Address :- Identification of a memory location. [illegible]
8085 µP uses 16 bit address for a memory. [0000 — FFFF]
(2^{16})

Memory word size :- Size of binary information that can be stored in a memory location. For eg - 8085 µP memory word size = 8 bit.
1 memory location has data stored = 8 bit.

µP :- (Microprocessor) - A programmable semi-conductor device which fetches the instruction and data from the memory, decodes and executes the instruction. It is used as a CPU in computers. The basic functional blocks of µP are — (i) ALU (Arithmetic & Logical Unit), an array of registers (1 bit stored in FF is internal memory), and a control unit.
µP is identified by size of ALU. ALU is of 8 bit → µP is 8 bit
(ALU bit = µP bit)

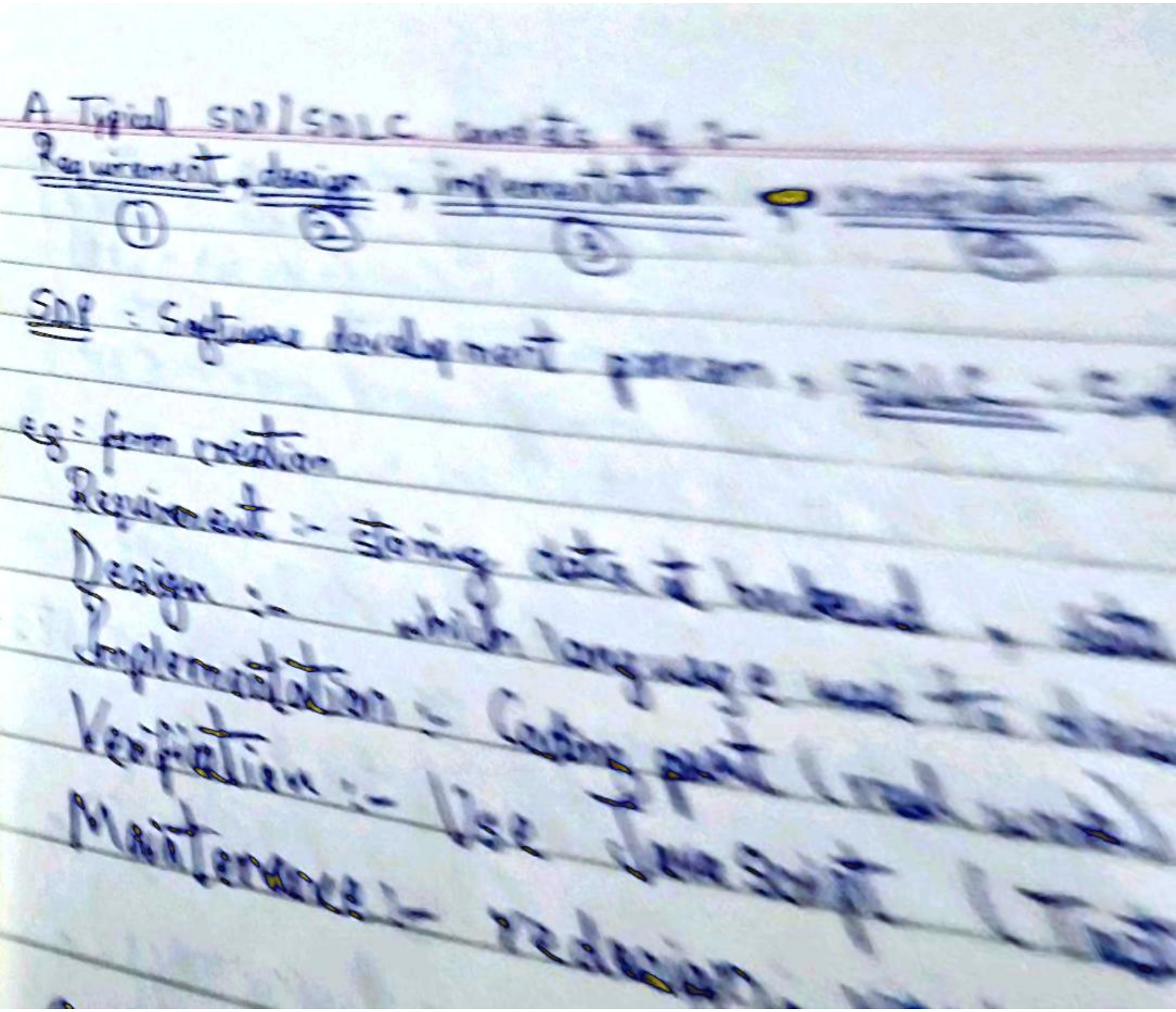
Bus — A bus is a group of conducting lines that carries data, address and control signals. Data bus, Address bus, Control bus
CPU Bus — group of conducting lines directly connected to the µP is called CPU bus. In a CPU bus signals are multiplexed (more than 1 signal can be passed through the same line but at different timings)

SE

Q. What is SE? **Ans** - According to IEEE, SE is the application of systematic, disciplined and quantifiable approach to the development, operation and maintenance of the software.

Ans - SE is a discipline whose aim is the production of fault free software that satisfies the user's need and that is delivered on time and within budget.

Ans - SE is the use of methodologies, tools and techniques to resolve the practical problem that arise in the construction, development, ~~and~~ support and evolution of software.



A Typical SDLC consists of :-
Requirement, design, implementation, ...
1 2 3
SDLC : Software development process, SDLC - Soft
eg : form creation
Requirement :- storing data in textbook, ...
Design :- which language use, ...
Implementation :- Coding part (front end ...
Verification :- Use Java Script (Test ...
Maintenance :- Redesign ...

A Typical SDP/SDLC consists of :-
Requirement (1), design (2), implementation (3), verification (4), maintenance (5)

SDP : Software development procem, SDLC : Software development life cycle.

eg: form creation
Requirement :- storing data at backend, data collection (My SQL)
Design :- which language use to design. (use HTML, CSS)
Implementation :- Coding part (real work)
Verification :- Use Java script (Testing phase – is it storing data or not)
Maintenance :- redesign, version update, debugging.

Software Crisis :- is characterised by the inability to develop software, on time, on budget and within the requirements.
Faitors for SC :-
- Over budget, over time project.
- Lack of communication b/w the software developers.
- Increase in cost of software compare to hardware.
- Increase in size of the software.
- Lack of understanding the requirement.
- Duplication of efforts.
- Absence of Automation (eg. Git, GitHub etc)
- Difficulty in maintaining the code

CD

(1°) Grammar	4 Tuples	} Finite Grammar
(2°) Regular Grammar (RG)	4 Tuples	
(3°) Content Free Grammar (CFG)	4 Tuples	
(4) DFA / NDFA (Finite Automata)	5 Tuples	
(5) Machines (Moore Machine)	6 Tuples.	
(6) Content Sensitive Grammar (CSG)	7 Tuples.	} Infinite
(7°) Push Down Automata (PDA)	7 Tuples	} Infinite
(8°) Turing Machine (TM)	7 Tuples	

SE

What is SDLC ?

Software Development life cycle or a software procen also known as software methodology is a set of related activities that leads to the production of software. Those activities may involve the development of software from scratch or modifying an existing system.

The steps are -

(1) Feasibility Study :-
→ Find abstract definition of the problem.
→ Checking the financial and technical feasibility
→ Analysis of cost.
→ Checking availability of infrastructure & human resources.
→ Alternative solution strategy.

(2) Requirement, Analysis & Specification :-
→ Try to understand the exact and complete requirement of customer and document them properly.
→ Try to collect and analyze all data related to the project.
→ Large document will be written in natural language which will describe what the system will do without describing it how. called SRS [Software Requirement Specification].

(3) Designing :-
→ We transform the requirement into a structure that is suitable for implementation of the code in a specific programming language.
→ In this we prepare a document called SDD [Software Design Description] which will describe how the system perform funtionally Overall architecture & algo strategy are chosen in this phase.

(4) Coding / Implementation :-
→ Translate the design of the system into code of programming language. They should be readable and maintainable code.

(5) Testing :- (Black box and white box)
→ Software testing is a procen of executing a program with the intention of finding faults / bugs in the code.

(6) Deployment :-
→ Software is installed on the user system side and training of software

Microcontroller : All chips components are internal inside IC.

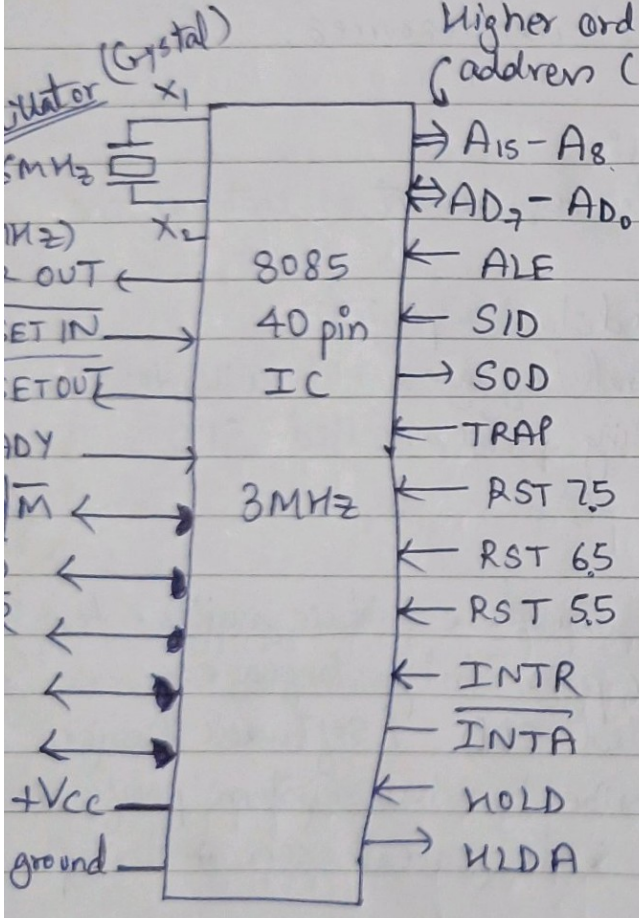
MP : Components are seperate in µP.

and hardware requirements are done/checked.

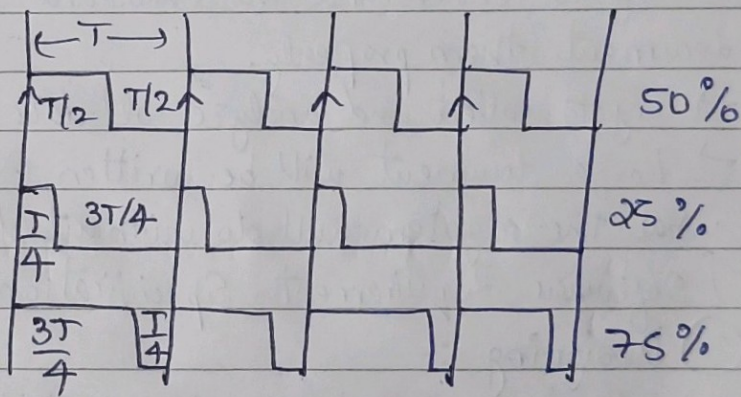
(70) Maintainance :- Any change made in the software after its initial release is called maintainance. It should be because of adaptive, corrective maintainance.

MP I

8085 µP Pin Diagram / Assignments.



(Crystal oscillator)
X1 X2 (6MHz)

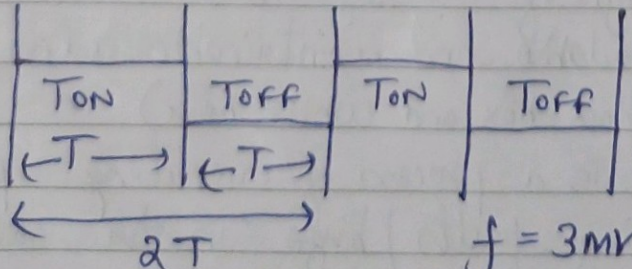


Duty Cycle :- $\frac{T_{ON}}{T_{ON}+T_{OFF}}$

$\Rightarrow \frac{T/2}{T/2+T/2} = 50\%$

ing T FF :-

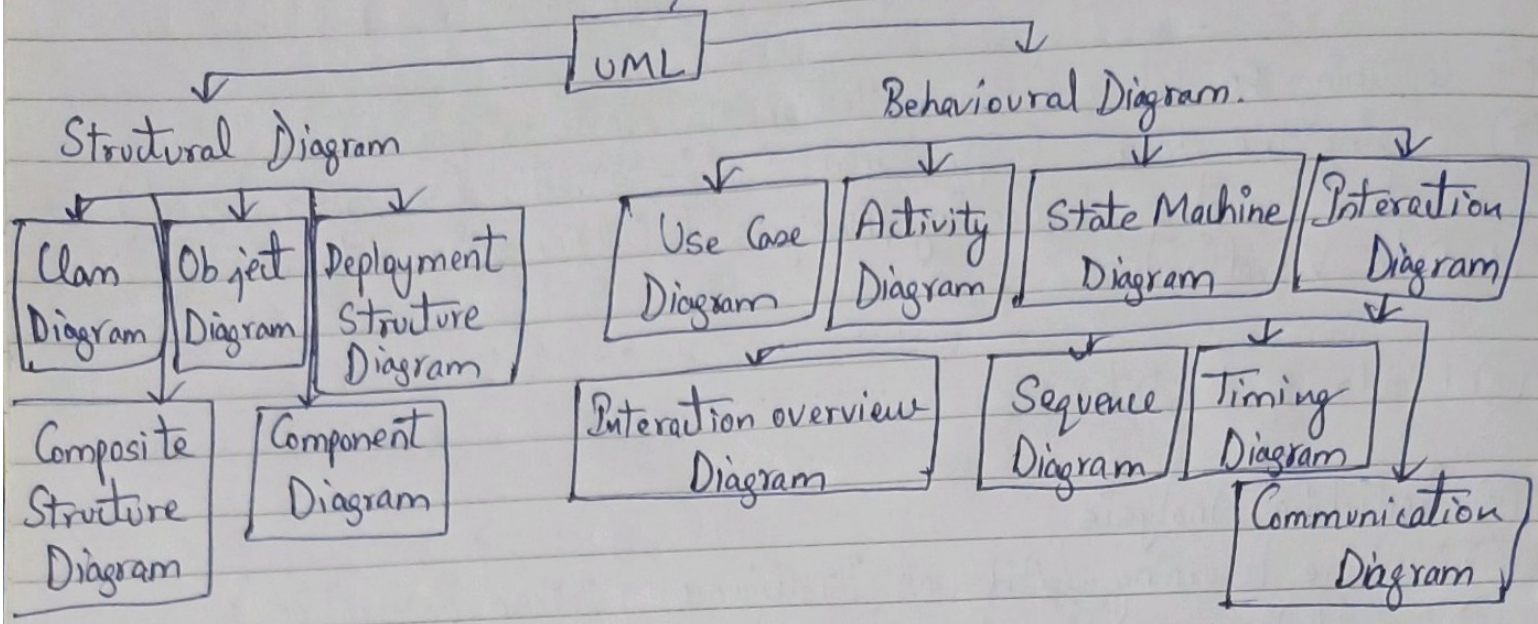
35 works on
% duty cycle.
convert into
. duty cycle by
g T-ff. (T=1 => FF will toggle)



Duty cycle $= \frac{T}{T+T} = 50\%$

depict system behaviour

SE → Unified Modelling Language

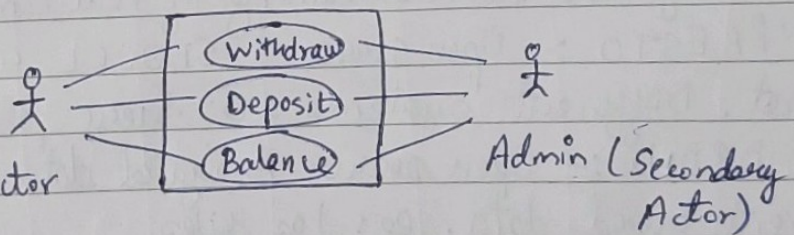


UML :-
Modelling language in the field of SE that is intended to provided standard way to visualize the design of a system.

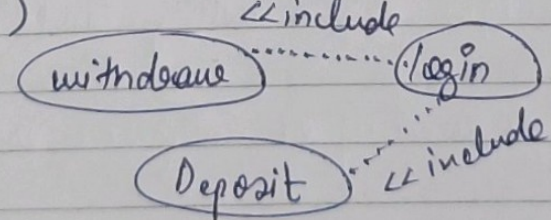
Use Case Diagram :- model the behaviour of a system, use to illustrate the funtional requirement of the system. and its interaction with the external agents / Actors. like customer etc. It gives the high level view of the system without going into implementation details.

Components of Use Case Diagram :-

→ Actor :- (☺)
→ Use Case :- (⬭)



→ Generalization (⟶▷)
→ Include :- (..... << include)
→ Extend :- (<< ..... extend)



→ Association Link :- (—)
→ System boundary :- (▢)

* Actor :- An external agent / Actor lies outside the system model but interact with the system in some way. It may be information

system, machine or person that
Customers, users, external device or any external entity interacting with the system treated as Actors.

MPI

8085 μP Addressing Modes

Instruction — operand
— opcode

MOV B, C
opcode → internal registers of μP

operand is a number which can be data or address.
opcode is operation code.

MVI B, 20H
opcode → immediate data
operand.

Register :- General purpose :- A, B, C, D, E, H, L.
Special purpose :- SP (Stack pointer : address of top of stack)
F (Flag), PC (program counter) (address of next ~~iteration~~ instruction to be fetched)

Addressing Mode :- The manner in which an operand is given in an instruction.

(10) Immediate Addressing Mode :- data is directly given in the instruction.

eg :- MVI B, 25H, ~~MOV B, C~~

1 byte 1 byte
2 byte = 16 bits

F - 15
10 - 16.
11 - 17
12 - 18.

25H
+ 3AH
515H
5FH
~~38H~~ 38H
~~3AH~~ 3AH
~~72H~~ 72H

Number
Unsigned ($0 - 2^n - 1$) Signed (-2^{n-1} to $2^{n-1} - 1$)
(follows 2's complement)

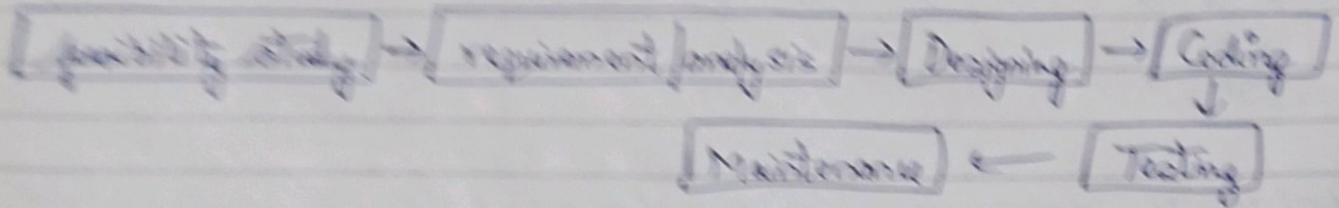
Drawback :- big size / memory Advantage: No need to search memory
Data already present.

Subjects	Unsubjects
(1) More than one LDT/RDT	(1) Only one LDT/RDT
(2) Different duration of DT	(2) Same duration of DT.

SE

Life Cycle Models :- Different types of SDLC Model -

Waterfall Model :- developed by the Winston W. Royce in 1970. Inspired by manufacturing and construction process, where each step relies on the completion of previous step.



→ It is the simplest SDLC model in which process are organised in a linear and sequential manner. This type of model is basically used in small to medium size projects with clear and well defined requirements.

Disadvantage :- Working version is produced in the last level, not good for large level projects. High amount of risk & uncertainty. Not suitable to accomodate any change.

Advantage :- Easy to implement and understand with well defined stages and clear milestones. Each level has well defined input and output for each stage. Low cost and easy to schedule as all the staff don't work concurrently on the same project so they can work on different projects.

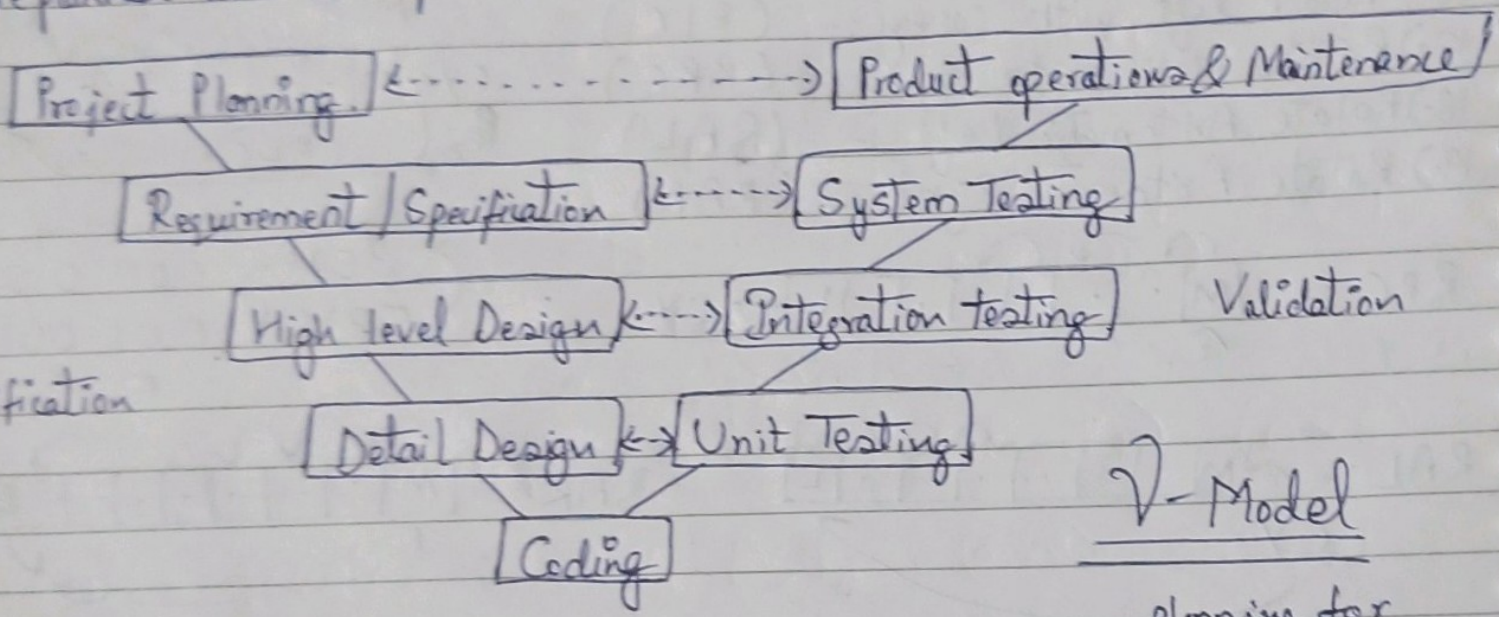
V- Model

Variant of the Waterfall Model. It emphasise on validation and verification
Verification :- are you building it right. Validation :- have you build it right

↓(Developer)
Review, inspection, walkthrough, unit-testing integration testing. (Static + Dynamic activity) only checking

↓(Tester)
involves system testing
(Dynamic activity → code is running, whole product testing)

Verification & Validation (V&V) activities are spread over the entire lifestyle of the V-Model. In every phase of development, testing activities are planned with development.



rification

V- Model

planning for

v :- Each phase is made testable. Easy to use. Emphasise ~~on~~ validation & fication of the software starting from earlier phase of V-Model.

v :- Does not support overlapping of the phases. Does not easily accomodate r changes to the requirements.

n to use :-
quirements are known upfront as well as tools and technologies known.

SE

Activity Diagram → Behaviour Diagram { Make flow chart }

(*) Advanced version of flowchart.

(1) Start state (2) Final state (3) Activity { ▭ } operational process of the system.
↓ begin process ↓ end process

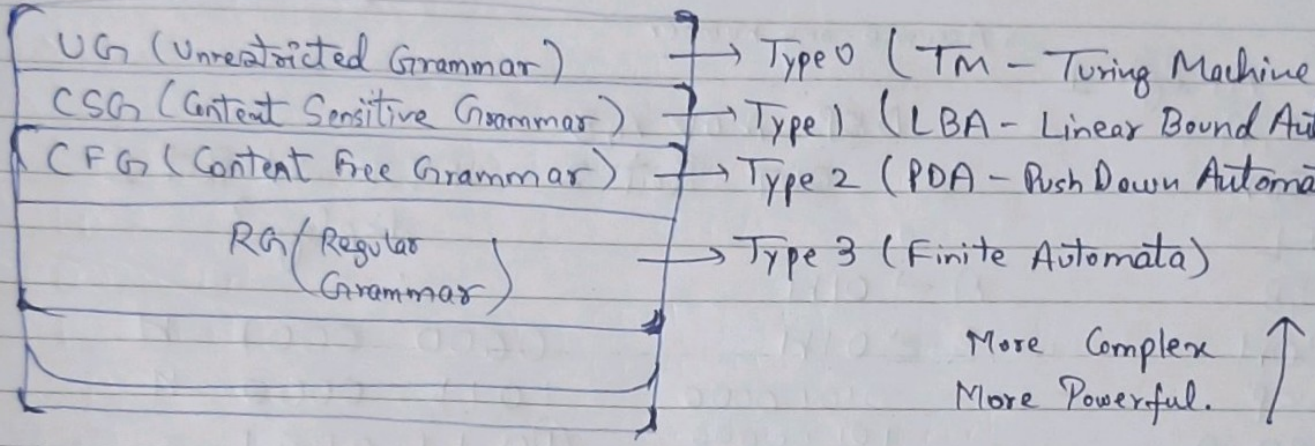
(4) Decision node {◇} (Diamond) use to encounter a decision within any activity node. Two outcomes → Yes or No.

(5) Fork and Merge :- Fork is used to split a process into subprocesses. Merge is used to combine subprocesses into a single process.

(*) Extension of workflow diagram.

CD

Chomski :-



(*) Type 0 : Unrestricted Grammar. (TM) (Recursive Language) eg :- A → AAAA, AA → A, AA → AA } no limit on terminal or NT alphabets

(*) Type 1 : Content Sensitive Grammar (LBA)
eg :- Infinite length (L < Right side value)
AA → AAA , A → AA.

(*) Type 2 : Content Free Grammar (PDA) (finite length)
eg :- A → aA, A → Aa, A → AA, A → aa.

(*) Type 3 : Regular Grammar (FA) eg : A → a, A → aA, A → Aa, A → aa.

Input Stream → [a.out] → Sequence of Token

YACC :- Yacc.y → [YACC Compiler] → Yacc.Tab.c

Yacc.Tab.c → [Compiler] → a.out

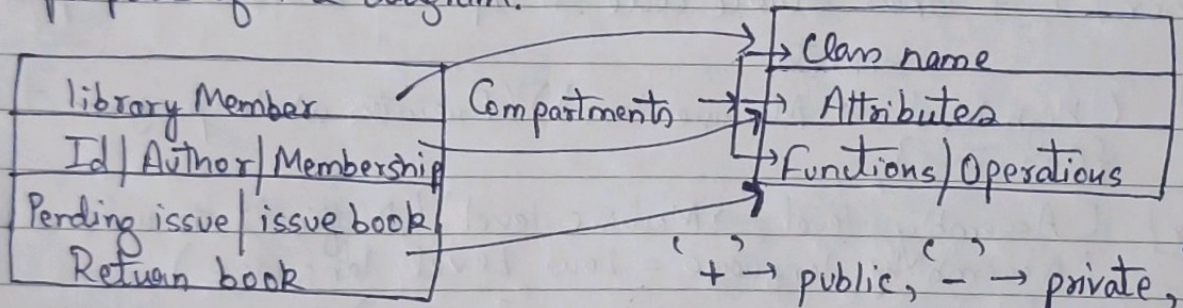
Input Stream → [a.out] → Sequence of Parser.

PP (Assignment) Q. 5) What is string. Explain any 5 string library fuc. with eg. used in Python. (Wednesday Submition)

SE

Class Diagram :- is a structural diagram. Classes are the entities with common features that is attributes & the operation. Represented as a solid outline rectangle with compartments. Compartments for name, attributes, and operation. Attributes and operation compartments are optional depending upon the purpose of the diagram.

eg :-

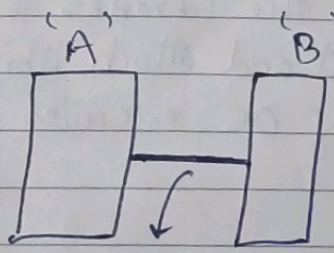


Class Diagram

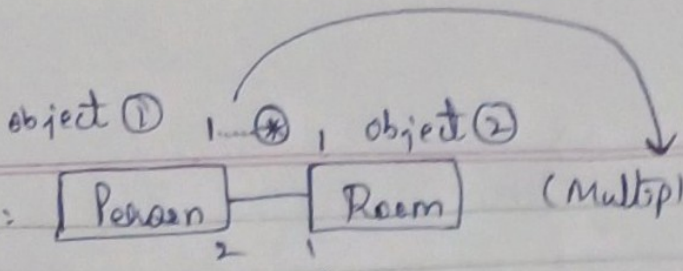
('+' → public, '-' → private, '#' - protected.

Relationships :-

1. Association
2. Aggregation & Composition
3. Inheritance



solid line (Unidirection) A can call B & B can call A but if '→' A can call B but not vice versa.



eg: (Multiplicity)

(2 person in one room) or (many person can accomodate one room)

② Aggregation : 'A' has instance of 'B'. (A ◇→ B)
'B' can exist without 'A'. Eg: Parent and child., tyre can exist without car.

→ Composition : ('A' ◆→ 'B')
'B' cannot exist without 'A' eg :-

③ Inheritance :- ('A' is a ▷ 'B') eg : (Base) Shape, Rectangle, Triangle, ...

BD

① Hadoop Pipes ② Hadoop Streaming. ③ Streaming Data Flow

CD

Relation b/w Compiler, Assembler, Linker & Loader :-

Pre Processor (It takes source code as input & share to next phase)
↓
Compiler (High level → Machine level) (converts) (Executes whole program at a time)
{C++, Java}
↓
Assembler (Assembly level → Machine level) (Difficult to understand)
(Assembly language = Low Level language)
↓
Linker (links the libraries to the source code) (it takes machine code as input and attach library files with it & share to next phase)
↓
Loader (part of OS, executes the program) (loads the whole code & run it)
↓
Memory

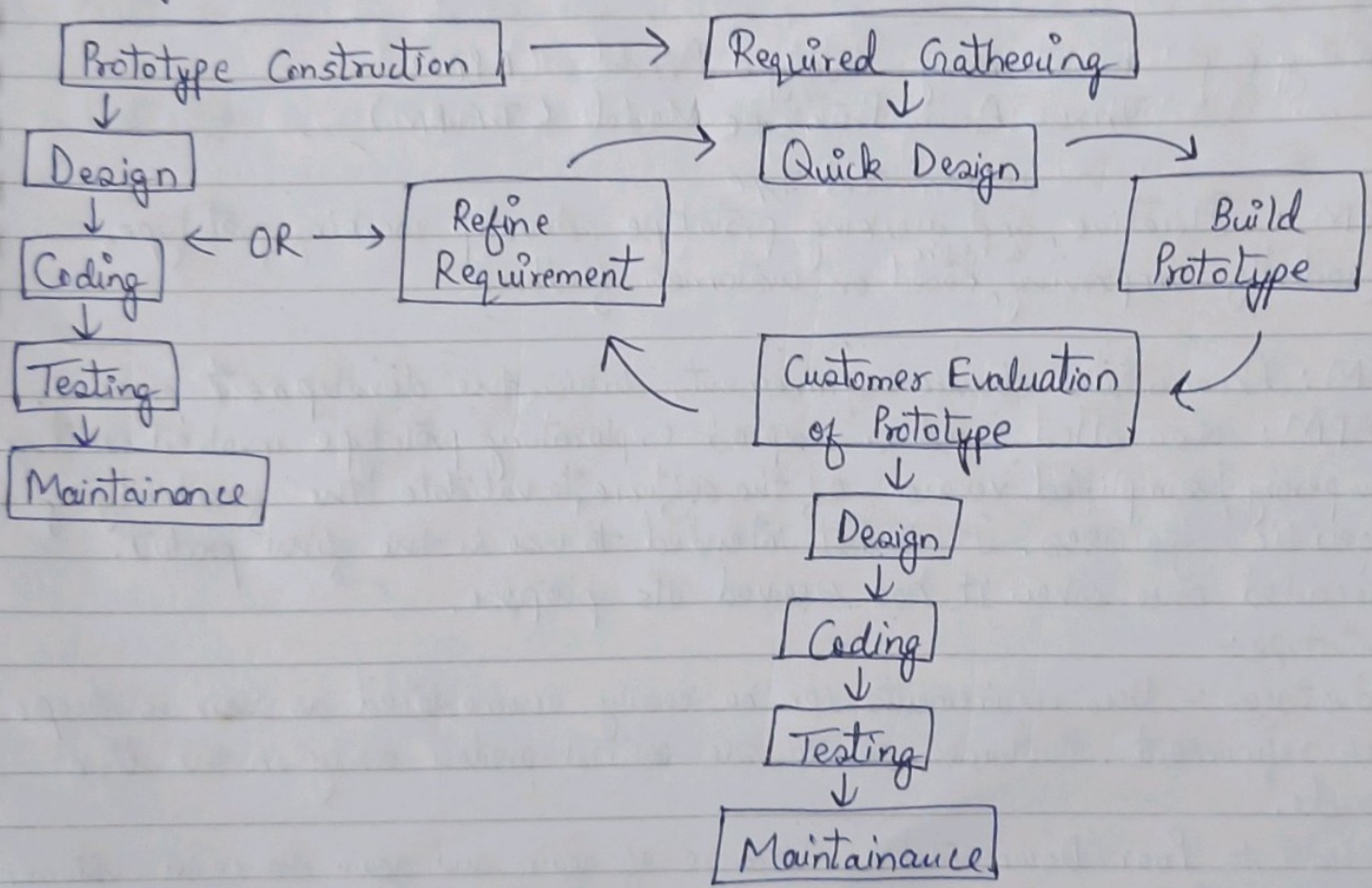
⑩ Lexical Phase : First phase of compiler. It takes source code as input and generate token as output. LEX tool is used.

(2) Syntax Phase : Second phase of compiler. It takes token as input and Parse Tree as output. YACC tool is used.
(3) Semantic Phase : Third phase of compiler. Semantic error. Type checking → Static Type checking (Compile Time) Eg: C, C++, Java.
→ Dynamic Type checking (RunTime) Eg: Perl, Ruby, Python

SE

(*) Evolutionary Process Model :- are iterative model. They are characterised in a manner that enable you to develop increasing more complete version of software.
→ Prototype Model :-
→ Spiral Model :-

(1) Prototype Model :-



CD

④ Intermediate Code Generator :- 4th phase of compiler. takes machine code as input and generate intermediate code as output.
⑤ Code Generator :- 5th phase of compiler. Takes intermediate code as input and generate code as output.
⑥ Code Optimization :- Best phase of compiler. Reduce complexity [→ Space, → Time.

Types of Compiler :-
① Multi Pass Compiler : it is also called 2 or 3 compiler. it is step by step execution. Ex: COBOL, Algol, Fortran.
② Source to Source Compiler : it takes input as high level language and output as high level language. Ex: C# Compiler, C++ Compiler, Java compiler, COBOL Compiler, ALGOL compiler

SE

Prototype [→ Evolutionary Prototype Model (EPM)
[→ Throw Away Prototype Model (TAPM)
/ Incremental type

EPM : Iterative, and working prototype, develop working prototype, gradually improving & refine based on customer feedback

PM : Accomodates change requirements during the development process.
APM : also called rapid or ~~explod~~ exploratory prototype involved creating emperory / simplified version of the software, to validate the feasibility of pecific features. It is not intended to use in the final product. iscarded once when it has served its purpose.
rototype :-
dvantages :- Uses requirements can be easily accomodated as their is scope r refinement. Customer get to see partial product early in the life ycle.
isadv :- Poor Documentation because of again and again change in customer ostly, Time consuming, Uncertainty in no. of requirement. stomer may loose interest by iterations.

(Meta Model)

Spiral Model :- Use for risk identification. Made by Barry Boehm.

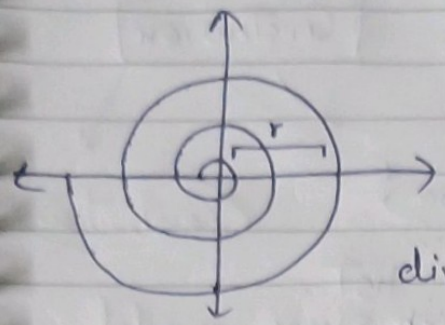
→ ~~Each~~ Incoporate project risk factor in software development.

Bigger the spiral, bigger the cost.

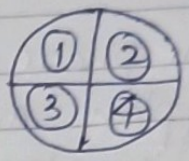
Each path around the spiral is indicative of the increased cost.

Radial dimension of the project represent the cumulative cost.

(r)



The angular dimension represent the progress made ~~rate~~ in completing each cycle. Each loop of the spiral from the x-axis through 360° represent one (1) phase. 1 phase is roughly divided into 4 sectors.



1 phase

① Planning ② Risk Analysis ③ Development ④ Assessment	During the 1st phase, planning is performed, risk are analysed, prototype are build and customer evaluate the prototype.

Spiral Model is the combination of Waterfall, Prototype, iterative.

During 2nd phase, a more refine prototype is build, validatedation requirement are documented, customer is involved in assessing the new prototype.

By the 3rd time, risks are known and is somewhat more traditional method of development. Focus is on identification & elimination of problem / high risk before they threaten software.

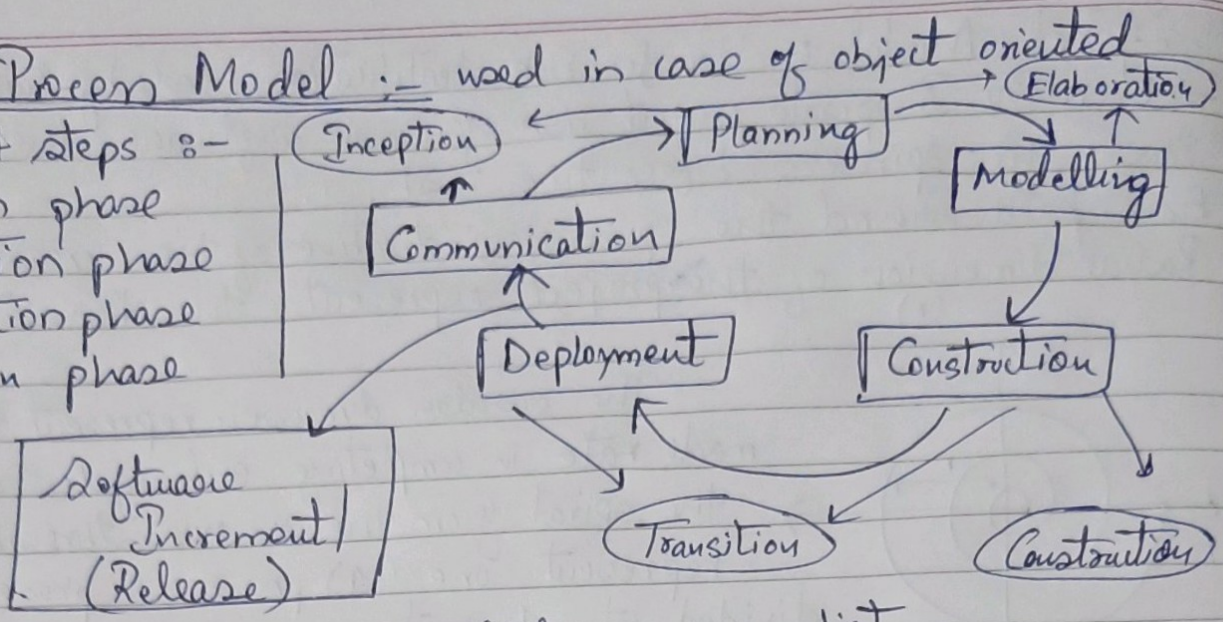
Adv:- Good for large and complex project, Risk Handling, Flexibility in requirements, software prototype is produced earlier in life cycle. suitable for high risk projects, where business needs maybe unstable.

Disadv: Time + cost consuming, more complex than other models, excessive documentation, too much dependency on risk analysis.

(Object oriented vs. Structural model.) vs (Procedural mode)

Unified Process Model :- used in case of object oriented system. 4 steps :-

1. Inception phase
2. Elaboration phase
3. Construction phase
4. Transition phase



- Inception phase :- initial business case / use case / risk list, phase model, project plan and prototypes.
- Elaboration phase :- detail, use case model, revised risk line, analysis model.
- Construction phase :- design model, test plan, procedures, test case, installation manual / user manual.
- Transition phase :- Software increment, new release, beta testing, user feedback