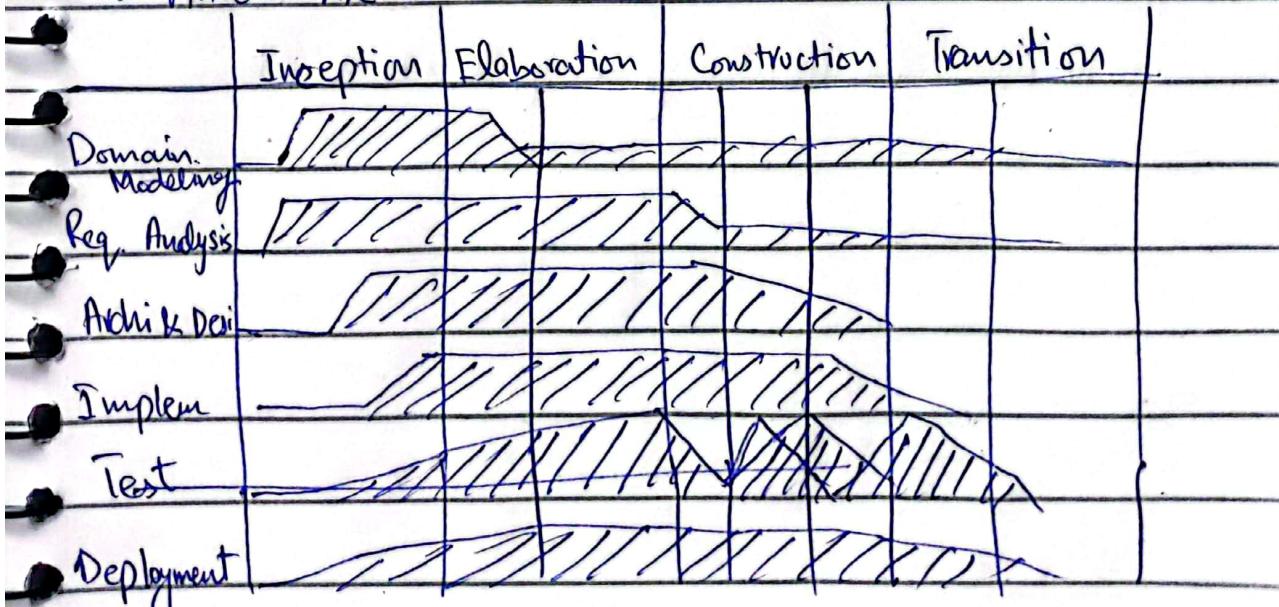
BPE Process



SCD

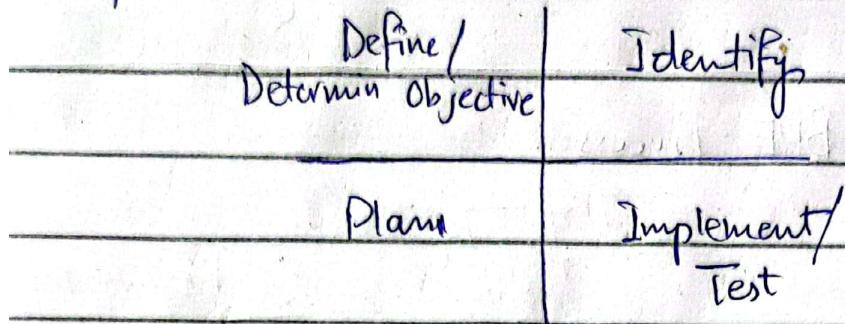
4-10

- Iterative Model
- Unified Model
- Architect Architecture



→ Understand & Deployment is Difficult

• Spiral Model



→ Risk analysis

→ Only those activities or steps are passed through spiral

↳ GATE CHECK

→ A gate is applied after every step

→ There are multiple ways to solve a problem in software development

→ Team stays motivated

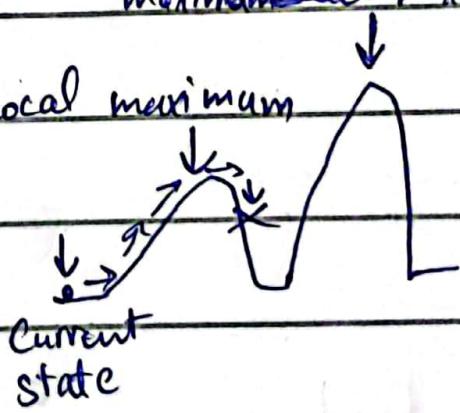
→ Saves time and money

• Agile

→ 17 people brainstorming → agile mindset

• Local Search

- We use it because it improve a single search iterately
- The goal itself is solution
- Don't keep track or path but keeps a single "current" state
- Hill climbing algorithm
- Stuck on local maximum
- Local maximum get worst in higher dimension
- Local maximum is ~~not~~ one solution but not optimal
- Continue in same direction (one direction) forward
- Plateaus
- Restart or change direction



- Random restart hill climbing
- Meta-heuristic
 - Optimization Problem: Reduce the Cost →
 - Local minima, Global maxima
 - up hard Problems

- $\Rightarrow f(x_2) > f(x_1) \rightarrow$ directly move
- $\Rightarrow f(x_2) < f(x_1) \rightarrow$ then use energy fn.
 $\Delta E = f(x_2) - f(x_1)$
- $P = \frac{\Delta E}{T}$ ← assumed number (random)

CN

10-10

Principles

Book Pg #133

• P2P

• Client Server

No data loss / Reliable

→ Packets \Rightarrow FCP \uparrow \rightarrow 100% Packets (Skype)

\Downarrow UDP \rightarrow (WhatsApp video call)

TCP \rightarrow Reliable \rightarrow Skype Vide Calling

UDP \rightarrow Not reliable \rightarrow WhatsApp Calling

Agile Principles Values

17 People → (Senior & Junior)

4 → Values 12 - Principles

① Individuals and Interactions over Process and Tools

② Working Software over comprehensive document

③ Customer Collaboration over Contract negotiation

④ Responding to change

Agile Principles

① Highest priority is to satisfy customer through early and continuous delivery of valuable software.

DSDM

FDD

SCRUM

Crystal

XP

Custom

Lean

② Welcome changing requirement even late in development. Agile process harness change for the customer is competitive advantage.

③ Deliver working software from couple of weeks to couple of months with the preference to the shortest timescale

④ Business people and developers must work

together daily throughout the project.

⑤ Build the projects around motivated individuals. Give them the environment and support they need, and trust them to go the job done.

⑥ The most efficient and effective method of conveying information to and within a development team is face to face conversation.

⑦ Working software is the primary measure of progress.

⑧ Agile processes promote sustainable development.
⑨ The sponsors, developers and user should be able to maintain a constant pace indefinitely.

⑩ Continuous attention to technical excellence and good design enhance agility - cost of exploration.

⑪ Similarity the art of maximizing the amount of work not done is essential.

The best architectures, requirements and designs ^{emerge} ~~emerge~~ from self-organization

teams.

- ⑫ At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

AI

10-10

Adversarial Search

- Competition
- Typical AI assumption
- Zero-sum game (gain of one is loss for other)

Search vs Game

SCD Continuous

10-10

IDEA

- ① Individuals should be able to plan, deliver, monitor and impose the quality and timeliness of their own work.

- ② Use data to justify refute unreasonable demands

Say "Yes" with confidence and "No" with data & options

- ③ Learn from experience use data from

one piece of work to improve the next

Principles of Personal software Process

Measure stuff (size, time, efforts, defects)

② Measure Consistently

Use correlation to judge usefulness of time

to predict future performance

SCD

Self Reading

Team Software Process

PSP

TSP

→ Motivated Team members

→ Scripts (2-3 page) [if there is a prob.]

Humphrey's Idea

[then follow this]

- The team should be self-directed

- Definite tasks assign to team individuals

- Communication [that plays role in achieving a single]

- Team members are dependant]

Principles

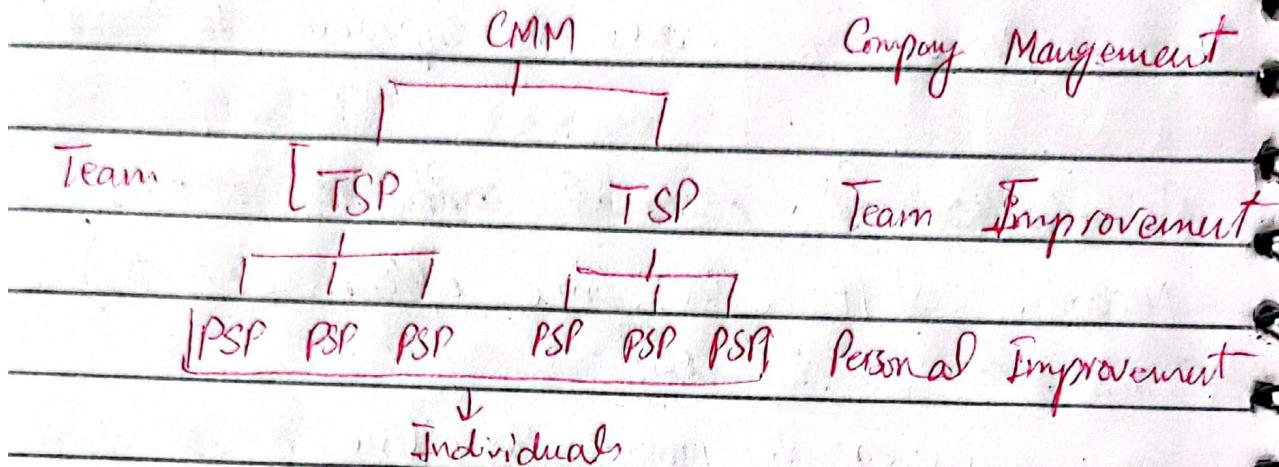
- Measure the task for each individual/Analyse

- Regular meetings

- Rigorous Planning → Planning for next tasks

(Achievable goals)

targets



SCRUM

- 1 - 4 week (Working software)

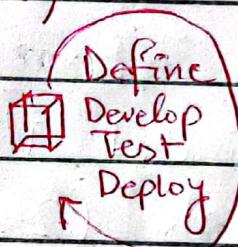
- Backlog

- 1 Deployment in Waterfall

method = 4-6 Deployment

in Scrum method

Deployment = Sprint



(Scrum master
manages product
backlog)

① (Priority) → statement

② Backlog → Simple English statement

③ Sprint Planning Meeting

④ Sprint Backlog ← Only for one specific sprint

↳ Daily meeting [what was previous task]

[Problems]

[Assignment
Time]

[Time Management]

[Team Members]

⑤ Finish Product (First Sprint)

⑥ Sprint Review (Problem, issues)

⑤ Conclusion

← conclude the research

— Sum-up research thesis

Research Proposal (7-8 pages)

- Summary of research thesis
- Attractive & concise and centered on the ideas that are in research
- Finding \Rightarrow Assumptions
→ Ch 4th and 5th is combined
- Citation is must

AI

Minimax Algorithm

→ 2 player game and each plays optimally.

→ Best of one is worst of other

Max starts from $-\infty$ $\max(-\infty, 0)$

Min starts from $+\infty$ $\min(+\infty, 0)$

→ Algorithm implementation

Multiplayer games

→ Two-sum cannot be follow

→ Player make alliance

Alpha beta pruning

→ based on min-max but it eliminate exponential space complexity

$$[\alpha, \beta] \\ [-\infty, +\infty]$$

→ Prune whenever $\boxed{\alpha \geq \beta \text{ for max}} \quad \boxed{\alpha \leq \beta \text{ for min}}$

if $\alpha \geq \beta$ or $\beta > 2$ then no need to explore

the β branches

→ Alpha-beta is best for pruning trees and

it complexity is $(b^{\frac{m}{2}})$ or $b^m (m/2)$

→ For worst case it is b^m

CN

12-10

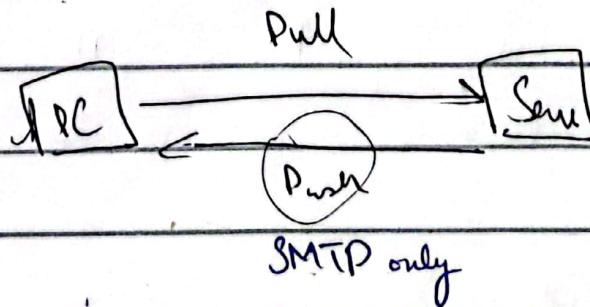
• Protocol

• TCP vs UDP

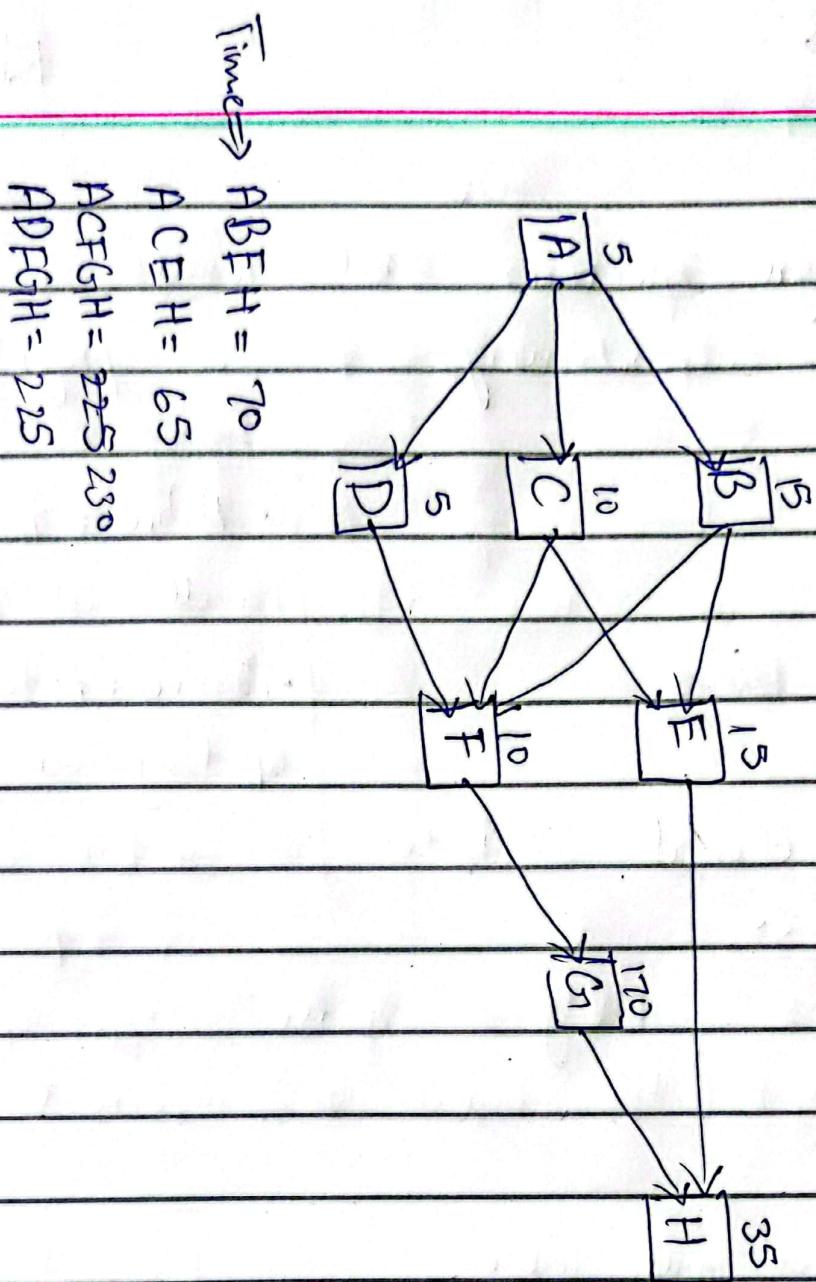
• HTTP

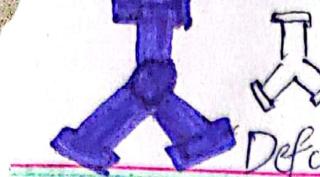
• Persistent HTTP and non-Persistent HTTP

• Cookie



Kroll business center
Country Engineers Design Department





Sound
Rish

SCD

17-10

Error: Illegal operations that results in abnormal/malfunctioning of a program.

① Syntax

② Logical

③ Runtime

Error handling

- Non-functional attributes
- Correctness
- Robustness - ability to handle many inputs that might be correct or incorrect

Approach

• Active : Program handle at realtime

• Passive : After inputting wrong that

the database cannot store

type of

Technique

- Return a null value
- Substitute with next valid value
- Logs and time stamp
- Error code
- Shutdown

Exception Undesirable situation that can arrive in the program. There are 2 types:

① Compiler Check - Compiler can't recognise

② Uncheck - Compiler can't recognise
(Runtime Err)

try:

check set of instruction that can cause exception

caught:

handling ^{routine} solution for those exceptions

if they occur

throw:

new input mismatch exception

throws:

① Class signature ② Multiple

final:

- runs compulsory if the exception occurs or not

AI

17-10

Genetic Algorithm

→ Implement real life genetic operations in AI

→ choose the survival one \Rightarrow better solution

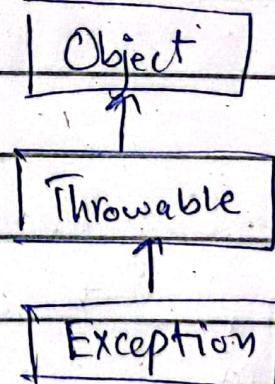
→ Mutation - Swap any random bit

- Evaluation function

Fault Tolerance

Collection of techniques that increase software reliability by detecting error and then recovering from it if possible or containing their effects of recovery if possible.

- ① Backup
- ② Retrying
- ③ Auxiliary code (Time tracking of events)
- ④ Voting Algorithm (Democracy with different scenario)
- ⑤ Replacing (erroneous input with phony input)
- ⑥ Shutdown & Restart



=> Every individual solution is called chromosome

• Fitness function

- Attempt to maximize the function

↳ mathematical function

• Roulette

↳ Highest fitness value = higher probability
~~probability~~

8-Queen Problem

Fitness → Selection → Pairs → Cross-Over - Mutation

CN

19-10

DNS

FAQ: how to map between IP address and name, and vice versa?

FAQ: When to use cname, record, alias and URL

Attacking DNS

SCD

24-10

• Design

- Is a sloppy process
- Is a wicked process
- Is about trade off, priorities and restriction
- Is non-deterministic

- Is non-deterministic & it is a heuristic process] → Error and trial (Improve)

- Is emergent

Characteristic

- Minimal complexity

- Easily ~~un~~maintainable

- Loose coupling

- Extensibility

- Reusability

- High fan-in: a class is used by many other classes

- Low to medium fan-out: a class uses low to medium number of other classes

- Portability

- Learnness

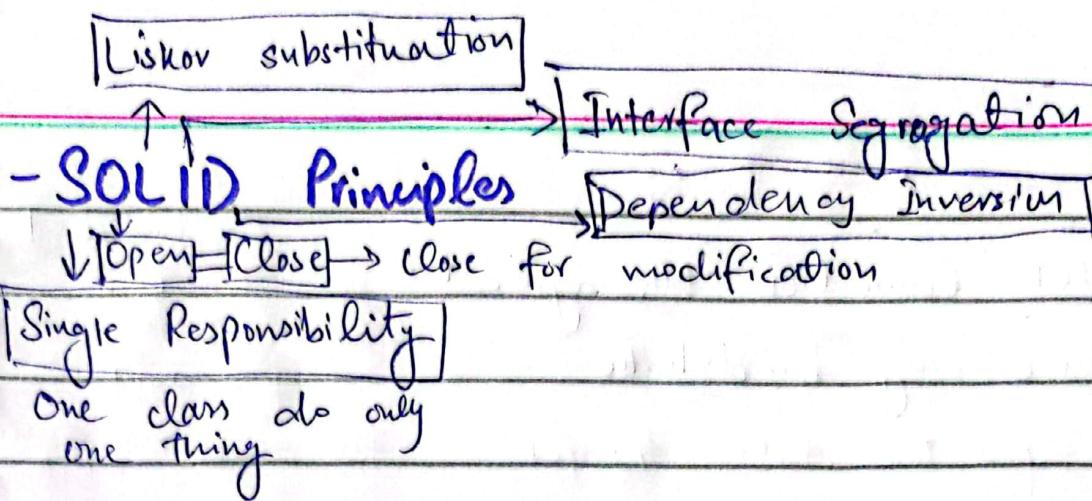
- Stratification (sorted format)

- Keep levels of decomposition stratified so that you can view the system at any single level and get a consistent view

- standard techniques

Principles

- **Abstraction**: hiding implementation details, abstract, interface
- **Encapsulation**: attribute and functions are contained in a single object
 - access \Rightarrow
 - ↳ Information hiding, hiding internal data from other classes
- **Polymorphism**
 - Overloading
 - Overriding
- **Modularity**
 - Breaking down of a component and reassembling it
 - design measure
 - coupling
 - cohesion
 - information hiding
 - separation of concern



Liskov substitution ⇒ Every sub class should be substitution for their parent class

Interface segregation ⇒ A client should never be forced to implement an interface that it doesn't use

Dependency Inversion: high level class should not depend upon low level classes instead both classes use abstraction

AJ

24-10

- ML

- ① Supervised Learning
- ② Unsupervised Learning
- ③ Reinforcement Learning
- ④ Semi-supervised Learning