Lab....

DATE: __/__/__

Class Diagram-

Class diagram in UML is type of static structure diagram that describes the structure of system by showing system classes, attributes, operations and relationship among objects. Class diagram is not only used for visualizing, describing and documenting different aspects of the system but also ~~continuously~~ constructing executable part of software applications.

Representation of class -

It is represented with rectangle.

Class name
Attributes
Operations

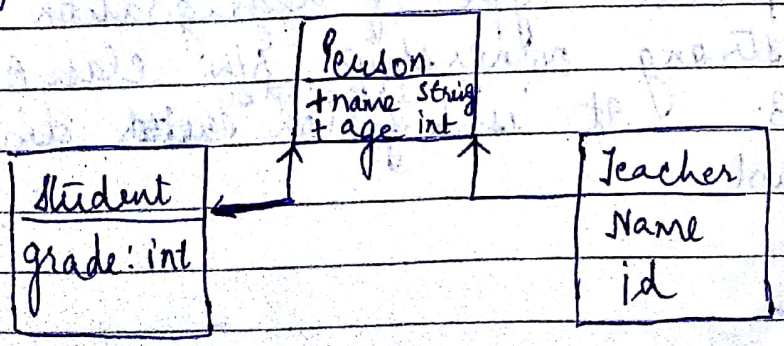
Eg -

Customer
+name : string +location : string
sendsorder recieveorder

visibility mode → + Public
- Private
Protected

Relationship-

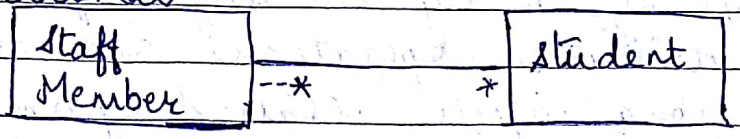
1. Generalization → Inheritance or 'is-a' relationship



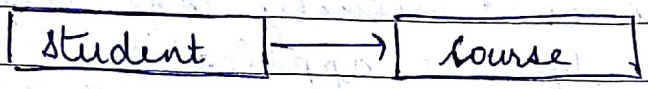
Be Positive.....

2. Association

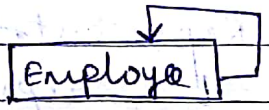
→ Bidirectional - When both classes know about each other, its association is called bidirectional.



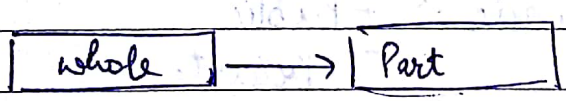
→ Unidirectional - One class know about other.



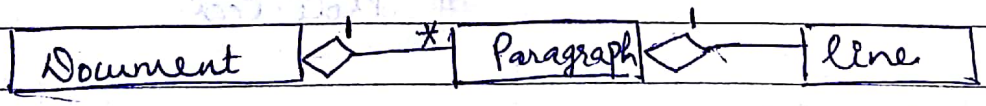
→ Reflexive -
Self relationship.



3. Aggregation:
In this whole class plays more imp. role than the part class but two class are not dependent on each other.

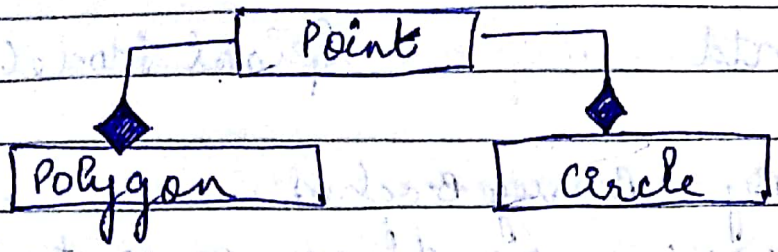


Eg-

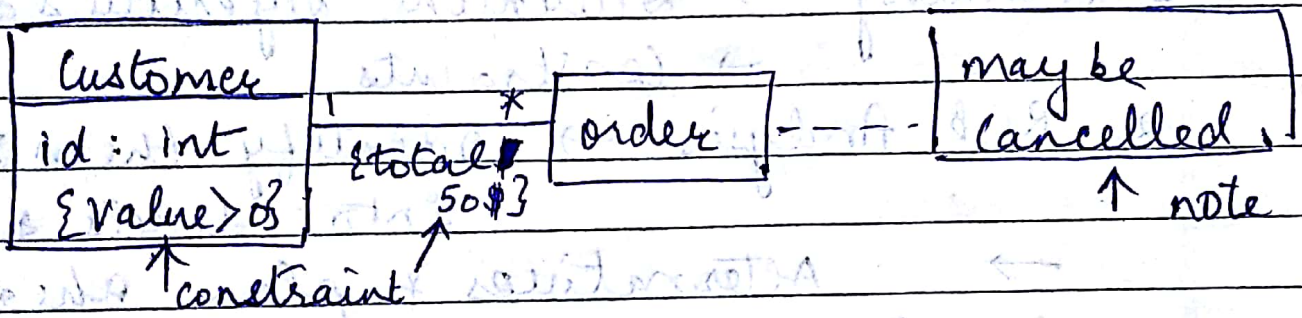


4. Composition
It is special type of aggregation that own a strong ownership b/w class A and class B. It is represented by filled with diamond symbol.

DATE: ___/___/_____



Constraint, rules, notes -



DATE: __/__/____

Models Contd.... Spiral Model

- Developed By: Barry Bochm
- Main features: Handling uncertainty / Risk
- Phases → Multiple phases having 4 activities which are-

1. Planning → Determine objective & alternatives
   → constraints
2. Risk Analysis → Identify them & classify into levels & resolve
   → Alternatives & plan ahead
3. Development & Testing
4. Assessment → Customer Evaluations

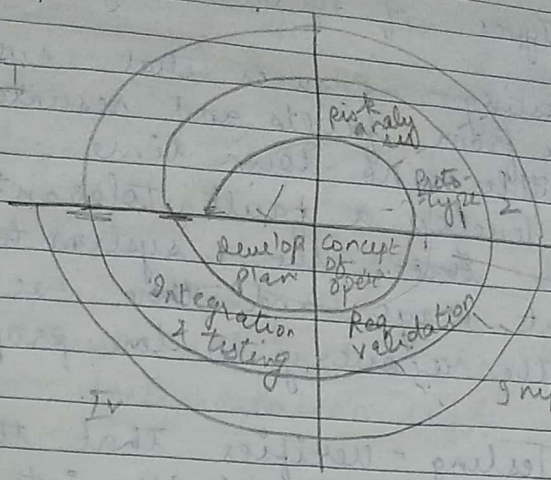
Phase-1 — Mainly Prototype is built
Phase-2 — Prototype is refined & req. document and validated by customer
final prototype is made
Phase3 — Risk are almost known.

- Focus Areas — Identify problem → classify into diff. levels acc to risk → eliminate before they harm sln
  Continuous validation is there

Disadvantag — Expertise req. to handle risk carrying spiral model

Be Positive.....

DATE: __/__/__



Software Characteristics

- Functionality - degree of performance of the software against its intend purpose
- Reliability - ability of the software to provide desired functionality under given conditions.
- Usability - Refers to the extent to which the software can be used with ease.
- Efficiency - Refers to ability of the software to use system resources in most & effective efficient way.
- Maintainability - Refers to the ease with which the modifications can be made in S/W
- Portability - ease with a software developer transfer S/W from one platform to another.

Be Positive.....

DATE: __/__/____

Types of System Testing

→ Recovery Testing – Ensures that system must recover from faults and resume processing with little or no down time
Aim to develop a fault tolerant system.
method – Force the system to fail in different ways and then verify that the recovery is done properly.

→ Security Testing – verifies that the position protection mechanism built into the system with actually protect it from attack.
Tester → acts as intruder
→ ensure that cost of attack is higher than the info. achieved.

→ Stress Testing – It executes the system in a manner that demands resources in abnormal quantity.
– excessive memory req.
– db req.
– simultaneous req.

→ Sensitivity testing – variation of stress testing. attempts to uncover data combinations within the valid input domain, which causes improper functioning

DATE: __/__/____

→ Performance Testing – aimed at testing the runtime performance of the S/w
ensure that performance is maintained

→ Deployment Testing – known as configuration testing
it tests the software in each environment in which it is to operate
examines all installation procedures, documentation rqd

Lab

DATE: __/__/____

Interaction Diagram

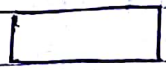
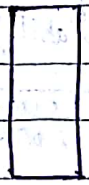
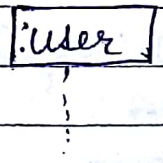
1. Sequence diagram are popular dynamic modelling sol. in UML bcoz they particularly focus on lifelines or the processes and objects that lie simultaneously and the message exchange blw them to perform a function before lifeline ends. A seq. diagram is a type of interaction diagram it describes how & in what order a group of objects work together.

Benefits -

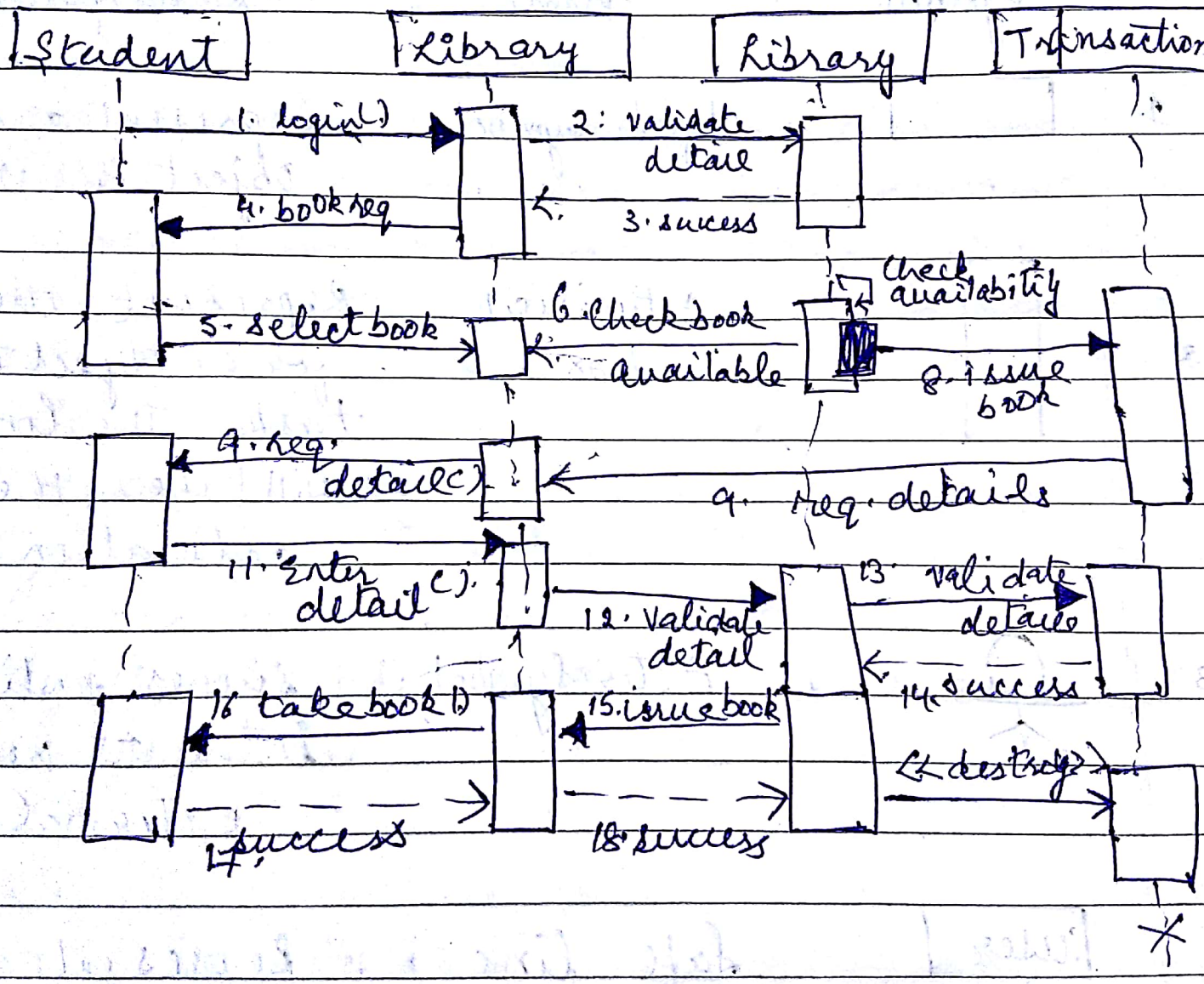
- Represents details of UML usecase
- Model the logic of sophisticated procedures, function or operation.
- See how obj. & component interact wd each other to complete a process.
- Plan and understand the detail functionality of an existing or future scenario

Basic Symbol & Components

DATE: __/__/____

	Symbol	Name	Description
1.		Object symbol	Represent a class or object in UML
2.		Activation box	Represent the time need for an object to comp. a task. The longer the task will take, the longer the activation box become
3.		Actor Symbol	Shows entities that interact with ore external to the systems
4		Safe line Symbol	Represent the passage of time as it extend downward
5		Synershonous message Symbol	Represent by a solid line wd a solid arrow. Used when sender must wait for a response to a msg. befor it continues
6.		Asynchronous msg. Symbol	Represented by a solid line with a lined arrow head. It does not require a response before the sender continu
7.		Return msg Symbol	message rply to calls.

DATE: ___/___/_____



ST-2

DATE: __/__/____

Integration Testing

1. Big Bang Approach.
2. Incremental Approach
   - Top down
   - Bottom up
   - Hybrid.

Top down Integration -

1. The main control module is used as test driver and stubs are substituted for all components directly subordinate to the main control module.
2. Depending on integration approach (selective) (BFS) (DFS) subordinate stubs are replaced one at time with actual Component. Test are
3. Test are conducted as components are integrated for on completion of each set of test Another stub is replaced with real components
4. Regression testing may be conducted to ensure that new errors have not been introduced

DATE: __/__/____

Advantages -

- Fault localization is easier.
- Possible to obtain early prototype.
- Critical module on priority
- Major design fault or error could be found and fixed first

Disadvantages -

- need many stubs.
- Module at lower level are tested inefficiently

Bottom-up integration

1. Low level component are combined into clusters that perform specific s/w stub fxn.
2. A driver is written to coordinate i/p & o/p of test cases
3. A cluster is tested
4. Drivers are removed & clusters are combined moving upward in program structure.

Advantage -

- Fault localization is easy
- no tym is wasted waiting for all module unlike bigbang approach

Be Positive.....

DATE: __/__/____

Disadvantages -
critical module (at top level of S/W architecture) which control the flow of app. are tested in labs & may be prone to default.

Hybrid → combination of both approaches

DATE: __/__/__

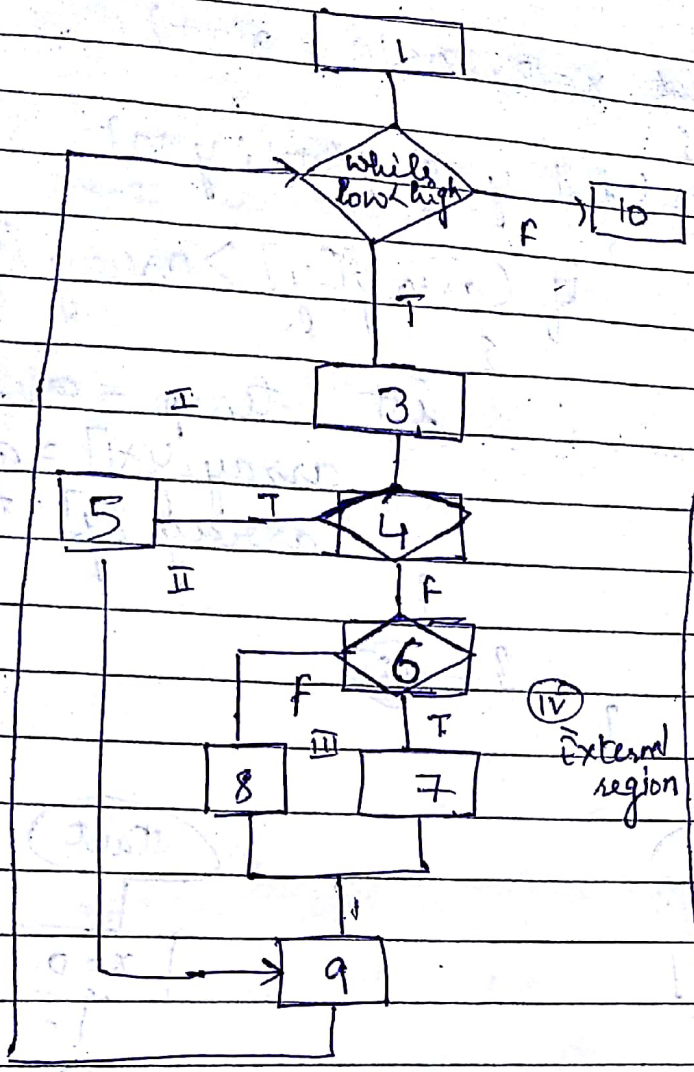
White Box Testing

Cyclomatic Complexity

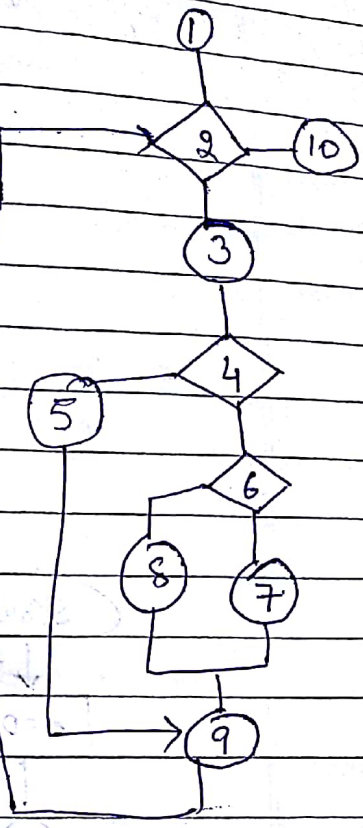
Eg- Binary Search

```
int binSearch (int x, int v[], int n)
{
    int low, high, mid;                    ]
    low = 0                                ] ①
    high = n-1;                            ]
    while (low <= high)                    -②
    {
        mid = (low+high)/2;                -③
        if (x < v[mid]);                   -④
            high = m-1;                    -⑤
        else if (x > v[mid])               -⑥
            low = mid+1;                   -⑦
        else                               -⑧
            return mid;                    -⑨
    }                                      -⑩
    return -1;
}
```

DATE: __/__/__



Control flow graph



Independent path

1 - 1-2-10
2 - 1-2-3-4-5-9-2-10
3 - 1-2-3-4-6-7-9-2-10
4 - 1-2-3-4-6-8-9-2-10

Calculate Cyclomatic complexity

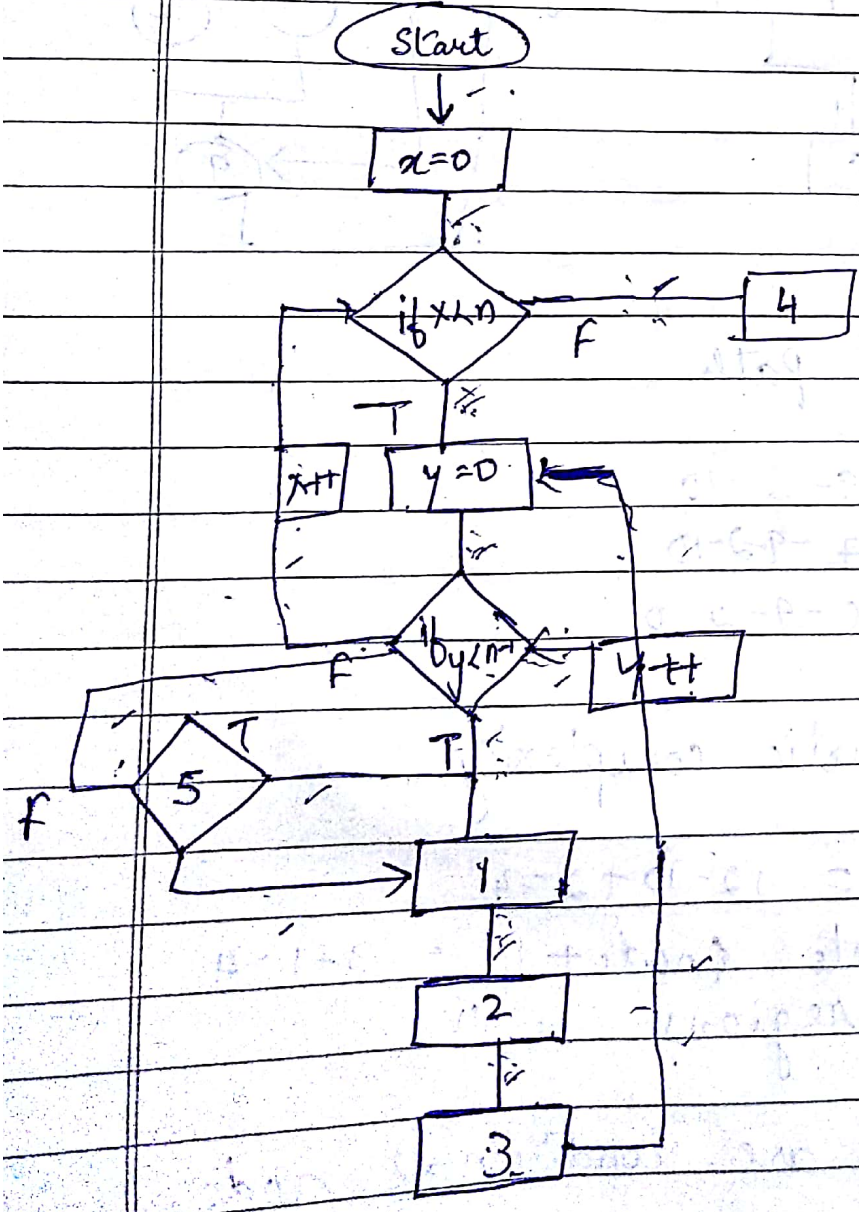
$V(G) = E - N + 2 = 12 - 10 + 2 = 4$
$V(G) =$ Predicate node $+ 1 = 3 + 1 = 4$
$V(G) =$ no. of regions $= IV$

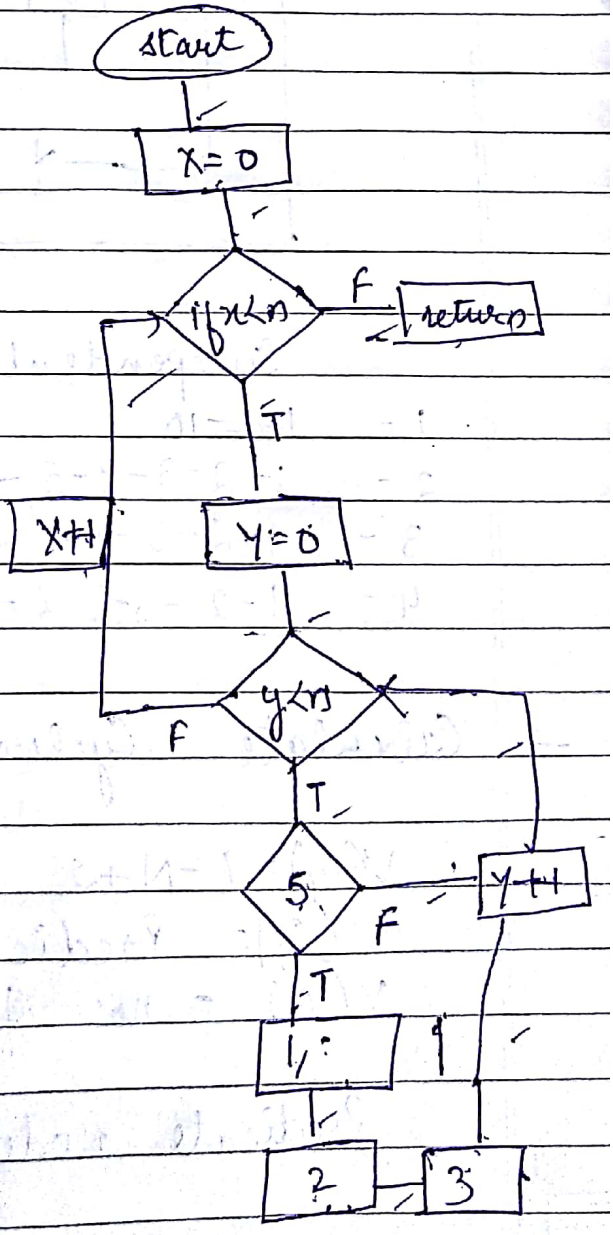
Predicate nodes are conditional nodes.

Be Positive.....

DATE: ___/___/___

```
for (int x=0; x<n ; x++)
{
for( int y=0; y<n-1; y++)
{
if (array[y] > array[y+1] )  -5
{
int temp = array[y+1]  -1
array[y+1] = array[y]  -2
array[y] = temp;  -3
}
}  -4
}
```





Be Positive.....

$V(G) = E - N + 2$

$= 13 - 11 + 2$

$= 4.$

$V(G) = 3 + 1 = 4$

$V(G) = 4.$

DATE: __/__/____

Functional Testing

It is only testing which involve on observation of O/P for certain I/P value. There is no attempt to analyse the code which produce O/P ignoring the internal structure of software.

Boundary Value

x (0-100)

0, 1, 50, 99, 100

(x, y)

(0, 50) (1, 50) (99, 50) (100, 50) (50, 50)
(50, 0) (50, 1) (50, 99) (50, 100).

no of boundary test cases = 4n+1

Q Consider a prgrm for determination of nature of roots of quadratic equation. Its I/P is tripple of integers (a, b, c) & values may be from interval (0, 100) The program O/P may have following work know the Q.E imaginary roots, real roots, equal roots. Design boundary value test cases.

→ (a, b, c) — (0 -100)

ax^2+bx+c

$b^2-4ac = 0$ equal roots

$b^2-4ac > 0$ real

$b^2-4ac < 0$ imaginary

B+

Be Positive.....

DATE: __/__/__

a=0 ⇒ not a quad eqn.

Test Case	a	b	c	O/P
1	0	50	50	not a good eqn
2	1	50	50	Real roots
3	50	50	50	Imag roots
4	99	50	50	real
5	100	50	50	
6	50	0	50	imag
7	50	1	50	imag
8	50	99	50	
9	50	100	50	
10	50	50	0	
11	50	50	1	
12	50	50	99	
13	50	50	100	

8 Consider a program to classify a triangle its I/P is a triple of the integers (x, y, z) And the data type for I/P parameters ? Show that there will be integers greater than 0 & less than eq. Zero. The O/P will be one of the following words.

Scalene, isosceles, equilateral, nos.

design boundary test cases.

DATE: __/__/____

Testcases	a	b	c	o/p
	1	50	50	equi isoscelless
	2	50	50	equi isos
	99	50	50	equi isos
	100	50	50	not a triangle
	50	50	50	equi
	50	1	50	isos
	50	2	50	isos
	50	99	50	isos
	50	100	50	not a triangle
	50	50	1	isos
	50	50	2	isos
	50	50	99	isosceles
	50	50	100	not a triangle

Robustness Testing

RC is extension of boundary value analysis. Here we see what happens when the extreme values are exceeded. Wd a value slightly greater than the max. Slightly less than then the value min. It we means we make test cases outside the valid boundary of i/p domain.

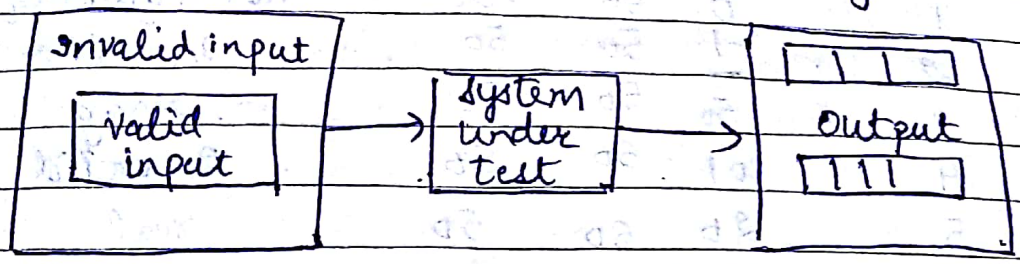
total no. of test cases - 6n+1

(0 - 100)

-1, 0, 1, 50, 99, 100, 101

DATE: / /

Equivalence Class Testing



Output domain

O1 = {<a,b,c> : Not a quad. eqn if a=0}
O2 = {<a,b,c> : Real roots if $(b^2-4ac)>0$}
O3 = {<a,b,c> : Imag. roots if $(b^2-4ac)<0$}
O4 = {<a,b,c> : Equal roots if $(b^2-4ac)=0$}

TestCase	a	b	c	Expected O/p
1	0	50	50	not aquad
2	1	50	50	Real roots
3	50	50	50	Imag roots
4	50	100	50	Equal roots

Test Case Based on Input domain

$I_1 = \{a : a = 0\}$
$I_2 = \{a : a < 0\}$
$I_3 = \{a : 0 < a \leq 100\}$
$I_4 = \{a > 100\}$
$I_5 = \{b : 0 \leq b \leq 100\}$
$I_6 = \{b : b < 0\}$
$I_7 = \{b : b > 100\}$
$I_8 = \{c : a \leq c \leq 100\}$
$I_9 = \{c : c < 0\}$
$I_{10} = \{c > 100\}$

B+

Be Positive.....

DATE: __/__/__

Test case	a	b	c	Expected o/p
1	0	50	50	not a quad eqn
2	-1	50	50	Invalid
3	50	50	50	Imag
4	101	50	50	Invalid
5	20	50	50	real
6	50	-1	50	Invalid
7	50	101	50	Invalid
8	50	50	30	real
9	50	50	-1	Invalid
10	50	30	100	Imag

⇒ Triangle

$O_1 = \{\langle a,b,c\rangle : \text{Scalene} \text{ if } (a!=b!=c)\}$

$O_2 = \{\langle a,b,c\rangle : \text{equi} \text{ if } (a==b==c)\}$

$O_3 = \{\langle a,b,c\rangle : \text{iso} \text{ if } (a==b!=c)\}$

$O_4 = \{\langle a,b,c\rangle : \text{not triangle if } (a+b>c)\}$

Test case	a	b	c	Expected
1	0	50	50	
2	1	50	50	
3	99	50	50	
4	98	50	50	
5	100	50	50	
6	50	0	50	
7	50	1	50	
8	50	99	50	
9	50	100	50	
10	50	50	0	
11	50	50	1	
12	50	50	99	
13	50	50	100	

Be Positive.....

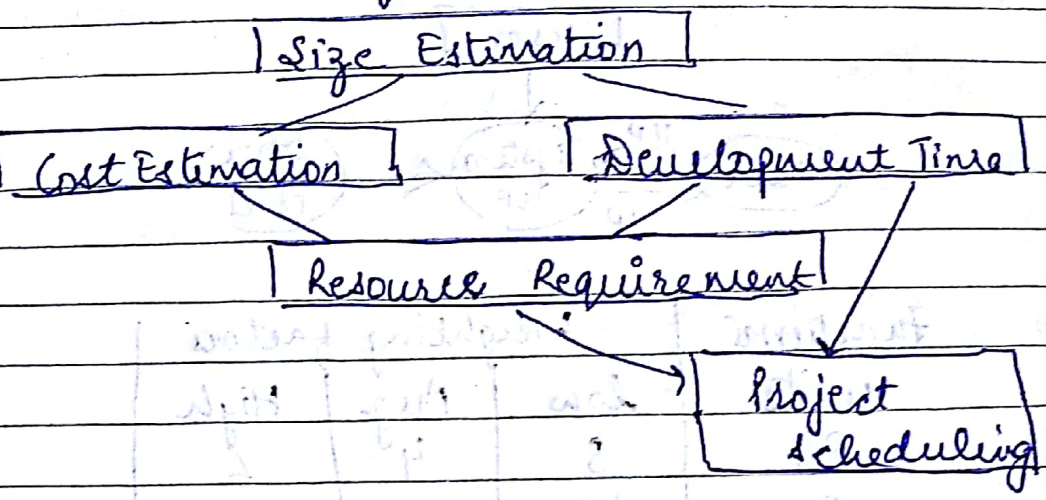
DATE: / /

Q
I/P

Consider a software module that is intended to accept the name of grocery items and a list of diff. sizes the item comes in the specification state that item name is to be alphabetic character 2 to 15 chars. in length. each size may be a value in the range of 1 to 48 whole number only. They are to be enter in ascending order means smaller size first. A max of 5 size may be enter for each item. the name is to be entered first followed by comma then by list of sizes. A comma will be used to separate each side. Spaces are to be ignored anywhere in the i/p.

DATE: __/__/____

Activities During Software Project Planning

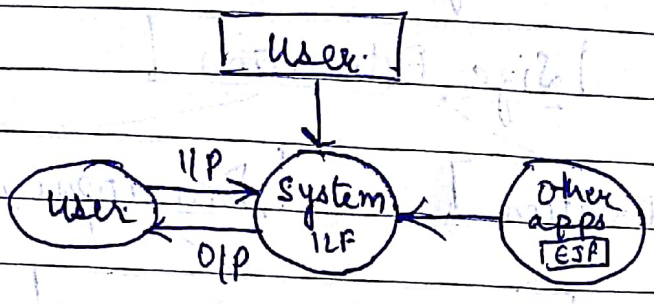


Size Estimation -

Function count or point - measure functionality from user point of view i.e on the basis of what the user requests & receive in return from the system. Function point analysis is decomposed into functional units - Input info - entering and output info. leaving the system. Request for instant access to info. Internal logical files - info held within the system.
External interface file - info held by other system. i.e used by the system being analyzed.

DATE: __/__/__

Line of Code



Functional Units	Weighting factors		
	Low	Avg	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
EIF	5	7	10

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} a_{ij} \cdot w_{ij}$$

Unadjusted function point

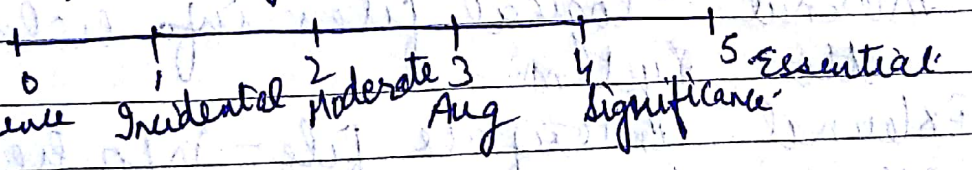
$$FP = UFP * CAF$$

function pt.

$$CAF = [0.65 + 0.01 * \Sigma f_i]$$

f_i (1 to 14).

Complexity adjustment factors



- Productivity – F.P / Person month (efforts by person)
- Quality – Defect / Person month
- Cost – Rupees / F.P
- Documentation – pages / F.P

Be Positive.....

DATE: __/__/__

Consider a project with following functional units I = 50, O = 40, Q = 35, Vf = 6 EI = 4 assume all adjustments factors to be avg. Compute the F.P for the project.

VFP = (50×4) + (40×5) + (35×4) + (6×10) + (4×7)

= 628

CAf = [0.65 + 0.01 + (14×3)]

= 1.07

FP = 628 × 1.07 = 672 (approx)

Q I - 10 (Low)
O - 12 (High)
Q - 12 (Avg)
ILF - 20 (Low)
EIF - 15 (High)

value of CAF = 1.10
what are the adjusted & UFP.

UFP = 10×3 + 12×7 + 12×4 + 20×7 + 15×10

= 30 + 84 + 48 + 140 + 150

= 452

FP = UFP * CAF = 452 × 1.10 = 497.2

200
150
80
84
484
48
532

DATE: / /

Q.

I = 10 Low
= 15 Avg
= 17 high
O = 6 low
= 13 high
Q = 4 Avg = 3 low
2 high
LF = 2 Avg
= 1 High
ELF = 9 Low

18
9
10

24
115

(10×3 + 15×4 + 17×6) + (6×4) + (13×7) + (3×3)
+ (4×4) + (2×6) + (2×10) + (1×15)
+ (9×5)

4/7
6
102
40
142

= 30 + 60 + 102 + 18 + 91 + 9 + 16 + 12 + 20 + 15 + 45
= 428

System req. significant data comp. performance is v. critical design code may be moderate. System is not designed for other installation

CAF = [0.65 + 0.01 × (4+5+2+0+10×3)
= 1.06

F.P = 1.06 × 428 = 449.44.

DATE: __/__/____

Constructive Cost Model (COCOMO) — Basic, Intermediate, Detailed

Composed by B. W. Boehm

Basic Model
- Organic
- Semidetached
- Embeded

Innovation	Mode	Project size	Nature of Project	Deadline of project	Develop. Environ.
little	Organic	2-50 KDSI	Small size, experienced developers in fam. env. eg- Payroll system	tight not	Familiar
Medium	Semi-detached	50-300 KDSI	Med. size Med size team. Avg. Prev. experience on similar projects eg- Complex db system	Medium	Medium
Embedd. significa nt	Embedded	over 300 KDSI	Large project, real time system Complex interfaces v. little prev. experience eg- ATM Air traffic control	Tight	Complex hard ware & Interface required

Project	ab	bb	cb	db
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

For calc. rough idea of effort & tym

Effort = $a_b(KLOC)^{bb}$ units - PM

(E)

Development tym = $C_b(E)^{bb}$ - M month

(D)

Person month

Average staff size = $\frac{E}{D}$ - Person

Productivity = $\frac{KLOC}{Effort}$ - KLOC/PM

Q Suppose that project was estimated to be 400 KLOC calculate the effort & development tym for each of the Key mod. i.e organic, S.D & embedded

→

Effort = $a_b(400)^{1.20} = 4772.8$

o - $2.4(400)^{1.05} = 1295.3$

d - $3.0(400)^{1.12} = 2462.7$

D = $2.5(4772.8)^{1.20} = 37.59$

D = $2.5(1295.3)^{1.05} = 38.075$

d = $2.5(2462.7)^{1.12} = 38.45$

DATE: __/__/__

Q) 200 KLOC size project is to be developed. S/W Deve. team has avg. experience on similar projects. Schedule is not very tight. Calc. effor, develop Time, Staff size, Productivity.

– $E = 3 (200)^{1.12} = 1133.12$ PM

$D = 2.5 (1133.12)^{1.12} = 29.304$ PL

$Staff = \frac{E}{D} = 38.667$

$Productivity = 0.176 PM = \frac{200}{1133.12}$

⇒) Intermediate

The basic model allowed for rough & quick estimate but resulted in lack of accuracy. Boehm introduced additional set of 15 predictors called cost drivers. Used to adjust the nominal cost of project to actual cost of project envison. Hence increasing accuracy of estimate. The cost drivers are grouped into 4 categories -

1- Product attributes -
- required s/w reliability
- db. size
- Product complexity

2- Computer attributes -
- Execution time constraints
- Main storage constraints
- Virtual Machine constraints

Be Positive

- computes turn around time

Difference b/w process & program

Program — set of instructions
process — set of programs in execution

3. Personal Attributes
- Analyst capability
- Application experience
- Programmer capability
- Virtual machine exp.
- Programming lang. experience

4. Project attributes
- modern programming
- use of S/W tools
- Req. development schedule
-

$E = a_i (KLOC)^{b_i} * EAF$
$D = C_i (E)^{d_i}$

Project	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

DATE: __/__/____

Q A new project wd estimated 400 KLOC embedded system has to be developed. Project mgr. has choice of hiring from 2 pools of developers: V. high capable wod V. little experience being used of the developers of low quality but a lot of ext. with programming lang. What is impact of hiring developers from one or other pool.

→ Programming cap v. high - 0.82
lang - 1.14

$$E = 2.8 \times (400)^{1.20} \times 0.9348$$
$$= 3470.14 \text{ PM}$$

$$D = c_i (E)^{d_i} = 2.5 (3470)^{0.32}$$
$$= 33.95 \text{ months}$$

$$E = 2.8(400)^{1.20}(0.95 \times 1.29)$$
$$= 4549$$
$$D = 2.5(4549)^{0.32} =$$

Product Matrix

DATE: __/__/____

- Describe the characteristics of the product such as size, complexity, design feature, performance, efficiency, reliability, portability
- Process matrix - Describe the effictiveness & quality of the product that produce the e.g. effort required in the process. time to produce the product, effectiveness of product during development. no. of effect found during testing majority of the process
- Project Matrix → Describe project characteristics and execution, no. of s/w developers, staffing pattern over the lifecycle of S/w. pace cost & schedule, Productivity

Reliability - failure free product.

Software Measurement -

It can be divided into two types -

1. Direct Measure
   Software process (cost & effort applied) and product (LOC, execution speed), defect reported over some set period of time.
2. Indirect measure
   Of the product that include functionality, quality, complexity, efficiency, reliability, maintainability etc

DATE: / /

Imp Types of Matrix.

1. Object oriented Matrix

→ No. of Scenario script — It is a detailed sequence of steps that describe the interaction b/w user & application

→ No. of Key classes - These are highly independent components. Tells amount of effort & reusability. ~~no. of sup~~

→ No. of Support classes - total no. of independent classes supporting key classes

→ Avg. no. of support per class → no. of supporting classes per key class.

→ no. of Subsystems — Aggregation of classes that supports a function that is visible to end user of system.

2. Quality Matrix

→ Correctness - is the degree to which the S/w perform its required function. Measured in defect / KLOC

→ Maintainability - It is the ease which a program can be corrected if an error is encountered. Adapted if its environment changes or enhanced when Customer req. change in environ

Be Positive.....

B+

Scanned by CamScanner

→ A simple time oriented Matrix is MTTC - (Mean time to Change), the time it take to analyse the change request, design an appropriate modification, implement the change, test it & distribute the change to all users: it should be low

MTTC ∝ 1/maintainability

→ Imp

Integrity - This measures systems ability to withstand attacks (both accidental and intentional) to its security. Attacks can be made to all 3 components of SW i.e program, data & documents. To measure it 2 additional attributes must be defined threat & security.

- Threat is the probability that an attack of a specific type will occur within a give time.
- Security is the probability that a specific attack will be repelled.

$$\text{Integrity} = \Sigma(1 - (\text{threat} * (1 - \text{security})))$$

→ Usability → If a program is not easy to use it is often let to failure even if the functions perform are valuable.

Be Positive.....

→ Defect Removal Efficiency (DRE) - It is a measure of the quality assurance & control activities

$$DRE = \frac{E}{(E+D)} \quad \text{or.} \quad D$$

where E is no of errors found before delivery of project to user. d is defects found after delivery.

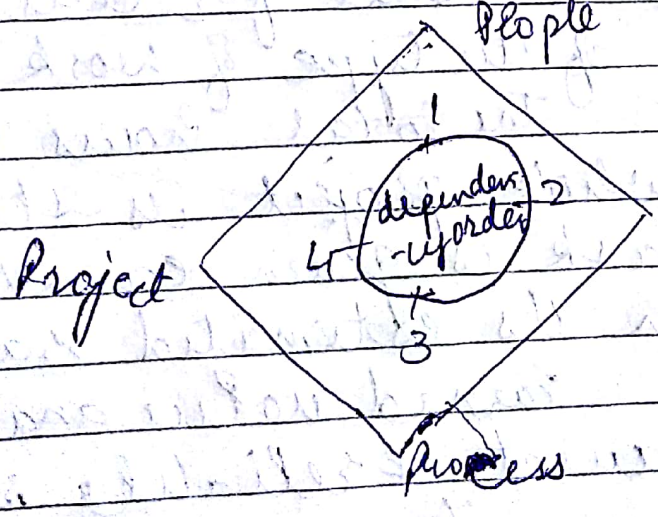
$$DRE_i = \frac{E_i}{E_i + E_{i+1}}$$

E_i is no. of errors found during i^{th} activity $i+1$

Estimated value (s) = $(S_{opt} + 4S_m + S_{pess})/6$

Monty Carlo estimation — optimistic — most likely — pessimistic

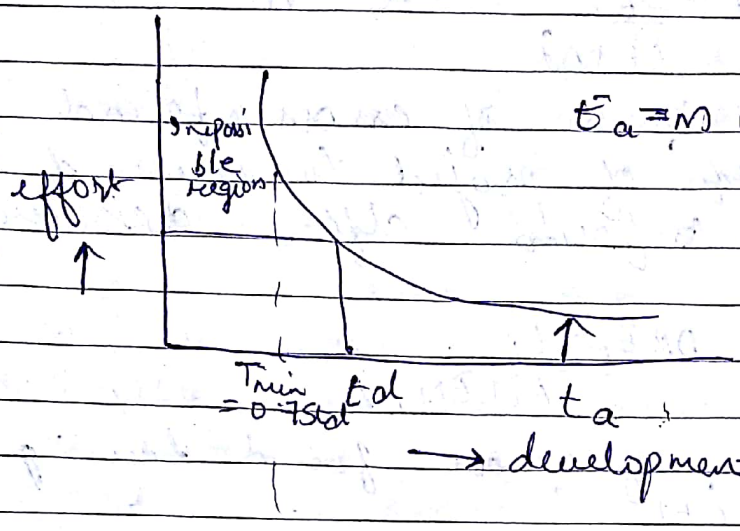
Role of Management in S.D / Management Spectrum



DATE: ___/___/_____

The relationship between people effort and time

P-N-R curve - Putnam-nardar-Rayleigh curve



E_a = Effort in person month
t_d = nominal delivery time
t_o = optimal development time
t_a = actual delivery time desired

Earned Value Analysis

The Earned Value system provide a common value scale for every task regardless of the type of work being performed. The total hours to do the whole project is estimated & every task is given ear value based on its estimated %age of the total. Earned value analysis provide accurate & reliable reading of performance. From as early as 15% into the project

PV = Planned completion (%) * BAC

EV = AC (%) * BAC

DATE: __/__/__

To determine the Earned value following steps are performed.

1. Budgeted cost of work schedule (BCWS) (Planned value) is determined for each work task represented in the schedule to determine progress at a given point time.

The value of BCWS is the sum of BCWSi value of i tasks.

2. The BCWS values of all work tasks are summed to drive the budget at completion (BAC)

3. Budgeted cost of work performed (BCWP) (Earned value) Its value is sum of the BCWS value for all work tasks that have actually been completed by a point in time of the project schedule

4. Schedule performance index
SPI = BCWP/BCWS EV/PV

5. Schedule variance = BCWP - BCWS

6. Percent schedule for completion = BCWS/BAC

7. Actual cost of work performed (ACWP) (Actual cost) is the sum of the efforts actually expended on work task that have been

Be Positive.....

DATE: __/__/____

completed by a point on the project schedule

8. Cost performance index (CPI) = $\frac{BCWP}{ACWP} = \frac{BV}{AC}$

9. Cost variance (CV) = BCWP - ACWP

Q Assume you are a software project manager and you have been asked to compute earn value for a small software project the project as per task is required

At the time you have been asked to do Earn value Anal. 12 tasks have been completed. however the project scheduled indicate that 15 tasks should have been completed. The following tasks schedule that data in days are variable.

DATE: __/__/__

Task	Planned Effort	Actual Effort
1	12	12.5
2	15	11.0
3	13	17.0
4	8	9.5
5	95	90
6	18.0	19.0
7	10.0	10.0
8	4.0	4.5
9	12.0	10.5
10	6.0	6.5
11	5.0	4.0
12	14.0	14.8
13	16.0	—
14	6.0	—
15	8.0	—

BCWS = 148.5

Compute the SPI, SV, Person scheduled for completion, person complete, CPI, & cost variance

→ BAC = 582

BCWP = 126.5

ACWP = 128.3

SBI = 80.8%

SV = -30

PSC = 156.5/582

CV = 105

CPI = 98.59

P.C = 27.9

Q Suppose you have a budgeted cost of a project at 9 lakhs $. The project is to be comp. in 9 months. After a month you have comp. 10% at total expense of 1 lakh dolar. The planned completion should have been 15%.

PV = 15% (900000). = 135,000
EV = 10% (900000) = 90,000

$$CPI = \frac{10\% \cdot 900000}{100,000} = 0.90$$

$$SPI = \frac{10\% \cdot 900,000}{15\% \cdot 900,000} = 0.67$$

$$\frac{10 \times 100}{100 \times 15}$$

0.6
3 | 2
20
18
2

Since, CPI & SPI are less than 1 it means project is going slow

Suppose you are managing a S.D project the project is expected to be comple in 8 months at cost of 10,000 $ per m After 2 months you realize the proje is 30% at the cost of 40,000 determine whether project is on time and in budget after 2 months

DATE: __/__/____

BAC = 8000

AC = 40,000

BCWP = 30%

PV = ~~80~~ 25% 80000 = $\frac{25}{100} \times 8000$ = 20,000

EV = 30% 40,000 = $\frac{30}{100} \times 8000$ = 2400

CPI = $\frac{EV}{AC}$ = $\frac{24000}{40,000}$

SPI = $\frac{24000}{20000}$ = 1.2

10

Activity Diagram (Sequence diagram)

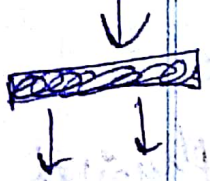
Activity diag. is a behaviour diagram because they describe what must happen in the system being modelled.

Benefits:

1. Demonstrate the logic of an algorithm
2. Describe the steps performed in an UML use case.
3. Simplify ~~a use case~~ and improve any issues by clarifying complicated use-cases.
4. Model software architecture elements such as methods functions and operations.

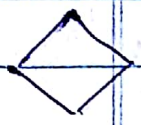
Components:

Symbol	Name	Description
●	Start symbol	Represents the beginning of a process or work flow in an activity diagram.
Activity	Activity symbol	Illustrate the activities that make up a modeled process
→	Connection	Show the directional flow or control flow of the activity
↓↓ ― ↓	Join symbol/ synchronization bar	Combines two concurrent and re-introduces them to flow where only one is at a time.



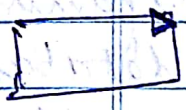
Fork symbol

Splits a single activity flow into 2 concurrent activities



Decision Symbol

Represents a decision and always has atleast two paths branching out with conditions to ~~ask~~ user to view options.



note symbol

additional message

flow final Symbol

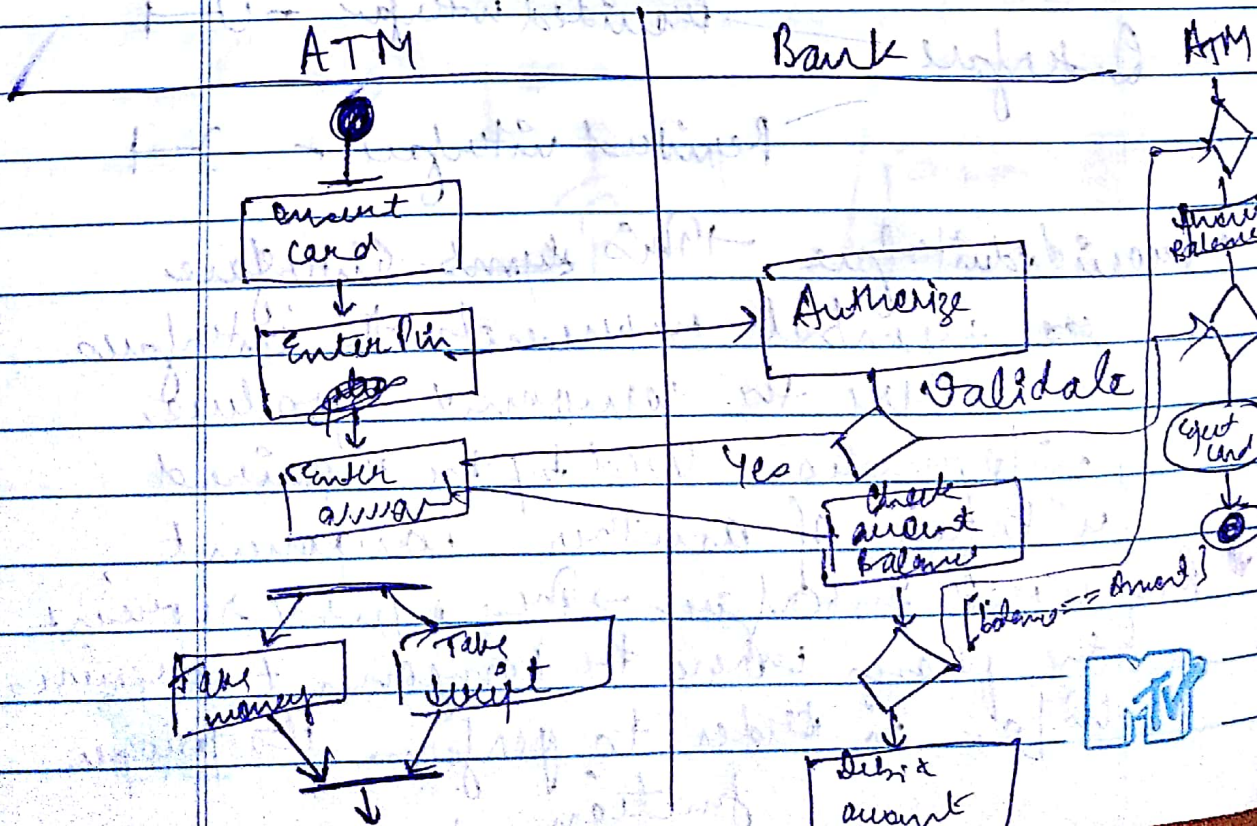
Represents the end of the flow

the specific flow final symbol should be placed at the end of the process in the single activity flow

End Symbol → Mark the end. state of an activity and represent the completion of all flow of the process

Activity Diagram is of 2 types ←

1. Simple
2. Parallel



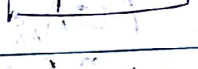
Component diagram

A component diagram describes the organisation and wiring of physical components and as in a system.

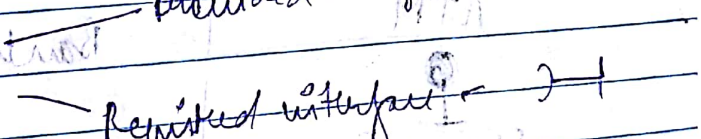
Components included in these diagram are physical like documents, database tables, files and event tables. all physical element with the locations.

Basic components diagram symbol and notation:

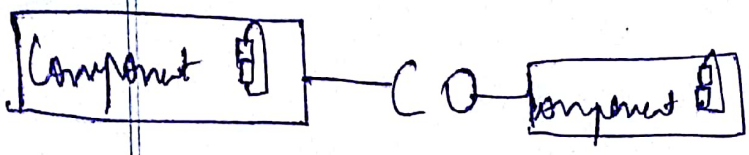
1. Component is a logical unit block of the system, a slightly higher abstraction than class.



2. Interface describes the group of operation used or created by components

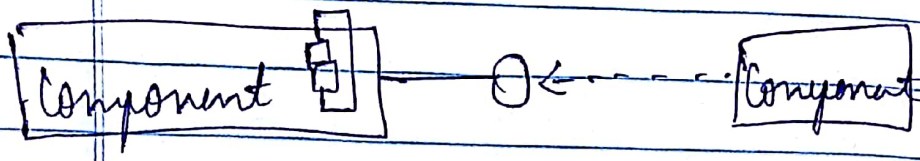


- Provided interface → This symbol represents the interface that the component provides. It specifies the operations and information that it offers to other components.
- Required Interface → This symbol represents the interfaces that a component needs to perform its function. It specifies the operations and information that it requires from other components in order to perform its function.

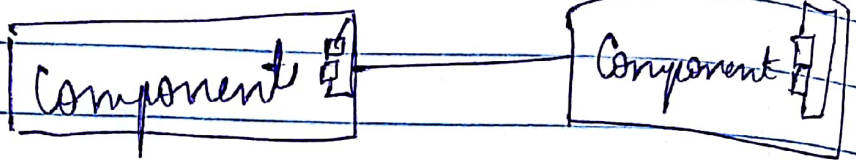


draw

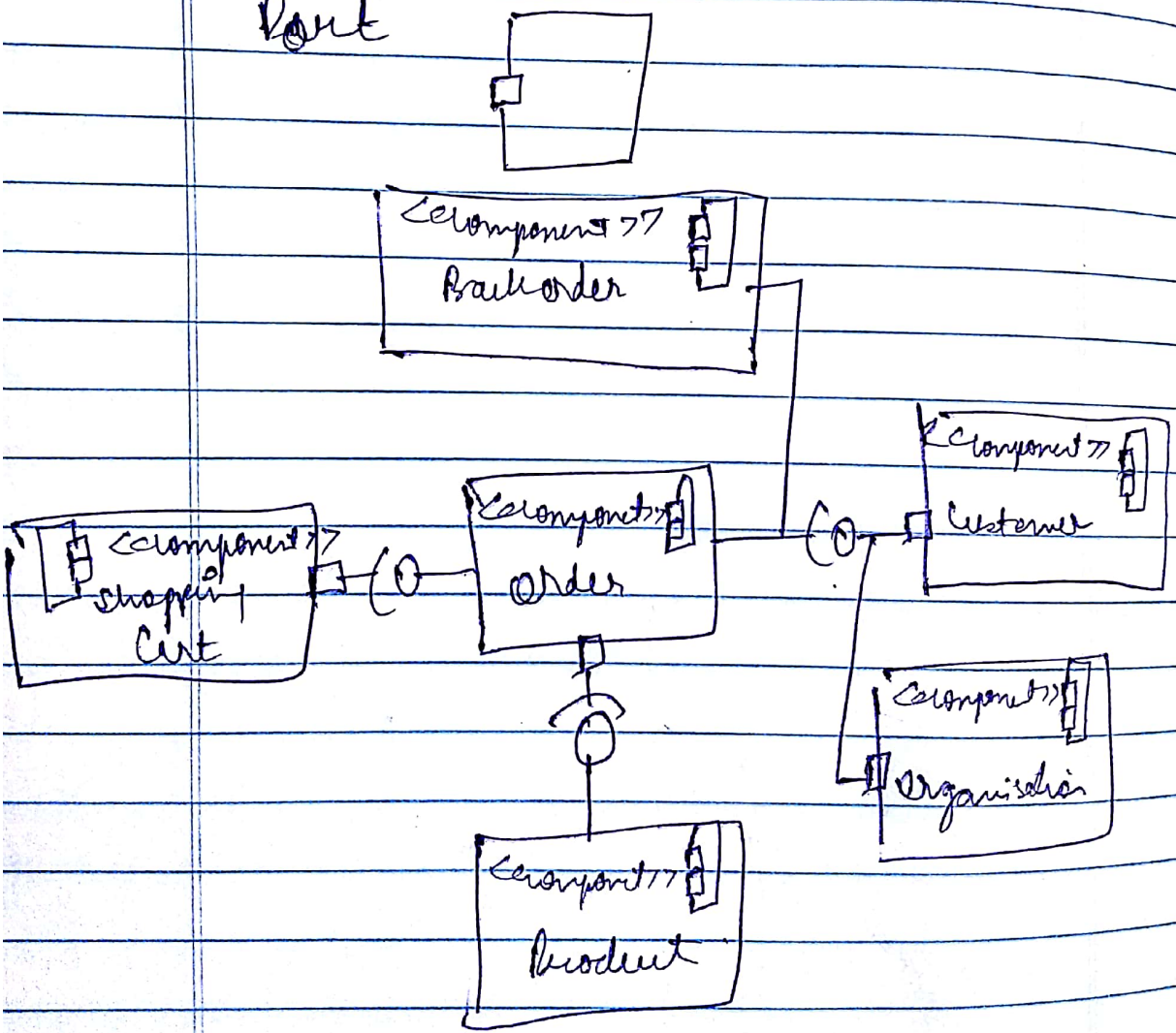
Draw one dependency among.



Association is a simple relationship between 2 components



Port



Q. Verification and Validation

Verification	Validation
① It is a static practice of verifying documents design, codes.	① It is a dynamic mechanism of validating and testing the actual product.
② It does not involve executing the code	② It always involve executing the code.
③ It is human based checking of documents and files.	③ It is computer based execution of program.
④ Verification uses methods like inspections, ~~Review~~ review, etc.	④ Validation uses methods like Black box testing, white box testing, etc.

(5) verification is to check whether the software conforms to specification.	(5) validation is to check whether the software meets the customer expectations and
(6) It cannot catch errors that validation can.	(6) It can catch error that verification cannot.
(7) Target is requirement specification and software credentials	(7) Target is actual product. A unit a module, integrated modules and effective final product
(8) A module a unit integrated modules and effective final product.	(8) validation is carried out with the involvement of testing team.
(8) Verification is done by quality assurance team	

α-testing | β-testing

α-testing	β-testing
(1) It is always performed by the developers at the s/w development site	(1) It is always performed by the customers at their own site.
(2) α-testing is not open to market and public	(2) β-testing is always open to market and public
(3) It is conducted for the s/w application and project.	(3) It is usually conducted for software product.
(4) It is always performed in virtual environment.	(4) Performed in real-life environment.
(5) It is always performed within the organisation	(5) Always performed outside the organisation
(6) α-testing is definitely performed and carried out at developing organisation location with a involvement of developers	(6) β-testing is performed and carried out by users or general people at their own locations and site using customer data.
(7) It comes under the category of both white box and black box testing.	(7) It is only a kind black box testing.
(8) α-testing is always performed at the time of acceptance testing when the developers test the product and project to check	(8) β-testing is always performed at the time when software project and product are marketed.