Information system S/w Engg

Textbooks 2 Copies

7/10

- S/w engg — A practitioners approach — Roger Pressman 1997.
- S/w engg Principles — Richard fairley, 1985.
- S/w engg — Ion Sommerville.
- S/w engg — Ghezzi.

S/w Engg — S/s mtc, disciplined, quantifiable approach to the developmt, opn and maintenance of s/w.

Objective

development/ maintenance	cost/time

100

$\underbrace{\frac{40}{60} \quad \frac{30}{70}}_{\text{effort distribution}}$ more for maintenance

Syllabus

2) Development
- Analysis & Design
- Code
- Testing

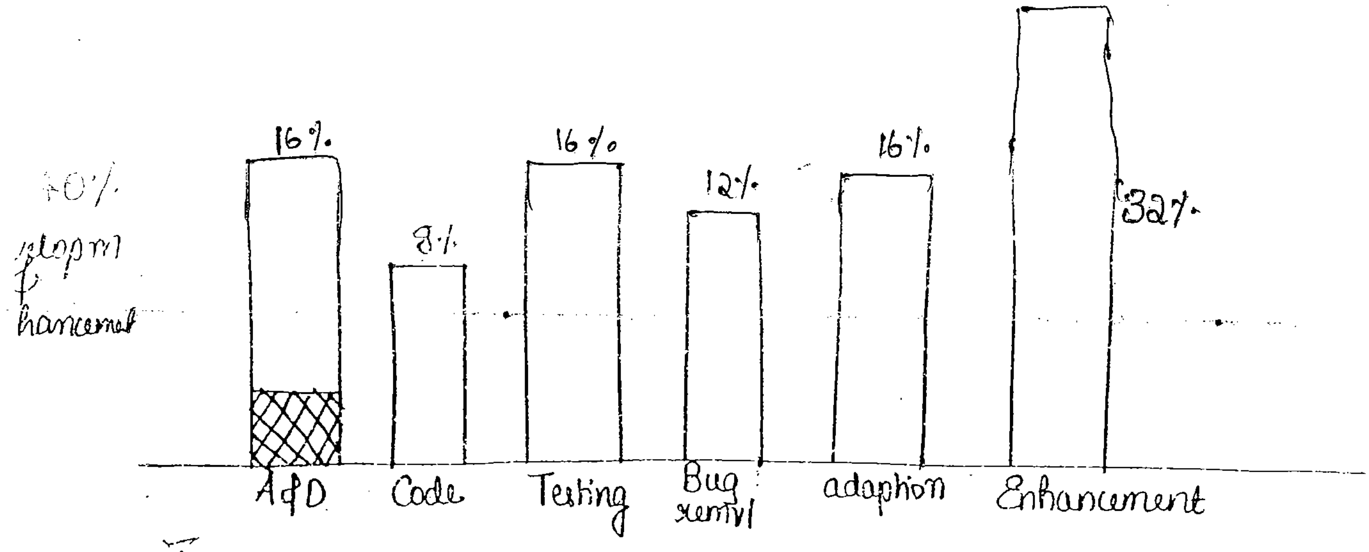
- data structure
- func implementation?
- procedural details
- interface characteristics
- design translated into pgmng lang
- how tested?

3) Maintenance
- Bug removal & corrective maintenance
- adaptive or adoption
- Enhancement / perfective

- error correction
- adaptations reqd.
- changes due to enhancemnts

1) Definition
- System / Info Engg
- S/s planng
- Requirement Gathering

key reqmts of s/s & s/w identified
- what inform processed?
- fns & performance reqd.
- expected s/s behaviour.
- interfaces.
- design constraints
- validation constraints reqd.



To develop a pdt $\Rightarrow$ Process model.

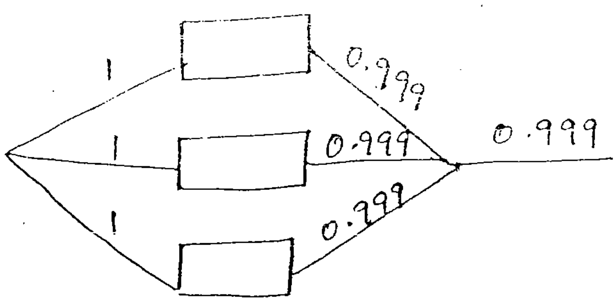
S/w Engg. Concepts

- Quality
- Reliability
- Defect-free

Module can be connected

$1 \longrightarrow 0.999$

Series



For series Reliability of S/S = $\left(\text{Reliability of a module}\right)^{\text{no. of modules}}$

$= (0.999)^4$

By default parallel conn

$$\frac{90}{60 \times 24 \times 365} = 0.00000179 \rightarrow \text{non availability of pdt}$$

$$1 - 0.00000179$$

99.99% reliable

Defect-free

Bug/error ——— Defect

75

$\underbrace{100}_{\text{Passive}}$ ——— $\underbrace{25}_{\text{Active}}$

→ Error becomes a defect when activated.

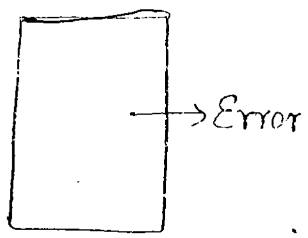
Errors
- Known Errors
- Predictable Errors
- Unknown Errors → reactive bugs

Known Errors, Predictable Errors → Proactive bugs → defect free

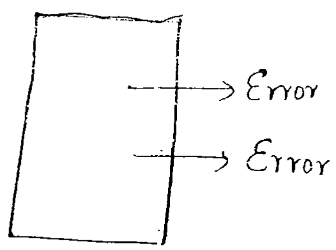
→ Proactive → we can take a decision

→ Once released, reactive bugs comes up

→ High reliable



Reliable



→ 2:1 productivity

Parameters

→ Inform Determination

time i/p → time o/p

→ Inform. content

value x → value y

S/w
- Application
- S/S
- utility

Eg: Rectangle

	fish	C	µP
progmr view		40	80
developer	300 (appli)	180 (utility)	80 (S/S)

Ratio = 1 : 3 : 9 [lines of instns]
= S/S : utility : application.

1st generation → interact directly with machine.
2nd → procedural based
3rd → structured lang
4th. → interface lang.

Size

<2K — small prjt
2 – <8K — medium prjt
8 – <32K – large prjt
>32K – very large project.

small | 2 | medium | 8 | large | 32 | v. large

COCOMO (Barry Boehm)

→ Cost Constructive model

→
- Organic mode prjt (applic) non-databased.
- Semidetached mode " (utility)
- embedded mode " (s/s)

} databased.

	Reqmts	Skills	Team Size	Duration	Examples.
Organic	simple	HLL	1-50	↑month-yr↓	- Business s/w - scientific s/w - small compiler - An OS - low end s/s laptop desktop - simple inventory.
Semidetached	simple + complex	Mixed mode	1-100	↓months-yrs↑	- OS s/s - medium level s/s VAX, PDP-11 - moderate dB applicat - simple command & control s/s.
embedded	composite	High technical	100-1000	>years	- OS → high end s/s. → mainframe, super cmptr - Very Large Transaction Processing System. - complex command & control.

⟹ Requirements → Services
Requirements → Operational constraints

Specification → functionality
Specification → performance
Specification → operatnl constraints

⟹ Whole world

Richly waged

Onshore { USA, Japan, Europe

Poorly waged

India, Russia, China, Brazil } Off Shore

hires projects
no own projts

CMM			Nature	
5	optimised	40	Continuous improvmt	Change mgmt & defect mgmt
4	Managed Predictable	28	Predictable	Prodt & Process Quality
3	defnd	21	Standard	S/w process Engg {A&D, C, T}
2	repeatable	14	consistent	Proj mgmt.
1	initial company	7	disciplined	Bkgnd Project work.

⟹ Only CMM 5 company can produce high reliable pdt

others → reliable pdt

SEI

S/w Engg Institute

is able to satisfy ... than dream company (in 3 yrs)

Handout − 1

1) A
2) A
3) A organ.
4) A
5) B
6) A
7) C
8) A
9) A
10) B (since days)
11) C
12) B
13) B
14) C
15) C
16) C
17) B
18) B
19) A
20) B
21)
22) D
23) C
24) A
25) C
26) D
27) C
28) B
29) D
30) B
31) B
32) $(0.999)^{100}$
33) (")
34) " $)^{1000}$
35) 0.999. B.

(3)

10 / 10 / day

Process Models

- provides description (guidelines)

Development — Modifications.

- without models, we can't assure quality.
- " " , we cant achieve continuous improvmnt
- cant control s/w engg process activities
- can't estimate properly.

Microsoft	IBM	Infosys
Stabilize/synchronize process model.	Rationalize unified process model. (1st Proc. mdl in market)	Waterfall model
		Spiral model

tie up for testing

- make user used to the new pdt
- then migrate to new.
- free evaluation period for antivirus

1) Inception
2) Elaborating
3) Construction
4) Transition

} phases.

	W/w XP	W/w vista	w/w 7

Performance

Size		
OM	14 GB	4 GB
SM	100 GB	60 GB
EM	400 GB	100 GB

1) Waterfall model

- classical Life cycle proc mdl.
- linear sequential model.
- in 1970
- Winston Royce developed it.
- also called Royce Model
- Document driven approach.

- changes couldnt be accepted
- pdt will show <u>low performance risk</u>.
- <u>low interface risk</u>.
- conventional pdts can be developed.
- Core pdt is itself the final pdt.
- No versions.
- now this model is used for background works.
- customer was not under control.

2) Incremental Process Model:

- n versions will be released.
- Pilot approach.
- upgraded versions will be released.

3) Evolutionary Process Model:

- Iterative approach - any reqr change could be worked on.
  - Spiral model (h/w & s/w) (modified waterfall model)
  - Concurrent process model
  - Prototype model
    - RAD (Rapid Action design) (high speed approach of linear sequential)
    - throwaway prototype [more than 1 prototype]
    - Evolutionary prototype [1 prototype]

Spiral model

- in 1985.
- Boehm model
- introduced risk analysis
- risk driven approach.

- not only s/w design, this model can be used for h/w design.
- only model which includes h/w & s/w design.

Concurrent model

- for simultaneous development
- eg: client/server s/w.

parallel s/s — Centralised s/s — Tightly coupled s/s (100% sharing)

Distributed s/s — decentralised s/s — loosely coupled s/s. (nominal sharing of resources)

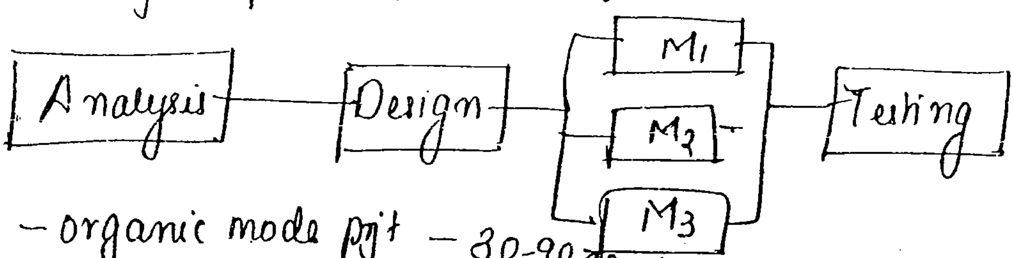
Prototype model

- customer not sure of requirements.
- in 1980s.
- for a look and feel.

RAD

Rapid ~~Action~~ Appln Design.

- high speed approach of linear sequential model.



- organic mode pjt - 30-90 day.

Throwaway prototype:

- using more than 1 template.

Evolutionary prototype

- only 1 template.

4) Component Based development

- Obj. Oriented technology.
- reusability paradigm.

Components

- OSC - off shelf component : ready made. comp from library.
- Fully experienced components : require modificatn.
- Partial experienced components : go for a new component. bcz partial exper. is risky.

) Formal method:

- transformational model.
- clear specification of the project.
- verifications have become simplified.
- defect free.
- mathematical approach - also called (specification driven)

) RUP

- in 1990s.
- Rational unified Process model.
- good quality pdts.

Linear seq. model of s/w development is (6)

1) Reasonable approach where reqmts are well defnd.

2) A good approach when working pgm reqd quickly. (Incremental)

3) The best approach to use for projects with large development teams.

4) An old fashion that cant be used in modern approach.

2. The linear seq. mdl of s/w developmt is also known as

1) Classical life cycle. mdl.

2) Fountain mdl

3) spiral model

4) waterfall mdl

The RAD is 1) another name for comp. based development

2) useful app. wen custmrs cant defn reqm clearly.

3) high speed adaptn of linear seq. mdl.

4) all of the above.

iterative evolutionary model

Evolutionary s/w proc mdls are

1) iterative

2) easily accomodate pdt reqm changes

3) dnt generally produce throwaway s/s

4) all of the above.

Spiral model of s/w development

1) ends with delivery of s/w pdts

2) more complex than incremental model. (No comparisons)

3) includes project risk evaluation in each iteratn

4) all of the above.

Concurrent development mdl is

1) another name of RAD

2) often used for development of client-server appln.

3) only used for development of || or distributed s/s.

4) used whenever large no. of change requests are anticipated.

Component based development is

1) only appropriate for h/w design

2) not able to support development of reusable components.

3) works best when object technologies are available for support.

4) not cost effective by nonquantifiable s/w matrix.

8. The formal method mdl of s/w development makes use of mathematical methods to

1) defn specification for computer based s/s.

2) develop defect free computer based s/s.

3) Verify correctness of computer based s/s.

4) all

9. Which of these is not one of the phase names defnd by unified process mdl for s/w developmt.

1) in ~~sufficient~~ ception

2) elaboration

3) construction

4) Validation

Software Planning

- S/w project estimation
  - * Delay estimation
  - * Base estimation
  - * Decomposition techniques → Problem based / Process based
  - *
  - * Empirical models.

Problem based:
- LOC size oriented metrices (direct method)
- fn oriented fp, indirect (functional pt)

$$E_v = \frac{S_{opt} + 4S_{likely} + S_{pess}}{6}$$

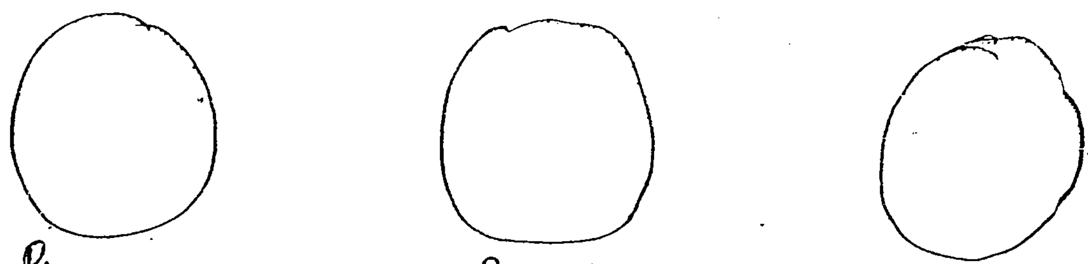
Size of optimistic
" likely
" pessimistic.

$$\text{Variance} = \frac{UB - LB}{6}$$

$$LOC = fp * \left(\frac{LOC}{fp}\right)$$

Delay estimation - late estimation
- time and cost cnt be estimated once the pdt is produced.
- report cant be prepared.

S/w attributes:



Process	Project	Pdt
reqm gathering	Quality cntrl	Quality = 1
	Productivity contrl	reliability = 0.99
	estimation contrl	efficiency = 0.995
	Activity contrl.	robustness.

egs:

Measure — Total available attributes

Eg: 100

metric — evaluate the degree of involvment of the above s/w attributes.

Eg: 60.

⇒ S/w developmt & maintenance involves a broad range of s/w metrices which determines the qualitative measure of the degree to which s/w attributes are involved at diff stages, whereas measure determines quantitive indication of s/w attributes at diff levels.

⇒ S/w metrices are used to evaluate, characterize, improve & predict the attributes.

› At process level, analyst ~~~~ uses these metrices for obtaining unique requirements from customer, generally the reqm provided by customer involves duplication, conflicts & disagreements. So, metrics helps in removing all this anomalies.

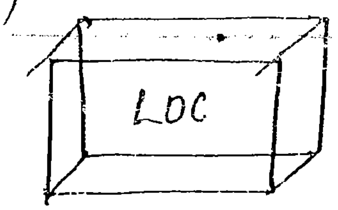
⇒ At projt level, these metrics are used for s/w projt estimation quality control, pdtivity control, s/w engg process so on. & activity control or s/w engg process so on.

› At pdt level, they are used to evaluate quality reliability, efficiency, portability ...

Decomposition Technique

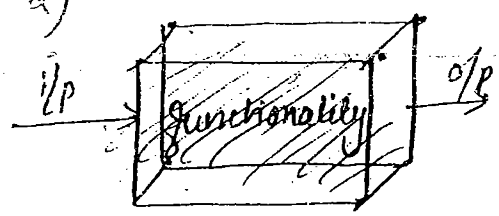
— Problem based decomposition

1)

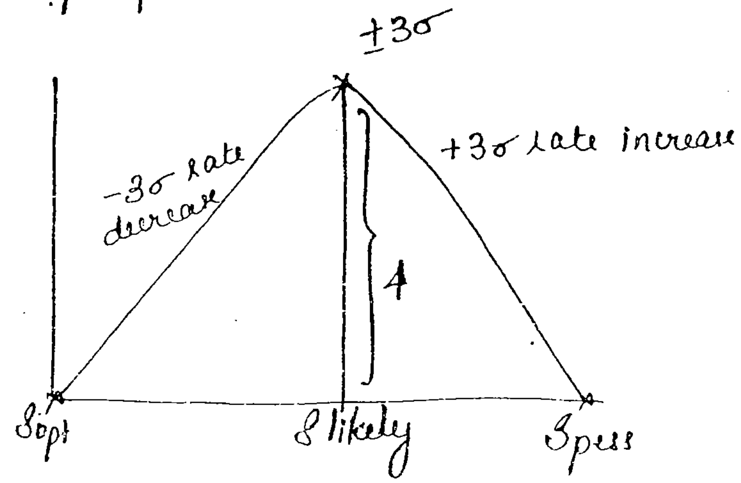


direct measure
white box measure
Internal specification based

2)



indirect measure
black box measure
External speci. based.



$$E_v = \frac{S_{opt} + 4S_{likely} + S_{pess}}{6}$$

Eg 100 UIF

Best	Moderate	Worst
4800	5300	6400

$$E_v = \frac{4800 + 4 \times 5300 + 6400}{6}$$

$= \underline{\underline{5400}}$

Bank appln
- UIF 5400
- DB 8500
- server 7300
- client 5500

Total size of prjt = $\underline{26700}$ LOC
= large project.

If Productivity − 870 LOC / month
of each practitioner.

effort = size ... $\frac{26700}{}$ = 30.68 ~ 31 person month cost of the pdt = effort × Pay   if pay = \$5000

$= 31 \times 5000$

$= \$\ 155K$

size oriented table

Name   Size   cost (\$)   effort   PPdoc   Errors   Defects

Name	Size	cost	effort	Page per doc	Errors	Defects	People
α	35K	\$450K	120	320	50	48	20

module info is provided under the size column.

$\frac{120}{20} = 6$ months required.

Note:

1) $\text{Effort} = \frac{\text{size}}{\text{productivity}}$

2) Productivity = size / effort

3) $\text{Quality} = \frac{\text{Errors}}{\text{KLoc}}$

4) $\text{Documentation} = \frac{\text{Ppdoc}}{\text{KLoc}}$

5) Cost of a line = \$ / Loc

6) Cost of s/w = effort × pay.

Functional oriented metric, FP (Indirect measure) (9)

17/10 Friday

FP
- 2D FP
- 3D FP
- adjustable fp
- unadjustable fp

$\boxed{FP = count\_total * EAF}$ adjustable fp

$\boxed{FP = count\_total}$ unadjustable fp.

Domain

Info Domain
Func. Domain —— no. of transformations
Behaviour Domain — no. of transitions

(7) 3D FP

Info domain
- no. of i/ps
- no. of o/ps
- no. of inquiries
- no of files
- no. of ext. interfaces

(5) 2D FP

$$FP = count\_total * \left[\begin{array}{l} EAF \\ 0.65 + 0.01 * \underbrace{\Sigma F_i}_{58} \end{array}\right]$$

1 – 14 factors

0 1 . . . . 5 = 58

=1.23

last column

fp count		S_{opt}	S_{likely}	S_{pess}	EV	wt
87	No. of i/ps	25	29	33	29	*3
104	No. of o/ps	21	27	29	26	*4
48	no. of inquiries	12	16	19	16	*3
42	no. of files	4	5	9	6	*7
25	no. of ext. interfaces	3	5	7	5	*5

306 = count_total
↳ unadjustable fp.

306
123

FP = 306 × 1.23
= 376

Effort = fp / productivity   productivity = 12 FP/PM
= 376/12
= 31.33 Person_month

cost of s/w = effort * pay
= 31.33 × 5000 = $ 155K

cedure based Decomposition
Analy – codg – Testing
40 – 20 – 40 out of 100:

customer communication   technical aspect   non tech. aspect   customer evaluation

Activities / Modules	CC	Planng	Risk analysis	analysis	Design	Code	Test	CE	Effort
UIF				0.60	2.5	0.70	2.8		6.6 $33K
DB				0.90	3.0	0.8	3.0		7.7 $38K
Server				0.80	2.8	0.9	3.2		7.7 $38K
Client				0.70	2.0	0.9	2.8		6.4 $32K
					10.3	3.3	11.8		

	loc	FP	Procedure based
	31	31	28.4
maim cost.	\$155 K	\$155 K	~~\$137K~~ \$141K

\$160K → \$15K

\$165 . profit 10 K.

Pg.27 6. size 33200
P 620
pay \$ 8000

effort = size/produc.
= 33200/620
= 53.5 = 54 person month.

cost = effort × pay
= 432000
= \$ 432 K

3/16 LOC = fp * LOC/fp
= 372 * 90

Pascal: 33480 33200

Ass.lang: 372 × 320 = 119 K

29/17. C/LOC = 372 * 128 = 47 K

29/18. 0/LOC = 372 * 30 = 11 K

29/19. LOC = 372 * 4 = 1488 K

Empirical Model

LOC or FP — all calculations depend on it

CoCoMo model _ size _ LOC

- Barry Boehm.
- hierarchy of estimation model.
  - Basic CoCoMo — basic idea
  - Intermediate CoCoMo — exact evaluation
  - Advanced CoCoMo.

Basic CoCoMo

- all computation based on size of project.
- $\boxed{E = a_b (kLOC)^{b_b}}$ Person_month.
- $\boxed{D = C_b (E)^{d_b}}$ months (duration)

No. of people in team $\boxed{N = E/D}$ persons.

	a_b	b_b	C_b	d_b	a_i	b_i
Organic mode	2.4	1.05	2	0.38	3.2	1.05
Semidetached	3.0	1.12	2.5	0.35	3.0	1.12
Embedded	3.6	1.20	2.5	0.32	2.8	1.2

Intermediate COCOMO

(11)

- Size + cost drivers considered.

- cost drivers
  - Personal attributes
  - Project "
  - Product "
  - H/w "

0.7 — 1.3 (range) old

0.9 — < 2.0

- Personal attr
  - avg pgmr: < 1 yr experience, 1 unit
  - Excellent pgmr: > 1 year, 2 unit

Assume out of 32 ($\times 5000$) members, if we include 20 excellent memb, then we need 24 avg memb.

20+24 = 44 x 5000

Super programmer — compt

3 unit

$$\boxed{E = a_i \, (kloc)^{b_i} \times EAF}$$

Effort adjustment factor (EAF)

Empirical mdls are estimatn models which uses empirically derived formulas for predicting based on LOC and FP

Diff authors provided diff mathematically derived Formulas based on either LOC / fp as shown.

LOC oriented Estimation Models

$E = 5.2 \times (KLOC)^{0.91}$ Waltson Felix mdl

$E = 5.5 + 0.73 \times (KLOC)^{1.16}$ Bailey-Basili mdl.

$E = 3.2 \times (KLoc)^{1.05}$ simple Boehm mdl.

$E = 5.288 \times (KLoc)^{1.047}$ Doty mdl (KLOC > 9)

FP oriented Estimation mdl

$E = -13.39 + 0.0545\ FP$ Albucht & Gaffey.

$E = 60.62 \times 7.728 \times 10^{-8}\ fp^3$ Kamerer mdl.

$E = 585.7 + 15.12\ fp$ Motson, Barnet & Mellichamp mdl.

COCOMO

- Cost Constructive Mdl by Barry Boehm.
- This mdls are based on LOC
- hiearchy of mdls include
  - Basic COCOMO
  - Intermediate COCOMO
  - Advanced COCOMO

Basic COCOMO

- computes S/w developmt efforts and duration as a func. of pgm size/proj size which is expressed as LOC.

- Basic COCOMO mdl provides rough idea abt estimation.

- The formula & table of info is as shown.

- refer P.TO -

Intermediate COCOMO

- computes s/w development efforts in terms of func. of prjt size or pgm size with a set of cost drivers which include subjective assessment of the drivers. (personal, prjt, pdt, h/w)

- Cost drivers value plays a major role in computation of efforts

- it ranges from 0.7 - 1.3 & 0.9 - (<2)

- I COCOMO provides exact evaluatn reqd for project

- formula & table — refer prev —

Advanced COCOMO

- incorporates characteristics of intermd. version with the subjective evaluation of cost drivers impact on the s/w ~~project~~ process activities, (A, D, C, T). when

Eg:- 20 K

Basic COCOMO

organic mode

$E = 2.4(20)^{1.05}$

$= 55.75$ Person-Month

$D = 2.5(55.75)^{0.38}$

$= 11.52$ months

$N = 55.75 / 11.52$

$= 4.8 \simeq 5$ persons

Semidetached mode

$E = 3.0(20)^{1.12}$

$= 85.95$ PM

$D = 2.5(85.95)^{0.35}$

$= 11.8$ months

$N = \frac{85.95}{11.8}$

$= 7.2 \simeq 7$ persons

Embedded mode

$E = 3.6(20)^{1.20}$

$= 131$ P-M

$D = 2.5(131)^{0.32}$

$= 11.89$ months

$N = \frac{131}{11.89}$

$= 11$ persons.

Time remains constant when the size is constant $\simeq 12$ months if size & EAF grn, go for Icocomo

Eg: (a) 20 K —— EAF = 1.250 —— D = 12 mnths

Intmd cocomo

organic mode

$E = 3.2 (20)^{1.05} \times 1.250$

$= 92.9$ PM N=8

Semidetached

$E = 3.0 (20)^{1.12} \times 1.25$

$= 107.4$ PM N=9

embedded

$E = 2.8 (20)^{1.20} \times 1.25$

$= 127.4$ PM N=11

Comparing with basic cocomo. the change in 'E' organic mode.

Eg. (3) 20 K —— 1.675 (EAF) —— 12 months

Organic

$E = 3.2 (20)^{1.05} \times 1.675$

$= 124$ PM N = 10

Semidetached

$E = 3.0 \times (20)^{1.12} \times 1.675$

$= 116.7$ PM N= 12

embedded

$E = 2.8 (20)^{1.2} \times 1.675$

$= 170.76$ PM N= 14

(3) if drivers not calculated properly calculations go wrng.

		OM	SM	EM
ort	Basic COCOMO	5 :	7 :	11
	Intermd COCOMO			
	1.25	8 :	9 :	11
	1.675	~~20~~ 10 :	12 :	14
cost	Basic COCOMO	\$278K	\$429K	\$655K
	Intermd. COCOMO	\$464K	\$527K	\$637K
		\$625K	\$720K	\$850K

S/w Equation

1/10 day

$$E = \left[\frac{LOC \times B^{0.333}}{P}\right] \times \left[\frac{1}{t^4}\right]$$

B – special skill factor
P – productivity
t – duration
E – effort

⟹ If project size is 5K–15K then B = 0.16
" >70K then B = 0.39

Productivity ⟹ Embedded Real Time System — 2000
System n/w or n/w based s/w – 10000
Scientific s/w – 12000
Business s/w – 28000

Duration ⟹ months / years

Effort ⟹ Person months or Person years.

Putnam & Meyer Method.

$$t_{min} = 8.14 \left(\frac{LOC}{P}\right)^{0.43} \text{ months}$$

$$\boxed{t_{min} \geq 6 \text{ months}}$$

$$\text{Effort } (E) = 180\, B t^3$$

t → yr
B → special skill factor.

$$\boxed{E \geq 20}$$

S/w equan is a multivariable estimation mdl used for predicting the efforts throughout the life of s/w development

S/w equan contains different parameters which include:

B - Spl. skill factor

If the size of the prjt changes, automatically complxty also changes related to integration, testing, quality assurance, human skills reqd for developmt so on.

P- Productivity

P doesnt remain constant for all categories of prjt, depending on type of projt it variates. Different const. values are assumd based on type of projt.

t duration

Depending on ~~type~~ size, the duration paramtr will be treated as months or years. Based on this, unit of effort changes to person-month / persn-year resp. Based on s/w equan, Putnam & Meyer derivd 2 formula for evaluation of minm time reqd for developmt of pdt. tmin & Effort equan assumd.

T-7

61. size = 60 KLOC
$E = 2.4\,(60)^{1.05} = 176.7\,pm$

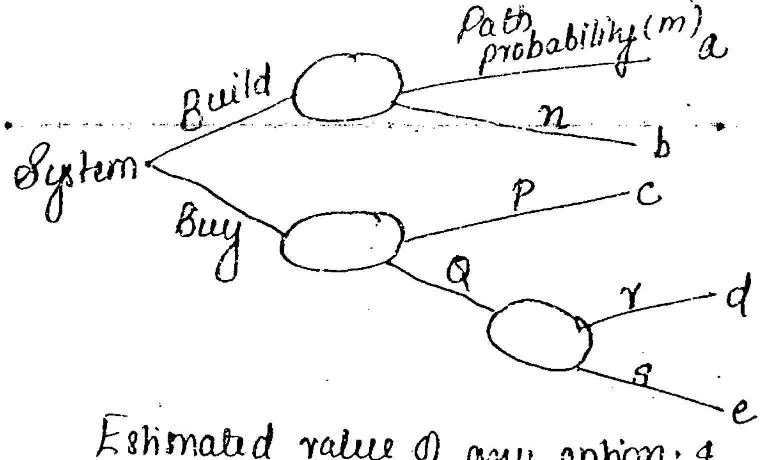
62. $E = 3 \times (60)^{1.12} = 294.21$

63. $E = 3.6 \times (60)^{1.2} = 489.87$

64. D = 2.5(

Decision Trees

Tool to help top level mngs to take decisions.



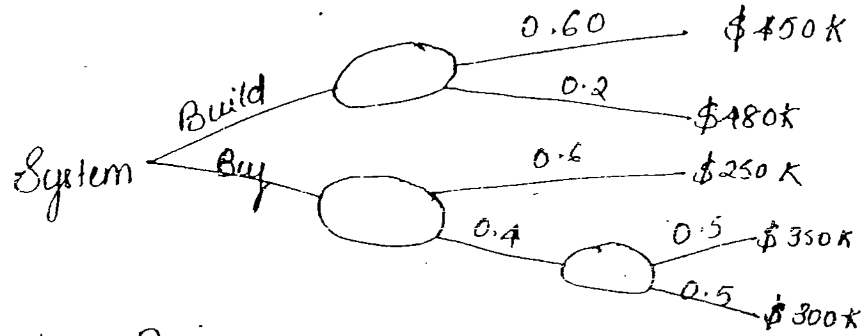
Tool used during s/w plang. It provides aid for top level mngrs in critical decisions during plang stage

Estimated value of any option; t

$$EV_x = \Sigma \text{ Path probability} * \text{Estimated cost.}$$

Eg: $EV_{build} = m(a) + n(b)$

$EV_{buy} = P(c) + Q[r(d) + s(e)]$



$EV_{build} = 0.80(450) + 0.2(480) = 456\text{ K}$

$EV_{buy} = 0.6(250) + 0.4[\cancel{250}\ 0.5(350) + 0.5(300)]$

$= \$280\text{ K}$

manager decides to buy.

~~For~~ s/w developmt involves 'n' no of tools, components apart from coding. So top level manager shd plan all these things prior to the development by chkg different options. A decision tree is used for evaluatg estimatd value for evry option for which 2 components are used which include path probabilities & estimated cost such that

EV can be derived by using formula

$EV_x = \sum$ Path Probability × Estimated Cost

·30

24. $EV_{build} = 0.3 \times 380K + 0.7 \times 450 = \$429K$

$EV_{reuse} = 275 \times 0.4 + (310 \times 0.2 + 490 \times 0.8)0.6$
$= \$382.4K$

~~$EV_{buy} = 210 \times 0.7 + 0.3 \times 40 = \$159K$~~

$EV_{contract} = 0.6(350) + 0.4(500) = \$410K$

Critical Path Method — Projt Evalutn & Review Technique

CPM and PERT

— Tool to determine the critical path.

CPM — deterministic — Fxd — Small pjt.

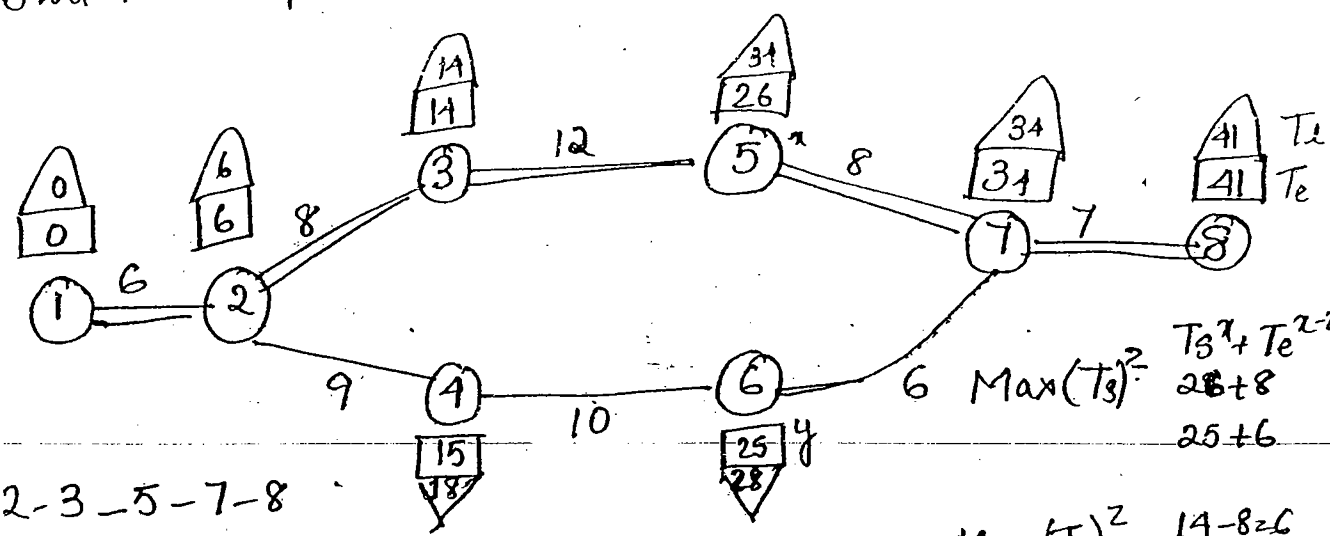
PERT — Probabilistic — Variable — large pjt.

S_{opt} S_{likely} S_{pess}

Two methods of computatn

Fwd Pass Computatn — Earliest time — ($1 \to n$) — Add — maxm

Bwd Pass Compn — Latest time — ($n \to 1$) — Subtract — min tym.



$T_5^x + T_e^{x-2}$

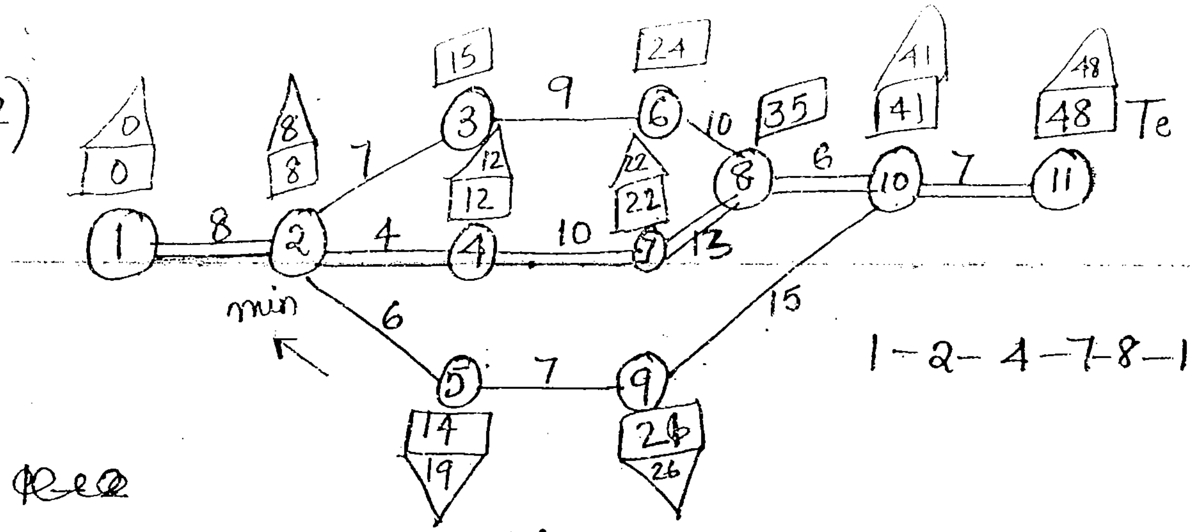
Max$(T_E)^2$: 26+8, 25+6

-2-3-5-7-8

Min$(T_L)^2$: 14-8=6, 18-9=9.

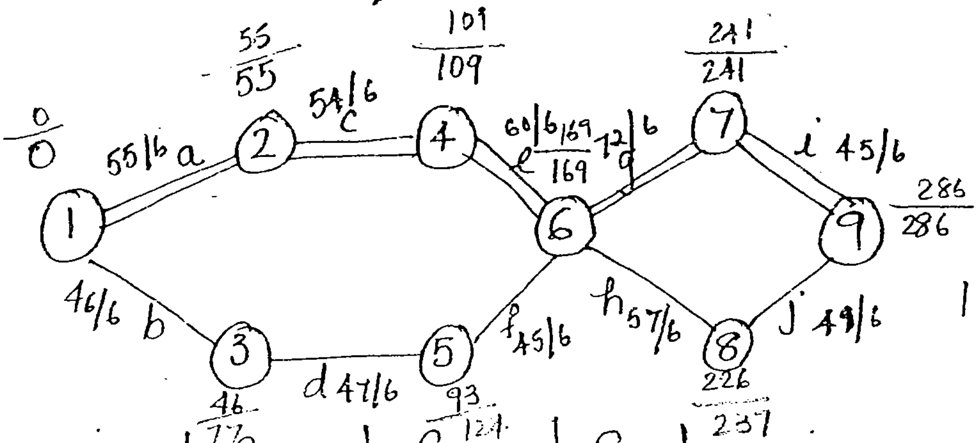
3 16

2)



1-2-4-7-8-10-11

3)



1-2-4-6-7-9

	S_{opt}	S_{likely}	S_{pess}	EV
a	8	9	11	55/6
b	6	7	12	46/6
c	5	9	13	54/6
d	6	8	9	47/6
e	5	10	15	60/6
f	6	7	11	45/6
g	10	12	14	72/6
h	8	9	13	57/6
i	6	7	11	45/6
j	7	8	10	49/6

CPM & PERT are tools used for determining critical path during planing stage. This tools are used for proper evaluatn of schedule. wrt events involved in a gvn system.

CPM is deterministic in nature where the resource b/w events are fxd. Generally used for small projects where requimts will be simple and obtaing unique requirements from this simple requiremt is not complex.

PERT

- probabilistic in nature
- the resource b/w events in that s/s is evaluatd based on 3 samples.
  - EV formula gvn by

$$EV = (S_{opt} + 4S_{lik} + S_{pess}) / 6$$

- generally used in large prjt where reqrmts will be simple as well as complex such that obtaing unique or distinct req. is difficult bcz the reqmts involves redundancy, conflicts disagreement
- Extracting unique reqmts from these is not smple task.
- Three samples are considered for evaluation.
- Steps in evaluating critical path

For evaluating critical path, two techniques are used —

Fwd pass computn
Bwd pass "

In Fwd Pass computn,
~~- it is required~~ to evaluate earliest tym from node 1 to last node in a gvn s/s.

- Two tym evaluatns are computed, includg.
  1) Earliest expected Start time. — 1^{st} node.
  2) Earliest " " ~~las~~ completion time — last node

- In FPC, earliest time is computed by addition opn while navigating from 1 event to another event, resource b/w them is added. This procedure continues till the last node is encountered. But care is taken while evaluating earliest time of imploded node.

$$\text{Max}(T_E)^Z \begin{cases} T_E^x + T_e^{x-z} \\ T_E^y + T_e^{y-z} \end{cases}$$

Earliest time is represented by rectangle and denote as T_E

— In BFC, latest time is evaluated from last node to first node in a gvn system such that 2 time evaluations are computed —
  1) Latest expected start time — last node
  2) Latest xpctd completion time — First node.

Latest time is represented by Δ & denoted T_L
— Bwd pass starts immediately after fwd pass and that the latest start time is initialized with

value obtaind from forwd pass.

While evaluatg latest tym, subtractn is performd while navigatg from last to first node, procedure continues till first node is encountered.

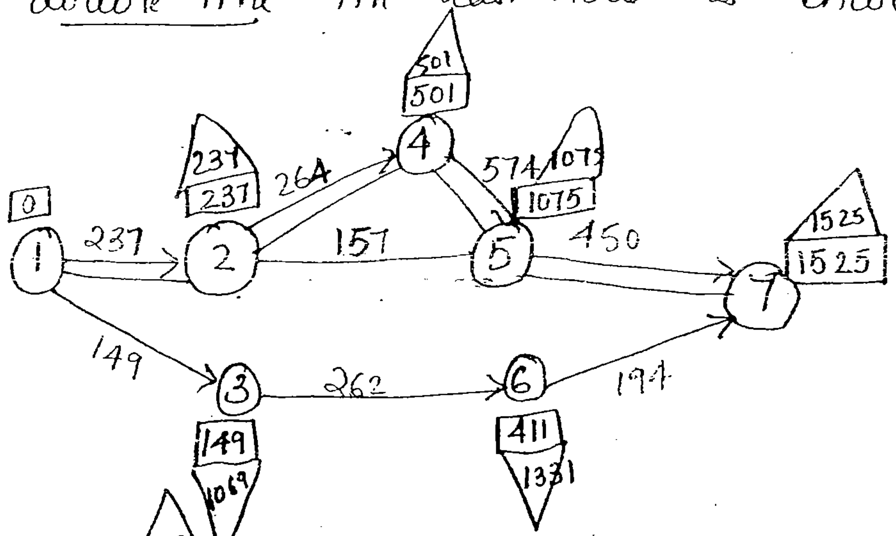
Care shd be takn while evaluatg latest tym of imploded node.

$$\underbrace{\text{Min }(T_L)^z}\left[\begin{array}{l} T_L^{x} - T_e^{x-z} \\ T_L^{y} - T_e^{y-z} \end{array}\right.$$

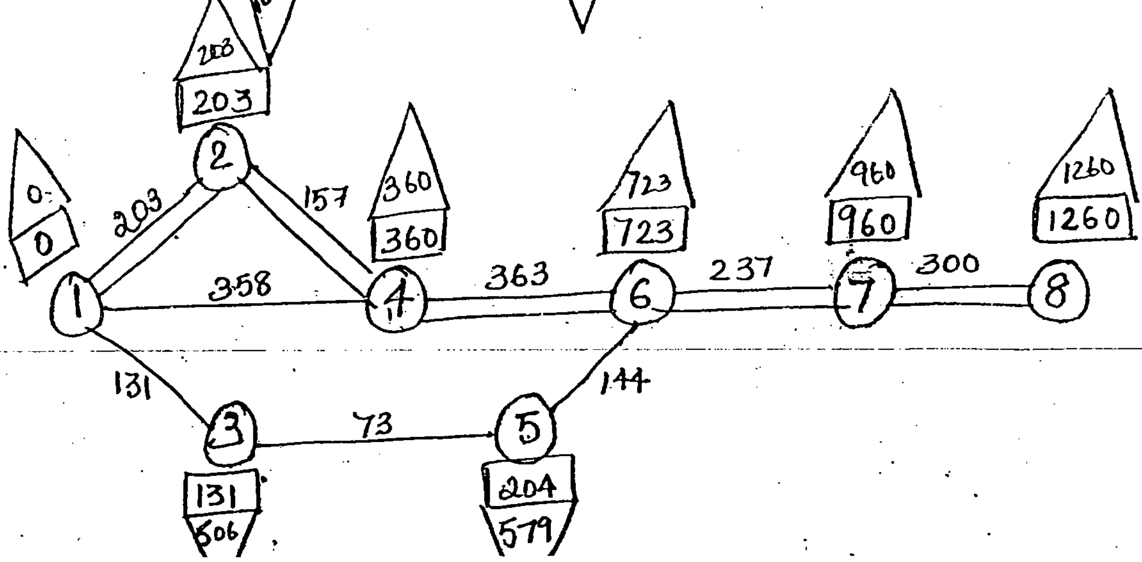
After completion of fwd and bwd pass, the latest and earliest time of every event is evaluatd. If both are same, that path is adapted, represented by double line till last node is encountered.

26

1.)



2.



Requirement Engg Process

System Specification
- SRS preparation
- Requirement document
- Specification document.
- System Requirement Specification Document.

Analysis model
- Info domain
- Func domain
- Behaviour domain

Requirements
- Services [Functional requirement]
- Operational constructs [non functional requirement] → expected requirements.

Functional + non functional: Domain requirement

Exciting requirements → accordg to stds of organisation.
eg: mobile phone & its new tech features

Non-functional requirement
- (Product based)
  - Quality
  - Reliability
  - Usability
- (Business based)
  - Delivery Requirmt
  - Implementatn Reqm.
  - Standard Reqm.
- (External based)
  - Interoperability
  - Ethical reqm.
  - Legislative reqm.
    - Privacy
    - Safety

User characteristics

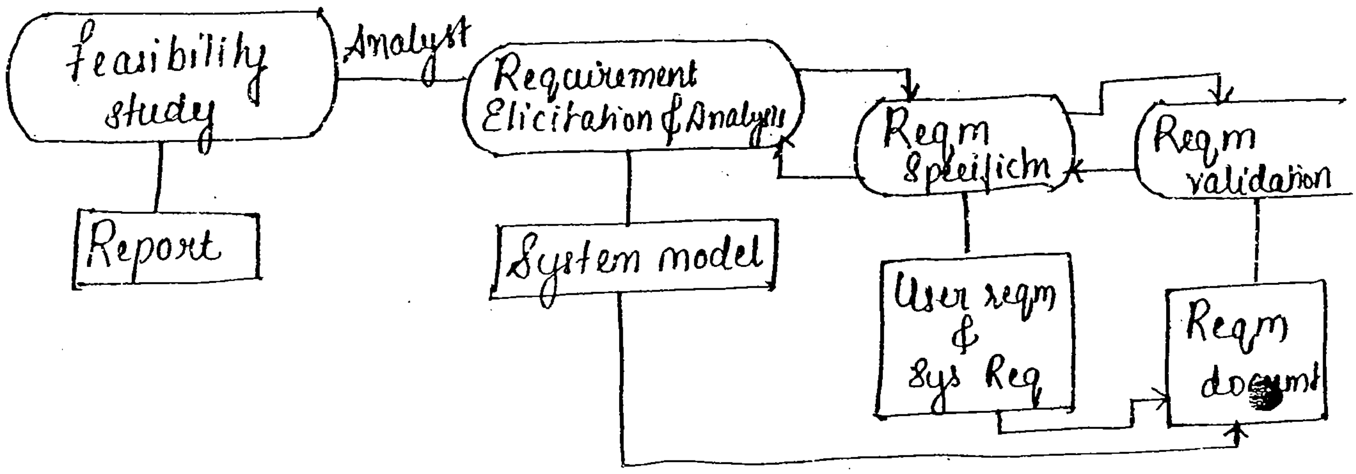
Novice – no syntax, no semantics
Knowledge based Intermittent
Semantic but not syntax

Integrity of a s/e depend on security & threat.

$$\text{Integrity} = 1 - \text{Security} \times 1 - \text{Threat}$$
$$(0.1 - 1) \qquad\qquad (0.1 - 1)$$

Updates are provided for continuous security.

5.

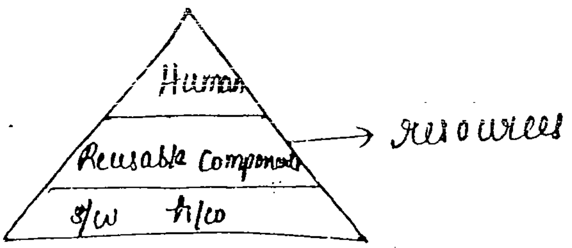


Feasibility study

- (Orga)nisation — Technically feasible - resource evaluation
- Economically feasible - profit evaluation
- Client → Operationally feasible - functionality evaluation

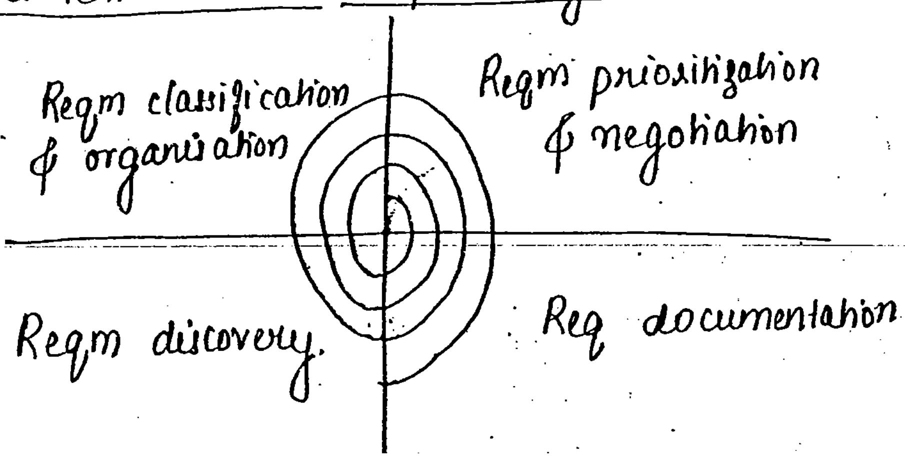
} when all are +ve, org. is in a position to develop pdt

→



→ Profit is directly prop to resources.

Requirement Elicitation & Analysis



Reqm — Primary / Secondary

Primary	Secondary
Mandatory	optional
100 no of Reqmts then set prios to each	80

Reqm discovery or Reqm Gathering

- Viewpoints
- Scenarios — Structured Specif. lang — UML
- Interviews
- Ethanography

Reqm (problems)

- Redundancy
- Ambiguity
- Contradictory

Framework used to overcome called viewpoints

Distinct Unique Reqm

Viewpoint

- Indirect
  - librarian
  - Mgmt
  - Supplier
- Direct
  - User
    - Students
    - External
    - Staff
  - library staff
    - Cataloguers
    - lib mngr
- Domain
  - Classification

DDC - Devey Decimal Classificatn (macro level)
UDC - Univ. Decimal " (micro level)

Interviews

- Open Interviews - Unstructured - Questionaires → not preferred Reqm dunt be complt
- Closed Interviews - Structured

FAST — Facilitate Appln Specificatn Technique.

- recorder
- Developer
- Venuer
- Customer.

Recorder prepares at SRS

Ethanography

— used by analyst.

— speak ~~at~~ the level of the customer.

IEEE SRS

1.0 Introduction

1.1 Purpose of Req document

1.2 Scope of Pdt.

1.3 References

1.4 Defns, Abbreviations

1.5 Remainder part of reqm document.

2.0 General Descriptions

2.1 Product Perspective

2.2 Product Functionality

2.3 User characteristics.

2.5 Assumptions & Dependancies

2.4 General constraints

3.0 Specific Descriptions

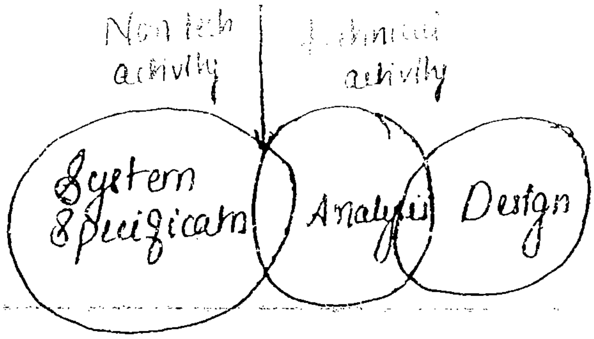
3.1 Functional Reqm.

3.2 Non-func reqm

3.3 domain reqm

4.0 Appendices

5.0 Index



40 — 20 — 40

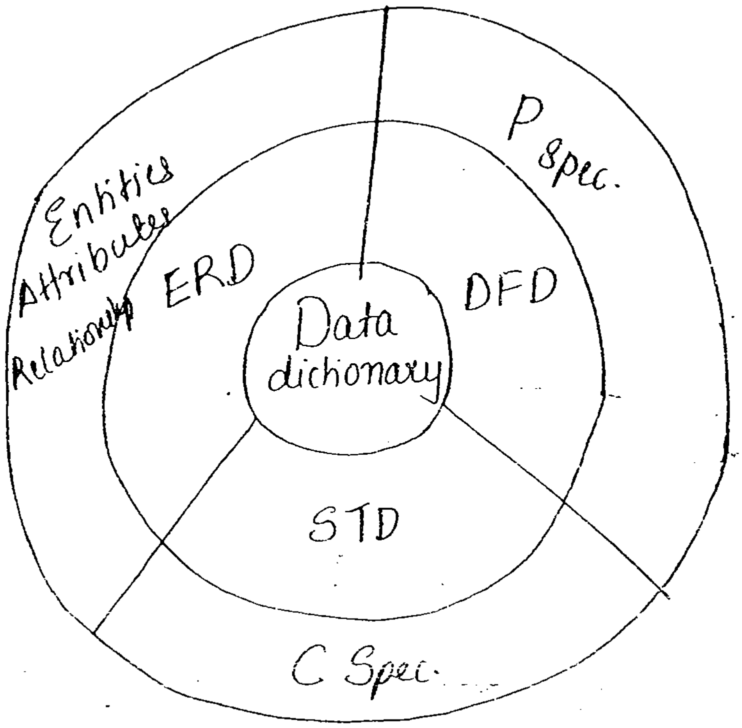
$\boxed{16}$ — 8 — 16

4% — 12%

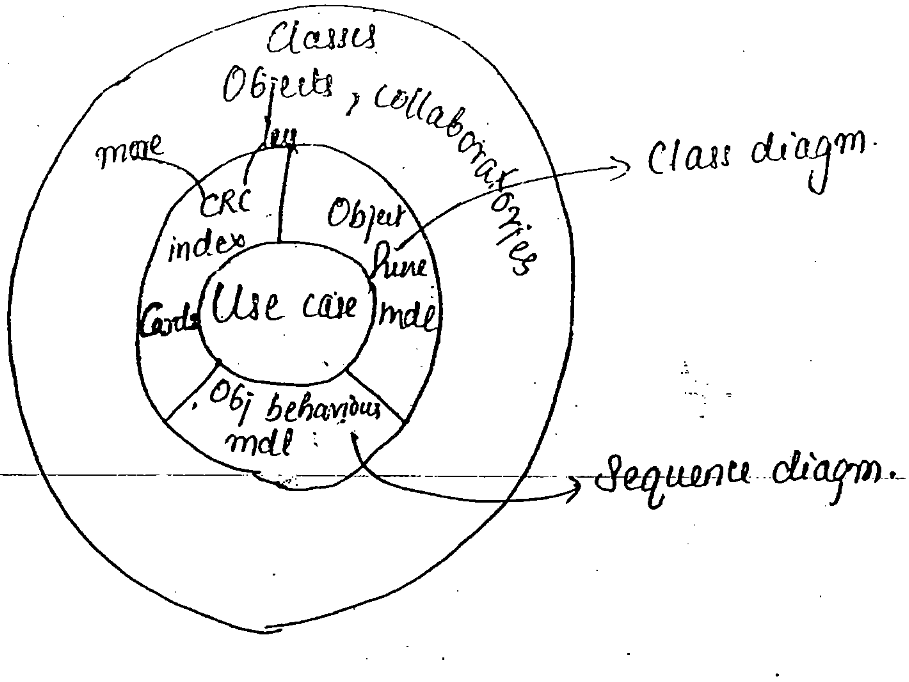
Analysis Model

- Structured Analysis model
- Obj Oriented Analysis model

Structured



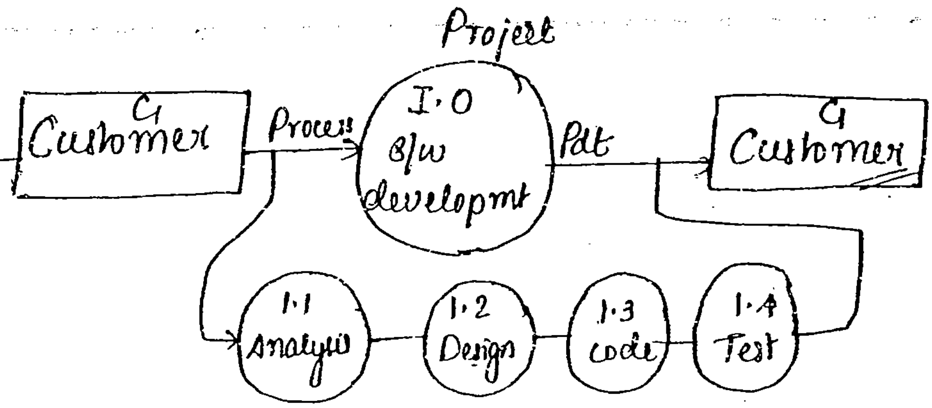
Obj Oriented



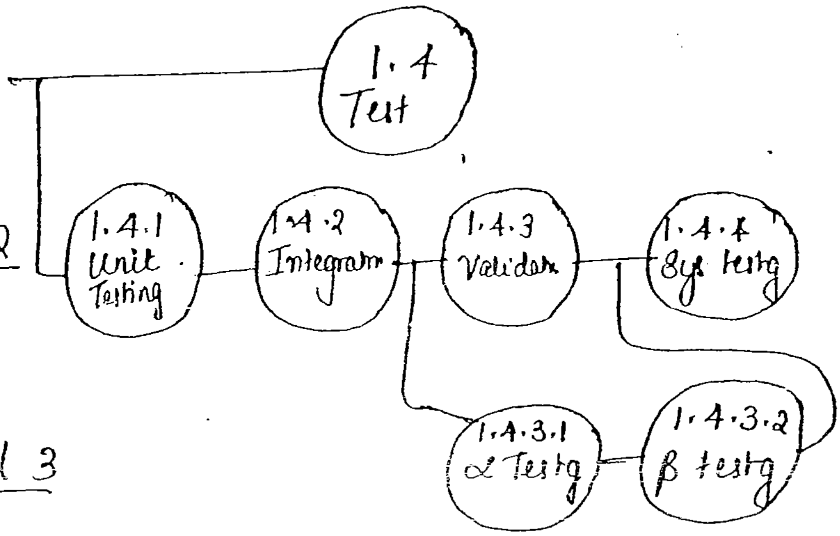
ERD — Data base reqm
UML — S/w reqm.

Data flow diagrams — Transformations — Bubble Charts

DFD level 0 context diagm



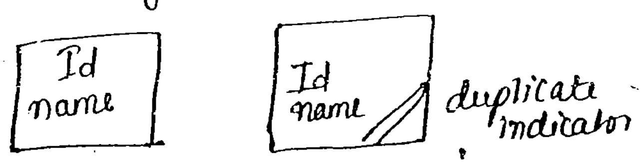
Level 1



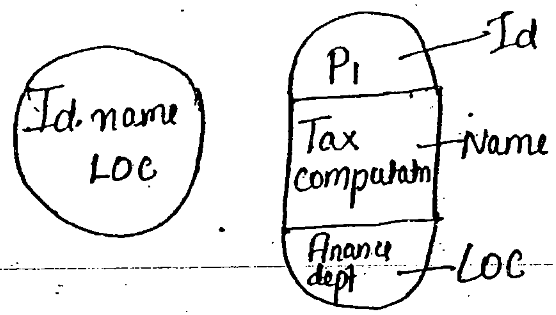
level 2

level 3

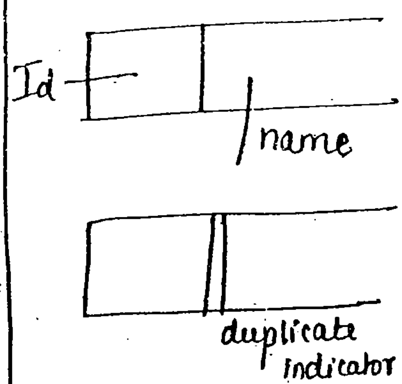
External Agent — Human, Subsystem



Process



Datastore (file)



Dataflow

Id →

Dataflow diagrams are used to represent transformation of information.

It is also referred as bubble charts.

The followg steps are followd:

1) Any s/s is represented by a single bubble, which shows abstract view of the pdl (context diagram/level 0 DFD)

2) Identify primary i/ps and o/ps from a keyprocess

3) Refine the key process into subprocesses such that internal details are shown. This refinement should not exceed 1 : 5.

4) Refine a single bubble at a time and also provide proper labels for dataflow. Otherwise, it leads to specification errors. After completion of final DFD for every process, a clear process specification shd be written.

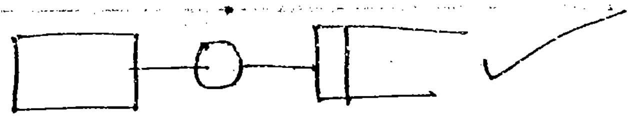
Guidelines for construction:

1) Dataflow diagram provides transformation bt not procedural aspects which involves selections & repetitions

2) Its not reqd to show the starting and endg of the structure when a grn process is refined.

3) At a gvn time, only one process shd be decompsd

4) An external agent and file cnt be connected directly

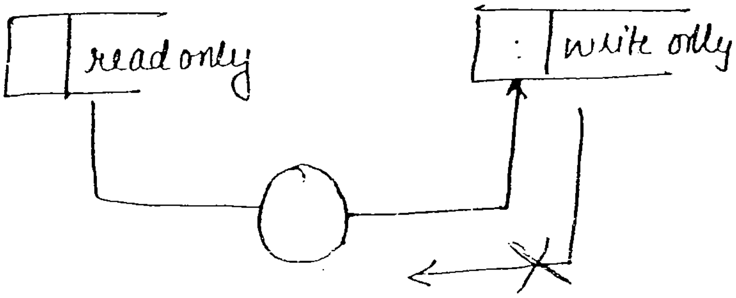


5) Two external agents cnt be connected directly.

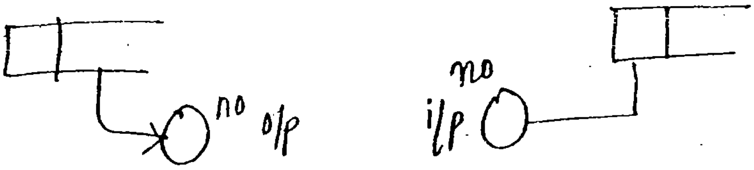
6) Two datastores cnt be conntd directly.

7) In dataflow diagms all notation should be connectd thru dataflow such that synchronization b/w notations will be provided.

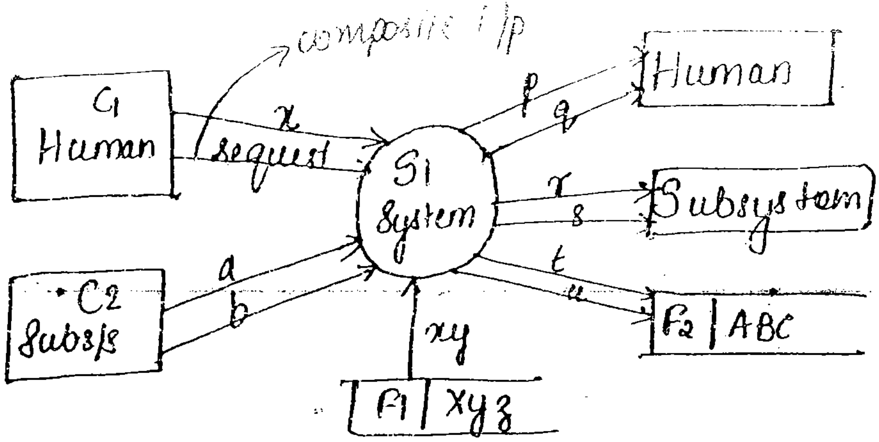
8) Ensure that a process will read from read only file & stores info in write only file.



9) Ensure that certain processes reads information from file but doesnot produce output & also certain process does not read but inturn produce o/p.



10) Ensure that a process request for the info available in file / datastore but not beyond that.



I/p area

— External i/ps ┌ human —— 2 ┐
└ subsystem —— 2 ┘ 4

— Inquiries ———— 1

— File ———— 1

O/p area

— External o/p ┬ human —— 2
├ subsystem —— 2
└ external file —— 2

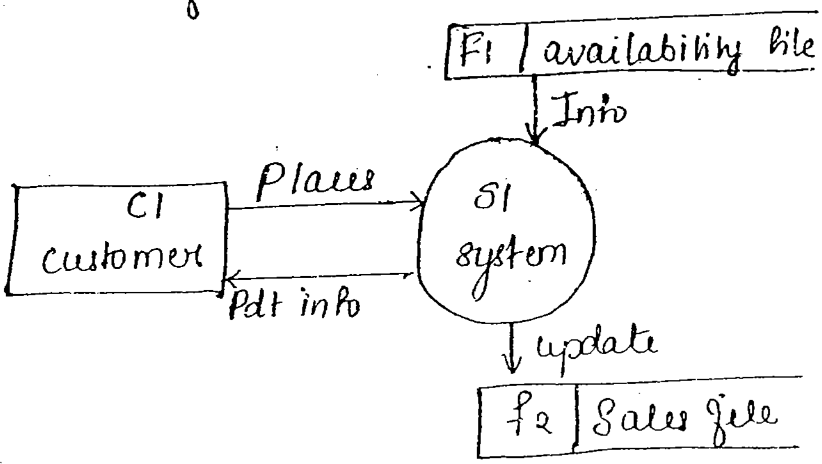
— External interface ┬ subsys —— 1 ┐
└ external file —— 1 ┘ 2

p 28 table

	EV		Simple	Avg	complex
No. of i/ps	4	×	3	4	6
no. of o/ps	6	×	4	5	7
no of inquiries	1	×	3	4	6
no. of files	1	×	7	10	15
no. of External interface	2	×	5	7	10
			56	74	107

A customer places a sales order & s/s chks for availability from available file based on availablity info sent to customer, as well as sales file is updated

Customer - Ext agent
system - Process
Availability - Datastore
Sales file - Datastore



i/p area

External i/p [human 1, sub s/s 0] } 1

Inquiries —— 0

File —— 1

o/p area

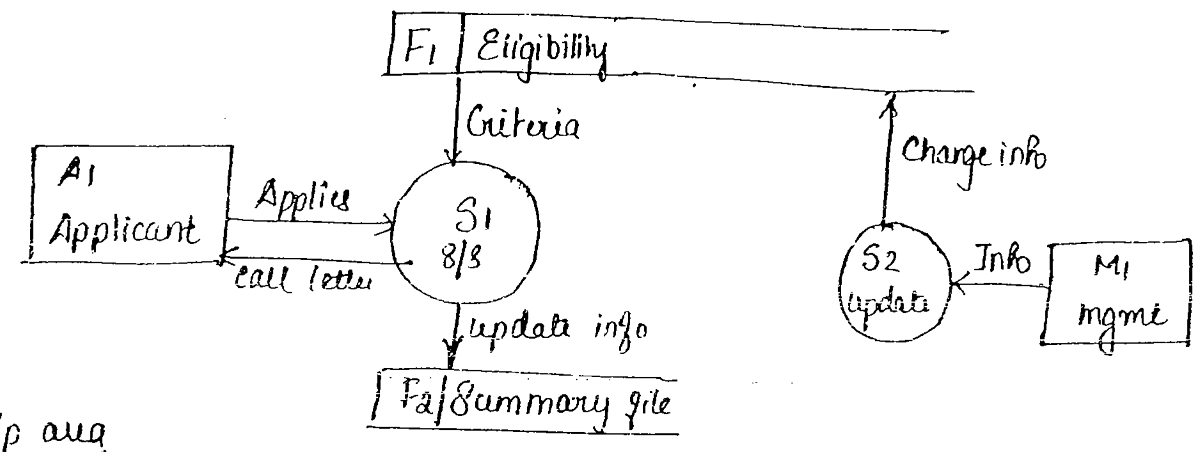
Ext. o/p [human —— 1, subs/s —— 0, Ext. file —— 1] } 2

Ext interfa [Subsys —— 0, Extern.file —— 1] } 1

	EV		Simple	Avg	Complx
no. of i/ps	1	×	3	4	6
no. of o/ps	2	×	4	5	7
of inquiries	0	×	3	4	6
of files	1	×	7	10	15
of Ext. interfa.	1	×	5 /	7 /	10 /

An applicant applies for a job, a s/s checks for the eligibility from eligib. file based on which an applicant receives (23) call letter. and also, updates info into summary file which contains a list of candidate called for interview. Dependg on org. s/s scenario, the mgmt can change eligibility criteria using update process.

Applicant – ext. agent
s/s – Process
eligibility file – Datastore
Summary file – Datastore
Management – Ext. agent
update – Process



i/p area

Ext i/p: human — 2, sub — 0 } 2

inquiry — 0

File — 1

o/p area

Ext. o/p: Human — 1, Subs/s — 0, Ext. file — 2 } 3

Ext. interface: subs/s — 0, Ext file — 2 } 2

no. of i/ps	2	3	4	6
outpts	3	4	5	7
inquiries	0	3	4	6
files	1	7	10	15
External	2	5	7	10
		35	47	68

next ~~B~~·53

44. human — 0 } 2 ——— 0 } 1
subsys — 2 } ——— 1

ing — 0 ——— 1
file — 1 ——— 1

Human — 2 } 4 ——— 0 } 3
sub — 1 ——— 1
ext. file — 1 ——— 2

subsys — 1 } 2 ——— 1 } 3
ext. file — 1 ——— 2

		53			
no. of i/ps	2	1	3	4	6
o/ps	4	3	4	5	7
inq	0	1	3	4	6
file	1	1	7	10	15
Ext. file	1	2	5	7	10
			34	45	65.
			35	47	68

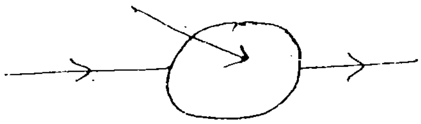
~~50.~~

~~i/p~~ 8
~~o/p~~
~~inq~~
~~file~~
~~Ext. file~~

2/08/10
6.15 – 8.15 – probability
test 03
8.30 – 9.00 – Process mgmt

Process Specification (P specification)

/08/10
Friday



show the details of i/p & o/p.

Base for the project.

- process specification shd be precise, verifiable & understandable.

- users of proc. spec : analyst himself. — analysis model
  - Designer — design model
  - Practitioner — implementation
  - S/s documentation

- techniques to write proc. spec.
  - narrative English → over flexible, lengthy, time consuming
  - Structured English.
  - Pseudocode.
  - Pre/post condition chart.

  } non graphical tools.

  - Flowcharts.
  - ~~Decision tables.~~

  } graphical tools.

→ non graphical tools not used in s/w companies
→ If condns are more, flow charts also not used.

pre/post condtn chart

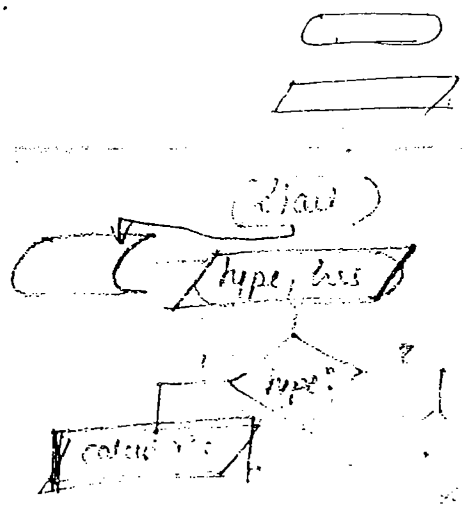
Pre condtn
←
Post condtn

Case study:
Pay roll application

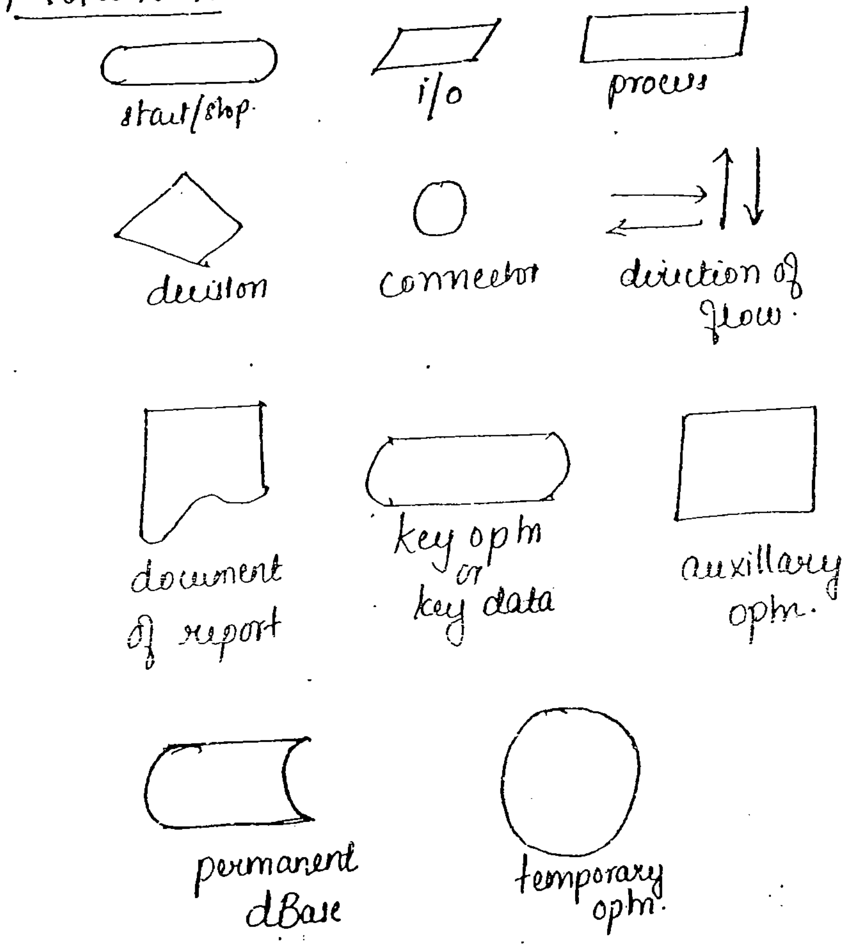
Type of employee — salaried / hourly worked.

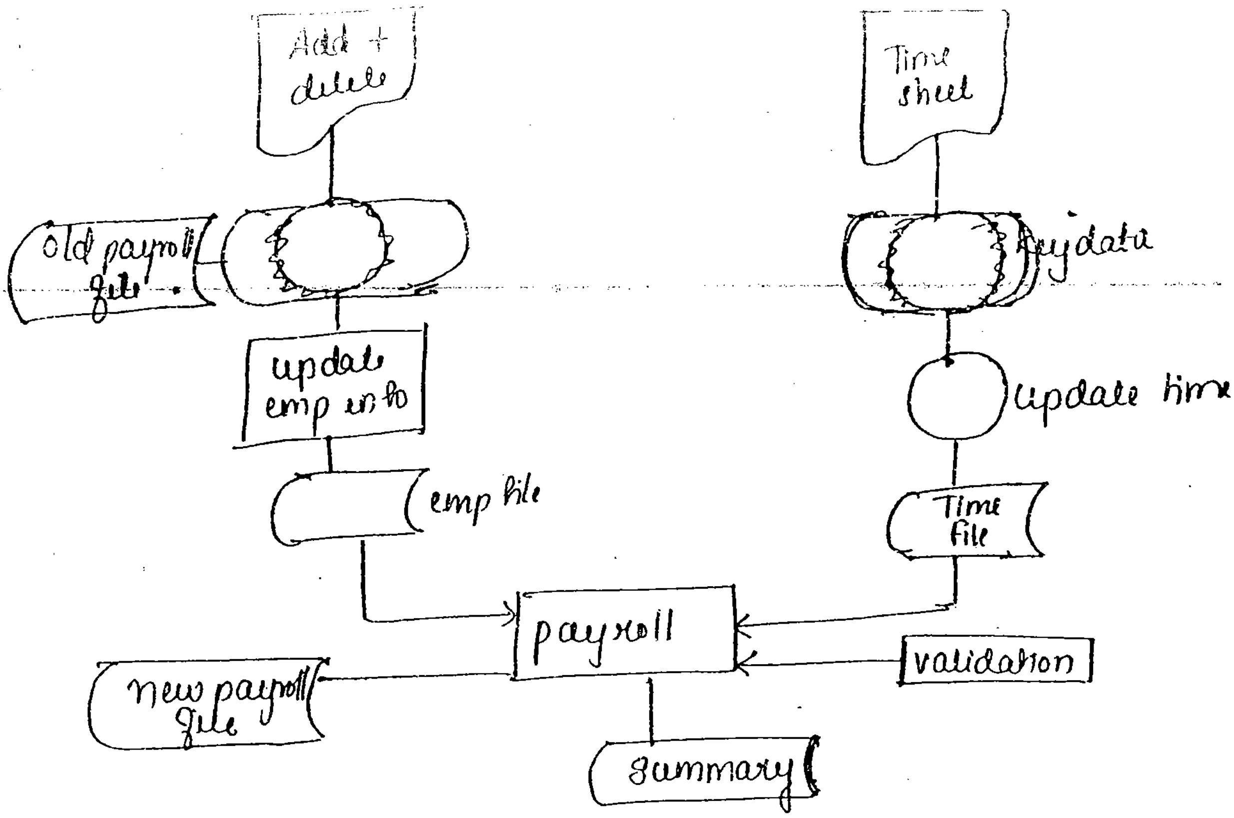
hrs worked — 40 hrs.

- salaried — { =40, <40, >40 } Basic pay.
- hourly waged —
  - =40 — salary
  - <40 — absent
  - >40 — overtime



Notations





Decision Tables:

Condtn Stub	Rules.
Action Stubs	Actions

$2^2 = 4$ — 6

Type of employee	S	H	S	H	S	H
Hours worked	40	40	<40	<40	>40	>40
Basic salary	X		X		X	
Compute salary		X				
ABsent				X		
overtime						X

Decision tables are widely used graphical tool for representing process specification. It includes 3 parts evaluated by analyst. i.e. condtn stubs, action stub & rule.

Condtn stub - determines a list of condtn involved in a gvn system.

action stub - determines a list of actions performed on a gvn condtn of the s/s.

rule - specify what action must be performed on a gvn condtn of the s/s.

<u>guidelines</u> <u>for constructing</u> <u>decision tables</u>:

1) a decision table, shd not contain redundant rules (same rules with same actions), ambiguous rules (same rules with diff actions), contradictory rules (disagreement rules, pair of ambiguous rules leads to disagreement)

2) a decision table will be optimal if the no of rules is equivalent to 2^n where n is no of conditions involved.

3) If the rules are redundant then, they can be modelled concurrently.

ie. executing redundant rule simultaneously doesnt change state of the system.

4) All the rules in a decsn table shd be complete otherwise, it leads to specification errors.

Case study (2) : Airways reservation

Business class [request, availability] } ticket reserved, pending.

Tourist class [request, availability] } ~~tkt~~ reserved, pending.

rule. Kmaps. $2^2 = 4$.

Business class request	Y	Y	N	N
Business class availability	Y	N	N	N
Tourist class request	N	N	Y	Y
Tourist class availability	N	N	Y	N
Business class reserved	✓			
Business class pending		✓		
Tourist class reserved			✓	
" " pending				✓

Decision Table:

- Redundant rules – same rules same action
- ambiguous rules – same rule diff actions
- contradictory rules – disagreement

2 pairs ambiguous pairs.

5) If the rules are complete then decision table can be converted to K-map and viceversa is only posible when the entries in Kmaps are not blank.

6) In case for a gvn rule, if it contains more than 1 action then action shd be separated by delimiters (" for ...... action & "," for

Petrinets:

— modelling & Evaluation — States / events

petrinets is a widely adapted tool in analysis phase for modelling and evaluating a s/s based on no. of states and no. of actions

Two properties are used for construction of petrinets which include.

1) condition
2) event.

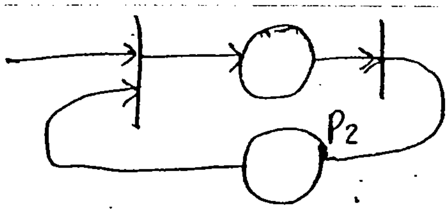
A condition is a boolean description which determines the state of the s/s. which is represented by place & denoted by circle.

Event specifies what action shd be performed on a given state. represented by transition & denoted by vertical bar. (|)

Petrinet is multi variable tool used for modelling a s/s. denoted by C.

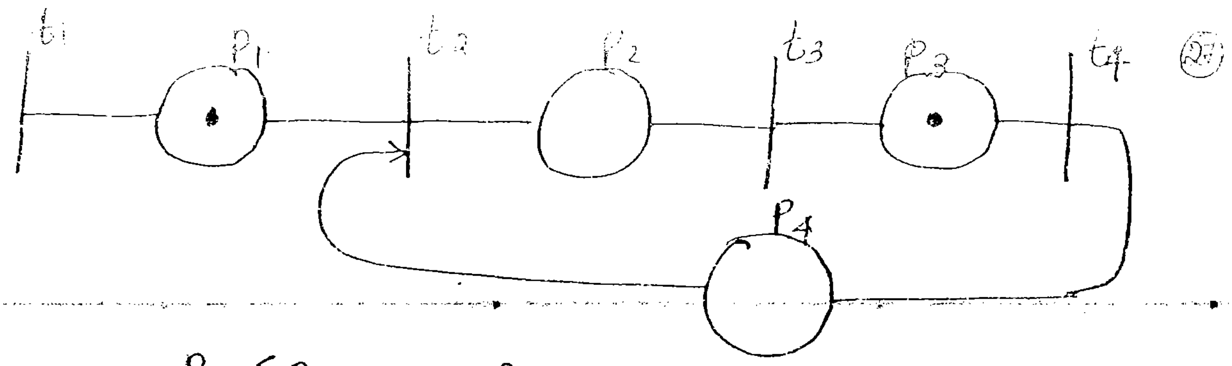
$$C = (P, t, I, O)$$

where P = no. of states.
t = no. of transitions
I = i/p place for transition
O = o/p place for transition



$P = \{P_1\ P_2\}$
$t = \{t_1\ t_2\}$
$I(t_1) = \{P_2\}$ $I(t_2) = \{P_1\}$

Case Study 1 (27)



$P = \{P_1\ P_2\ P_3\ P_4\}$

$t = \{t_1, t_2, t_3, t_4\}$

$I(t_1) = \{\phi\}$ $I(t_2) = \{P_1\ P_4\}$ $I(t_3) = \{P_2\}$

$I(t_4) = \{P_3\}$

$O(t_1) = \{P_1\}$ $O(t_2) = \{P_2\}$ $O(t_3) = \{P_3\}$

$O(t_4) = \{P_4\}$

Steps of evaluation/executing of petrinets

Step 1: Identify any transition in a gvn s/s is enabled.

Step 2: A transition is enabled iff all i/ps contains a token in it.

Step 3: An enabled transition fires. i.e. the transition removes all tokens from i/p places & deposit them at o/p places.

Step 4: Procedure continues until unless ~~s/s~~ reaches ~~hold~~ halt state. i. wen all transitions are disabled, sys. halts.

A petrinet is evaluated based on certain properties which include safeness

1) petrinet is safe., if it contains tokens in it ie. if a place contains a token, will have value 1, otherwise value 0.

If all places has value of 0, then petrinet

2) A petrinet is <u>bounded</u> only if the no. of tokens received by a place at a grn time shd not exceed 'n' integer value.

3) A petrinet is <u>conserved</u> if the no. of token in it remains constant, ie, neither tokens created nor destroyed.

4) <u>Reachability</u> is the primary property of petrinet which determines whether every node is visited atleast once throughout the course of execution.

5) <u>Coverability</u> determines the no. of subset of paths involved in a grn petrinet.

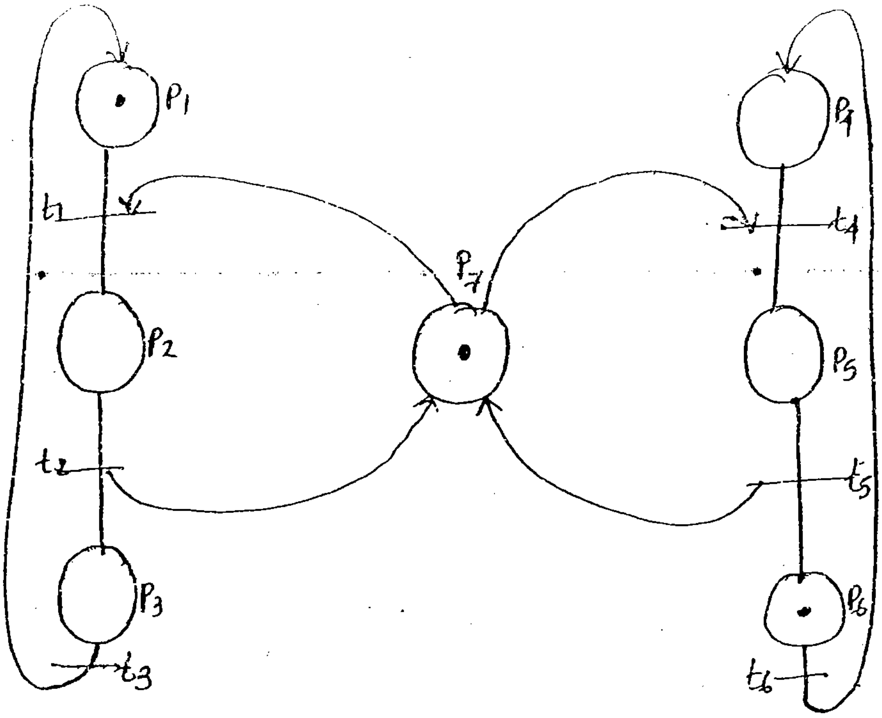
6) <u>Liveness</u> determines whether every transition is enabled atleast once during the course of execution.

Conversely, if one or more transitions are disabled throughout execution, then it will be in deadlock state.

, i/p transition
bled

Two techniques are used to evaluating this parameters.

1) reachability tree.
2) matrix equan.



$P = \{ P_1\ P_2\ P_3\ P_4\ P_5\ P_6\ P_7 \}$

$t = \{ t_1, t_2, t_3, t_4, t_5, t_6 \}$

$I(t_1) = \{P_7, P_1\}$ $\quad I(t_2) = \{P_2\}$ $\quad I(t_3) = \{P_3\}$

$I(t_4) = \{P_4, P_7\}$ $\quad I(t_5) = \{P_5\}$ $\quad I(t_6) = \{P_6\}$

$O(t_1) = \{P_2\}$ $\quad O(t_2) = \{P_3\ P_7\}$ $\quad O(t_3) = \{P_1\}$

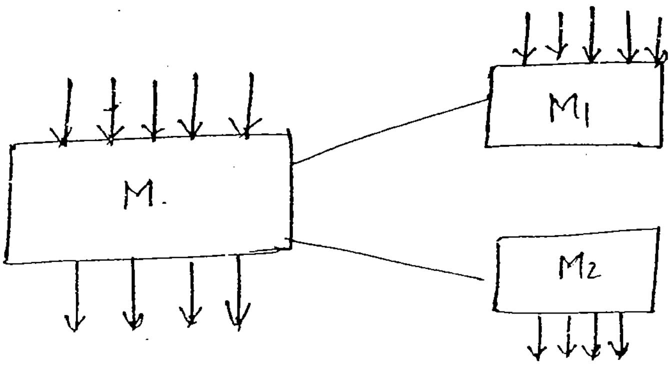
$O(t_4) = \{P_5\}$ $\quad O(t_5) = \{P_6\ P_7\}$ $\quad O(t_6) = \{P_4\}$

Design activity

- modularity

Meyer's properties

- modular decomposability.
- modular composability
- " understandability
- " continuity
- " Protection.



Henry & Kafora

Fan in & Fanout

$L\,(\text{Fan in} * \text{Fanout})^2$

length.

$$L(5*4)^2 = \frac{L}{2}(5*x)^2 + \frac{L}{2}(x*4)^2$$

$$\cancel{L}\,400 = \frac{\cancel{L}}{2}(25x^2 + 16x^2)$$

$$800 = 41x^2$$

$$x = \sqrt{\frac{800}{41}} = 4.41 \approx 4$$

08/10 sday.

Architectural Design Metric

Structural complexity. S(i)

Data complexity. D(i)

System complexity C(i)

$S(i) = \text{fanout}^2(i)$

$S(A) = 4$

no of ips & o/p.

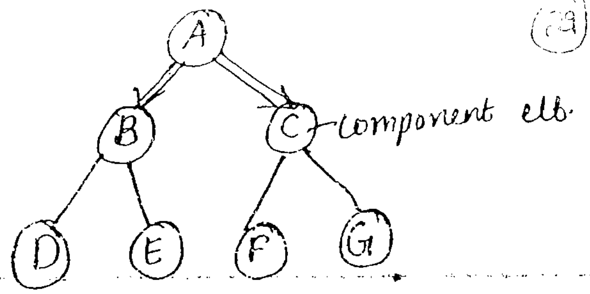
$D(i) = V(i)\,[\text{fanout}(i) + 1]$

I/O

{simple}

$D(A) = 2[2+1] = 6$

$D(B) = 3[2+1] = 9$



$C(i) = S(i) + D(i)$

$C(A) = 4 + 6 = 10$

$C(B) = 4 + 9 = 13$

In highlevel lang, the constructs are:

sequential stmt

selective stmt — bidirection — if else
 — multidirection — switch

Repetitive stmt — entry control stmt — while, for
 — exit control stmt — do-while

for loop is powerful construct.

Functional independence

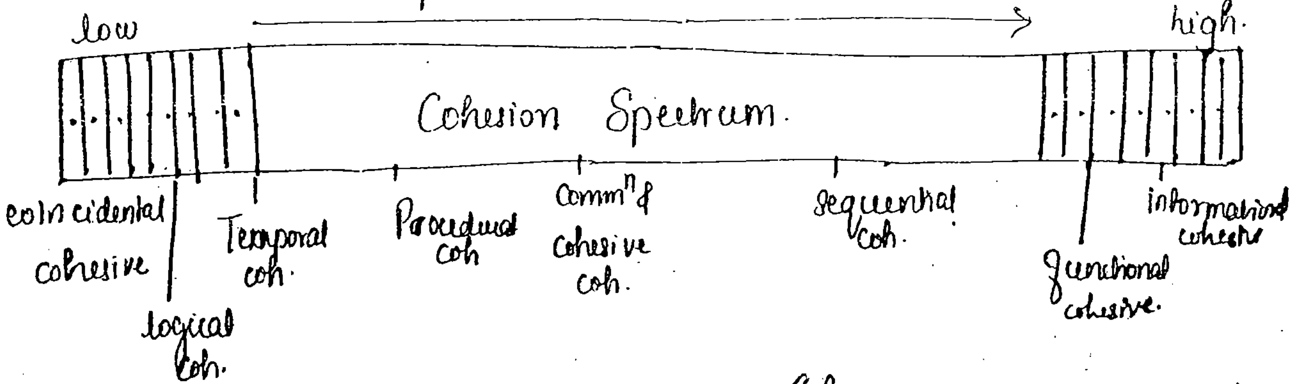
Cohesion metric — internal strength. (shd be more)

Coupling metric - comm^n b/w modules for integration. (shd be less)

Types of Cohesion & Coupling

Cohesion

Spectrum.



Cohesion metric → high

Coupling " → low.

Procedure v/s function

every var. confined to one name (abc) and perform a single task. — fn.

every var perform 'n' no. of ~~task~~ opns.

abc()
{
}

Seque.

o/p of one opn is i/p of another. opn.

opns are one after other.

Info.

If the elts are abstracted.

Cohesion and coupling are the two metrics used by designer to chk or achieve fnl independence.

Cohesion is a measure of internal relative strength of a module. whereas Coupling is the measure of interdependency amg modules. The main objective of modularity is to achieve fnl indep. with high cohesion & low coupling. ie. to minimize fanin and fanout, also referred as reducing the control complexity.

Cohesion is not uniform for every module. It is evaluated based on internal elts in a module. It is categorised into different types based on strength of the module as shown in cohesion spectrum.

(i) Coincidental cohesion

If elts of a module are unrelated, then it is coincidental cohesive.

(ii) Logical cohesive:

If elts of a module are related, and

said to be logical cohesive

(iii) Temporal: cohesive:

If elts of a mod are related and elts are confined to initialization or time, then its mod temporal cohesive

(iv) Procedural cohesive:

If the elts are confined to one name & if they perform set of operations, then the mod is said to be procedural cohesive.

(v) Commnl cohesive:

If the elts in a mod interact through data declared in it, then the mod is said to be commnl cohesive.

(vi) Sequential cohesive:

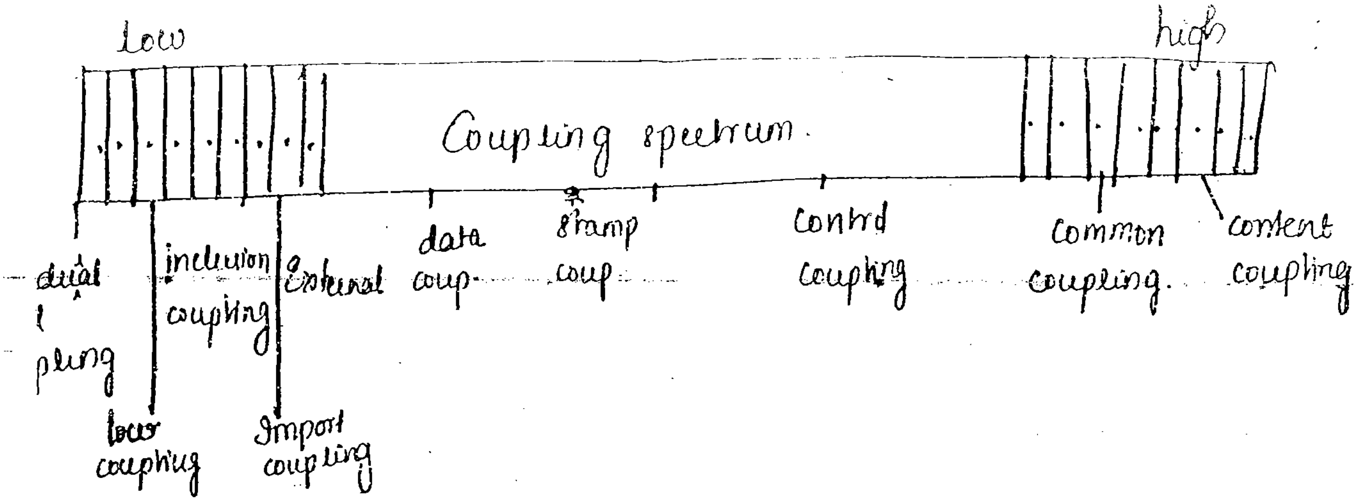
If the elts are related and if they perform set of opns in which the o/p of one opn will be input of another logical opn.

(vii) Functional cohesive:

If the elts are related & if they are confined to one name and if the perform one & only 1 task, then the mod is functional cohesive.

(viii) Informational cohesive:

If elts of a mod are confined to abstraction, then the mod is informational cohesive.



Coupling is broadly categorised into no. of types as in coupling spectrum. Coupling increases based on weakness of the module. It includes

1) procedural call coupling.
A form of coupling in which modules interact nominally more or less they are almost independent

2) Low coupling
Form of coupling in which mods interact minimally. In extreme case ~~no.~~ no coupling b/w them.

3) Inclusion coupling
A coupling in which source code of 1 mod is included in another mod.

4) Import coupling
A coupling in which one mod is declared in another mod for its functionality.

5) External coupling
A coupling in which modules interact with modules written by 3rd party. which may include specific hardware or s/w.

6) Data coupling
If the interaction b/w mod occurs thru simple or homogenous data, which include var, parameter

7) Stamp coupling:

If interacn b/w mod is thru composite or heterogen data, then it is said to be stamp.

8) Control coupling:

Coupling in which one module controls the order of execution of other mod. by using flags.

9) Common coupling:

If the mods interact thru, common sharable dBase, thn ⇒ common coupling.

10) Content coupling:

Type of coupling in which one module refers to other module, in ~~the~~ extreme case, it changes internal structure of other mods for its functionality.

Code activity

08/10 tuesday

Halstead's s/w science

OO CK metrics suite

Halstead's s/w science

operators, operands

```
int i,j,k,l,m,n;
for (i=0; i<=10; i++)
  {
    if (i>5)
      {
        pf ("%d", i);
        break;
      }
  }
```

operators		Operands	
int , ; , , , , ;	7	i j k l m n	6
for (= ; <= ; ++)	8	i 0 i 10 i	5
{	1	i 5	2
if (>)	4	i	1
{	1		
pf (" %d " ,) ;	8		
break ;	2		
}	1		
}	1		

$N_1 = 33$ $\quad N_2 = 14$

$n_1 = 17$ $\quad n_2 = 9$

$n_1^* = 12$ $\quad n_2^* = 8$

Basic primitives:

N1

N2

n1

n2

Additional primitives

n_1' or n_1^*

n_2' or n_2^*

N1: The total no. of operators in a gvn source code.

N2: The total no of operands in a gvn source code.

n_1: No. of unique or distinct operators in a gvn source code.

n_2: no. of " " " operands in a gvn source code

n_1' or n_1^* : no of single appearance of operators in a gvn source code.

n_2' or n_2^* : " " " operands " "

Basic metrics Steps for evaluating conventional pgmmg project

1) Vocabulary metric

$$\boxed{n = n_1 + n_2}$$

2) Implementation or pgm length

GA →

$$\boxed{N = N_1 + N_2}$$

I) Length equan N'

$$\boxed{N' = n_1 \log_2 n_1 + n_2 \log_2 n_2}$$

II) Quantification of intelligent content

1. Pgm volume (V)

$$\boxed{V = N \log_2 n}$$

2. Pgm level (PL or L)

$$\boxed{L = \frac{\text{Potential Volume}}{\text{Pgm Volume}} = \frac{V^*}{V}}$$

3. Pgm level equan (PL' or L')

$$\boxed{L' = 2/n_1 \times \frac{n_2}{N_2}}$$

4. Intelligent content (I')

$$\boxed{I' = L' \times V}$$

III Pgmmg time:

1. Potential volume (V^*)

$$\boxed{V^* = (2+n_2^*) \log_2 (2+n_2^*)}$$

② Effort Equan (E)

$$\boxed{E = V/L \quad \text{or} \quad V^2/V^*}$$

③ Time Equan (T')

$$T' = n_1 N_2 \left(N' \log_2(n)\right) / 2 n_2 S$$

$$\boxed{S = 5 - 20}$$

④ Pgmmg time

$$\boxed{P' = E/S}$$

Eg:

Basic metrics

① Vocabulary

$n = 18 + 9 = \underline{27}$

② Impl. $N = N_1 + N_2$

$= 33 + 14 = \underline{47}$

I $N' = n_1 \log_2 n_1 + n_2 \log_2 n_2$

$= 18 \log_2 18 + 9 \log_2 9$

$= \underline{103.58}$

$\log_a b$

$\frac{\log b}{\log a}$

II ① $V = N \log_2 n$

$= 47 \log_2 27$

$\frac{\log 9}{\log 2}$

$= \underline{223.47}$

② Pgm level :

$L = \frac{28.57}{223.47} = \underline{0.127}$

③ Pgm level Equan (PL' or L') $L' = 2/n_1 * n_2/N_2$

$$L' = \frac{2}{18} \times \frac{1}{14} = 0.071$$

(4) Intelligent content (I')

$$I' = L' \times V$$

$$I' = 0.071 \times 223.47$$

$$I' = 15.96$$

(III) Pgmmg Time.

(1) Potential volume (V^*)

$$V^* = (2 + n_2^*) \log_2 (2 + n_2^*)$$

$$V^* = 2 + 8 \log_2 (2+8)$$

$$= 28.57$$

(2) Effort equan (E)

$$E = V/L \text{ or } V^2/V^*$$

$$E = \frac{223.47}{0.127} = 17.59$$

(3) Time Equan (T')

$$T' = n_1 N_2 (N' \log_2(n)) / 2 n_2 S$$

$$\boxed{S = 5-20}$$

$$T' = 18 * 14 (103.58 \log_2(27)) / 2 * 9 \times 12$$

$$= 574.59 = 9.57 \text{ minutes}.$$

(4) Pgmmg time:

$$P' = E/S \qquad P' = 1759/12$$

$$P' = 146.52.$$

OO CK metric suite

Chidambar - Kemerer

~~categories~~

WMC — weighted methods for class.

DIT → Depth of Inheritance.

NOC — No. of children.

CBO — Coupling b/w object classes.

RFC — Response for class object.

LCOM — lack of cohesion in object method.

Class Object

Properties

operations

int a, b, c, d;

xyz (a, b); — c1
pqr (c); — c2
abc (d); — c3

Cyclomatic complexity

$\therefore$ C or $V(G) = P+1$

$$WMC = \sum c_i = c_1 + c_2 + c_3$$

Two authors provided metric for object oriented projt evaluation, which include.

WMC - Wtd Method per class.

This metric determines the wait of a class by evaluating complexity of every method involved in it. The complexity of every method is evaluated using cyclomatic complexity formula,

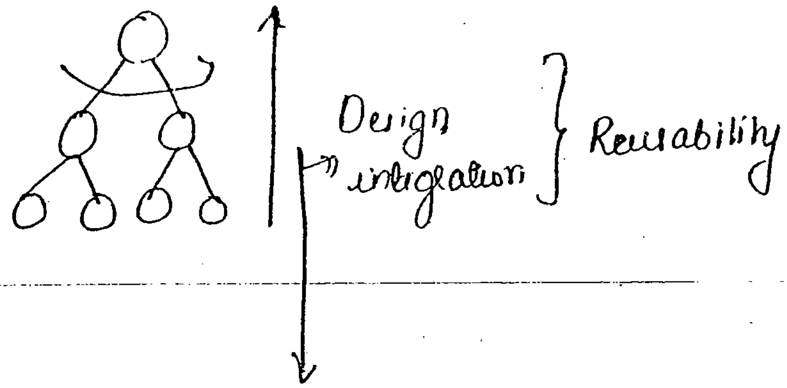
$$C \text{ or } V(G) = P + 1$$

$P \Rightarrow$ no. of predicates involved in a gvn method

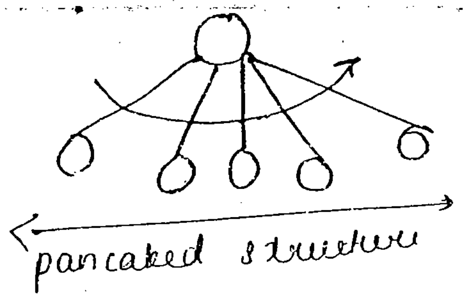
a predicate may be simple predicate or composite predicate. where composite predicate contains more than 1 condn separated by logical operators. After evaluation of complexity of every method in a class, finally wt of class is evaluated by using WMC.

DIT:

This metric evaluate the no. of levels from leaf node to root node. As per obj. oriented approach no of levels shd be restricted otherwise it leads to design complexity as well as integration complexity and also it violates the primary objective of obj. orientd approach.



This metric determines no. of immediate subordinates for a gvn node.
shd be restricted. Otherwise, it leads to integration complexity as well as testing complexity.



CBO

This metric evaluate relationship b/w int. classes and external collaborations. A project shd contain more of internal classes & less of external collaborators to achieve functional independency.

Internal classes ↑more R external collaborators ↑less

RPC:

LCOM (35)

This metric evaluate sharing of common property or resource b/w the methods in a class. As per OO approach, the sharing of common resource shd be moderate. Evaluated based on worst case scenario.

xyz (a, b)
pqr (b, c)
abc (b, d)

LCOM = 3

→ xyz (a, b);
pqr (b, c);
abc (c, d);

} LCOM = 3.

Test Activity:

S/w testing technique – Test cases.

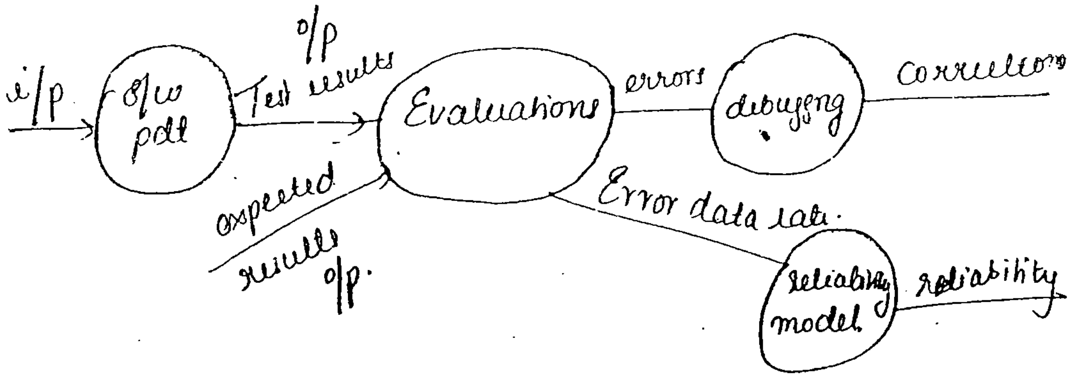
S/w testing strategies – level of testing.

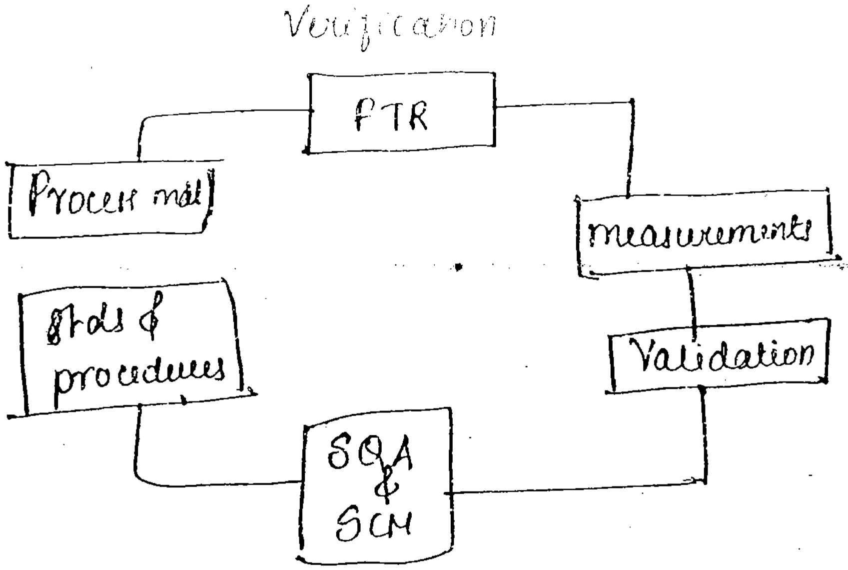
V & V

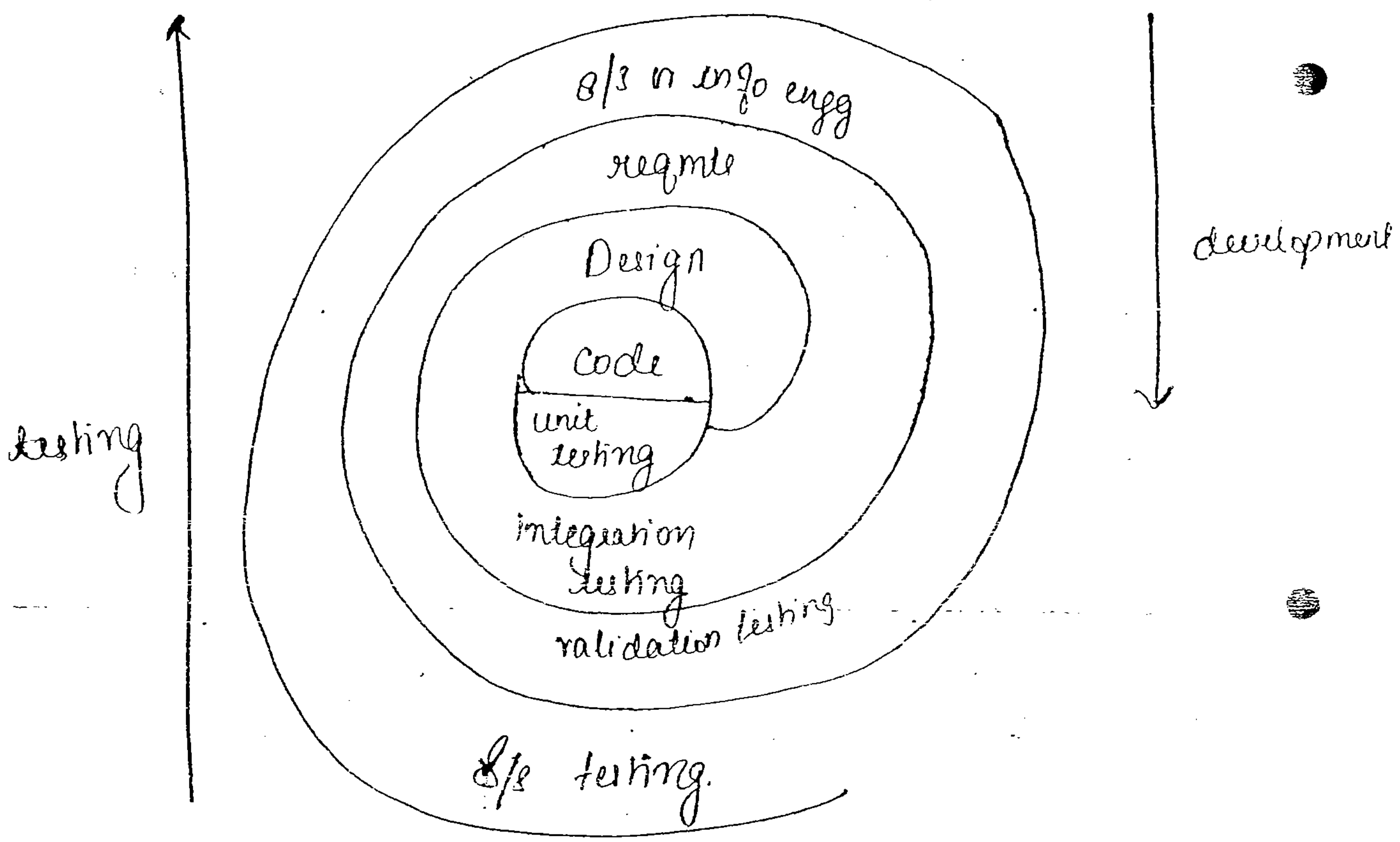
Verification – static – before code – logic flow – FTR (checklist)

Validation – dynamic – after code – logic flow, data flow

Formal Technical review committee – FTR







Testing is a process of executing a pdt to determine difference b/w test results & expected results and also to evaluate all the parameters of the pdt that include quality, reliability, efficiency so on.

Primary objective of testing is to identify bugs at the earliest because if the bug is not identified, if it migrates to other activity, not only cost increases to fix the bug, it also deteriorates quality of pdt.

- Testing is performed properly by preparing test cases which are scenarios with ability to identify bugs in the pdt.

Testing is not exhaustive ie complete. Testing is successful when maxm. proactive bugs are removed from the pdt such that defect free pdt is achieved.

Generic characteristic of testing:

Testing works outwardly by satisfying all the activities of development in a reverse direction.

S/w testing techniques are available to prepare test cases which are used at different levels of testing.

Testing is performed bttr by testers rather than developers because testing performed by developer is macroscopic level & by tester it is microscopic.

Testing and debugging are two diff process former identifies bugs and debugger removes bug.

9/10
sday.

S/w Testing Techniques

- how to prepare testcases?

G Questions

- how many testcases
- cyclomatic complexity
- Independent paths.
- Reachability measure.

White Box Testing. - logic driven - Internal to code.
Black Box Testing. — I/o driven. — External to ~~i/p~~ code.
↳ functionality driven.

White Box testing
- Basic Path Testing
  - Flow graph approach
  - Graph matrices approach
- Control Structure Testing
  - condn testing
  - Dataflow Testing
  - Loop testing

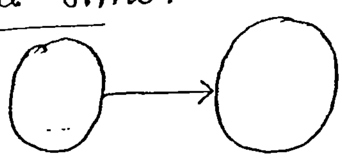
logic coverage
data coverage

Flow graph approach:

Step 1: Construct a flow graph for a Task.
Step 2: Evaluate cyclomatic complexity
Step 3: Determine independent bugs.
Step 4: Prepare test cases depending on no. of independent bugs.

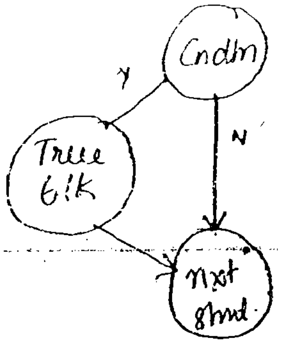
Notations:

Sequential stmt:

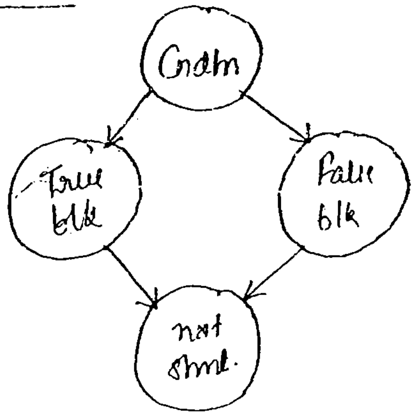


Selective stmt:

- Bidirection stmt:
  - Simple if

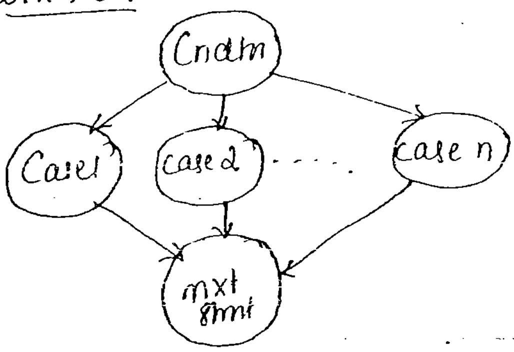


if else



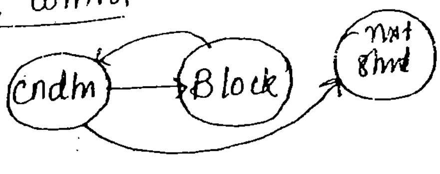
Multidimensional stmt

switch-case

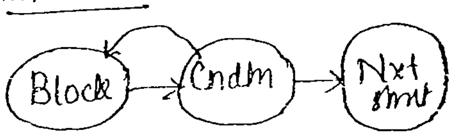


Repetitive stmts

Entry control



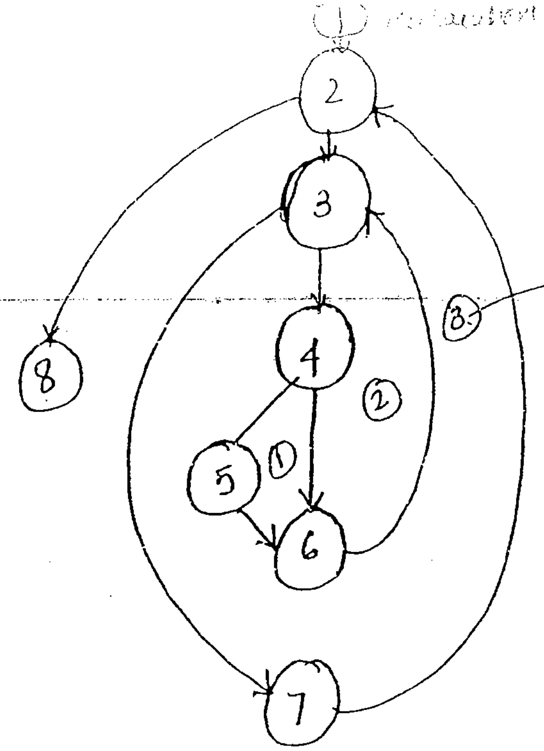
Exit contrl



Case Study 1: Swapping elts of an array. (PDL) (Pgmg Design Logic)

```
Integer : I, J, N, A[100]
While(I > N)
  do
    while(I > N)
      do  if(A[I] > A[J]) then
```

Step 1:



→ Starting

③ → 3 internal regions

Step 2: Cyclomatic complexity

logic coverage exhaustive

$$V(G) = E - N + 2$$
$$V(G) = P + 1$$
$$V(G) = \text{no. of internal regions} + 1 \text{ external region.}$$

P → predicate.

for the no. of nodes (N), no. of edges (E), we have to get the same value, otherwise we cant proceeds to next.

$V(G) = 10 - 8 + 2 = 4$

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

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

III Independent paths

Simple paths 1-2-8

1-2-3-7-2-8

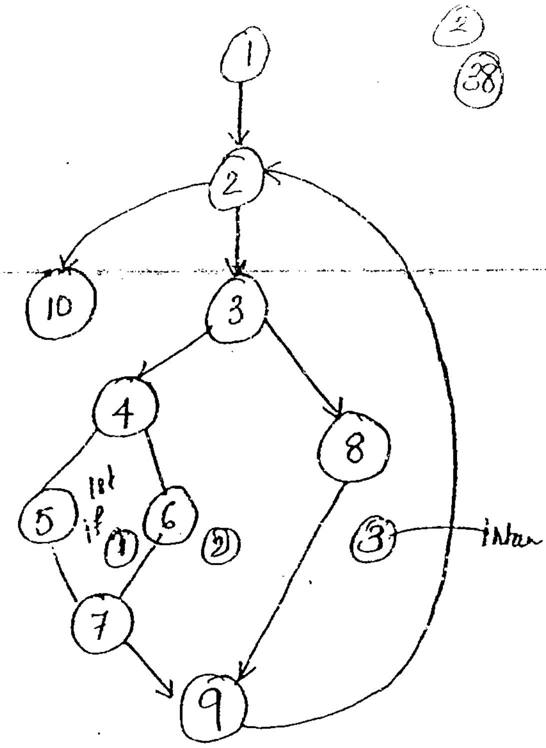
1-2-3-4-6-3-7-2-8

complex 1-2-3-4-5-6-3-7-2-8

Case Study 2 : (PDL)

I

```
Declaration
While(condn)
  do
    if (cond) then
      if (cond) then
          True Blk
      else
          False Blk
      endif
    else
      False blk
    endif
  end;
end;
```



II $V(G) = 12 - 10 + 2 = 4$

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

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

III

1 – 2 – 10

1 – 2 – 3 – 8 – 9 – 2 – 10

1 – 2 – 3 – 4 – 6 – 7 – 9 – 2 – 10

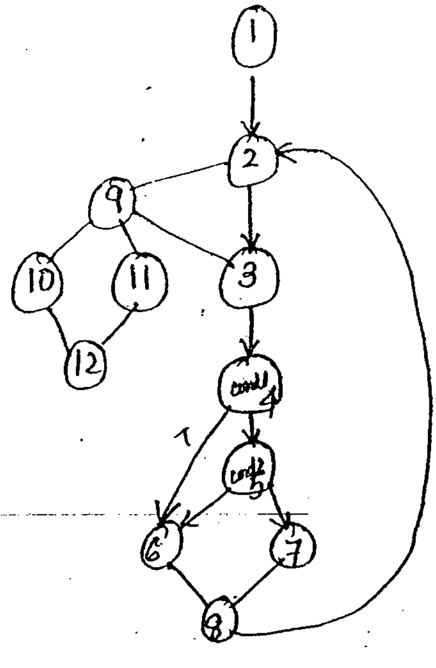
1 – 2 – 3 – 4 – 5 – 7 – 9 – 2 – 10

} 4.

IV # Test cases : 4.

Case Study 3 :

```
Declaration
while(cond1 && cond2)
 do
   if (cond1 || cond2) then
        True blk
   else
        False blk
   endif
 endif
if (cond) then
     True blk
else
     False blk.
endif
```



II $V(G) = 16 - 12 + 2 = 6$
$V(G) = 5 + 1 = 6$
$V(G) = 5 + 1 = 6$

} logic coverage is exhaustive.

III ~~1 - 2 - 9 - 10 - 12~~ ~~1 - 2 - 3 - 9 - 10 - 11 - 12~~ 1st while

1 - 2 - 9 - 11 - 12 1 - 2 - 3 - 9 - 11 - 12 2nd while

1 - 2 - 3 - 4 - 6 - 8 - 2 - 9 - 10 - 12
1 - 2 - 3 - 4 - 6 - 8 - 2 - 9 - 11 - 12
1 - 2 - 3 - 4 - 6 - 8 - 2 - 3 - 9 - 10 - 12
1 - 2 - 3 - 4 - 6 - 8 - 2 - 3 - 9 - 11 - 12

1 - 2 - 3 - 4 - 5 - 6 - 8 - 2 - 9 - 10 - 12
1 - 2 - 3 - 4 - 5 - 6 - 8 - 2 - 9 - 11 - 12
1 - 2 - 3 - 4 - 5 - 6 - 8 - 2 - 3 - 9 - 10 - 12
1 - 2 - 3 - 4 - 5 - 6 - 8 - 2 - 3 - 9 - 11 - 12

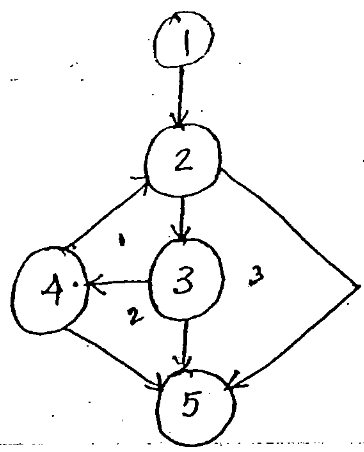
1 - 2 - 3 - 4 - 5 - 7 - 8 - 2 - 9 - 10 - 12
1 - 2 - 3 - 4 - 5 - 7 - 8 - 2 - 9 - 11 - 12

1 - 2 - 3 - 4 - 5 - 7 - 8 - 2 - 3 - 9 - 10 - 12
1 - 2 - 3 - 4 - 5 - 7 - 8 - 2 - 3 - 9 - 11 - 12

Test cases = 16

[test cases dependent on independent paths only.]

Graph Matrices Approach:



II $V(G) = 7 - 5 + 2 = 4$
$V(G) = 3 + 1 = 4$

we can't get the no. of predicates.

To evaluate predicate, draw 5 x 5 matrix

	1	2	3	4	5	
1		1				→ 1 − 1 = 0
2			1		1	→ 2 − 1 = 1 +
3				1	1	→ 2 − 1 = 1 +
4		1			1	→ 2 − 1 = 1
5						add / subtract 3

no. of predicates = 3 (B)(31)

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

$\therefore$ logic coverage is exhaustive.

<u>Independent paths</u>

1 - 2 - 5

1 - 2 - 3 - 5

1 - 2 - 3 - 4 - 5

1 - 2 - 3 - 4 - 2 - 5

1 - 2 - 3 - 4 - 2 - 3 - 5

Test cases = 5

iQ How to calculate reachability measure?

$$\boxed{\text{Reachability measure} = \frac{\text{Total no. of paths}}{\text{no. of nodes}}}$$

Node 1: as source node 1 as destination ; self path = 0

Node 2: $\left.\begin{matrix} 1-2 \\ 1-2-3-4-2 \end{matrix}\right\}$ 2

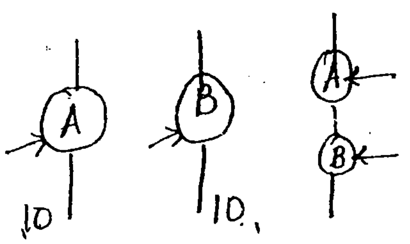
Node 3: $\left.\begin{matrix} 1-2-3 \\ 1-2-3-4-2-3 \end{matrix}\right\}$ 2

Node 4: 1 - 2 - 3 - 4 — 1

6+5 = 11

Reachability measure $= \frac{11}{5} = 2.2$

→ If Cyclomatic complexity > 16, then we subdivide the module so that C.C < 16.



First add A & B comp. 10+10>16 so subdivide here 10, 10 so max = 10.

here, CC. never exceeds 16

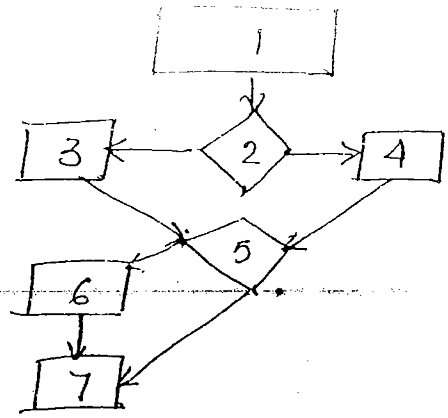
If A=13 B=10 ⇒ max 13.

~~10~~ 19 20 21

Overall complexity depends on max complexity of s/s.

Q. 47, 41 $(C_1 + C_2)$

Pg 19 Flowchart



$V(G) = 8-7+2 = 3$

$V(G) = 2+1 = 3$

$V(G) = 2+1 = 3$

Independent paths:

Node 7: 1-2-3-5-7
1-2-3-5-6-7
1-2-4-5-7
1-2-4-5-6-7

Test cases = 4.

Node 1: Self — 1

Node 2: 1-2 — 1

Node 3: 1-2-3 — 1

Node 4: 1-2-4 — 1

Node 5: 1-2-3-5
1-2-4-5 — 2

Node 6: 1-2-3-5-6
1-2-4-5-6 — 2

Reachab. measure $= \frac{8+4}{7}$
$= 1.7$

Control Structure Testing

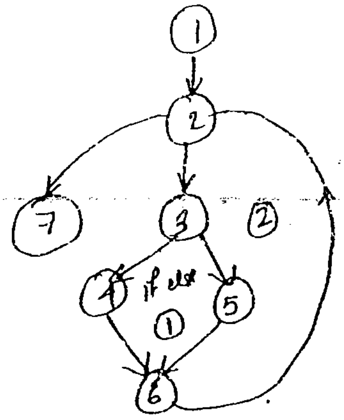
Condition testing:

Condition
- Simple → while(cond), if (cond) } less testcase
- Composite → if (cond||cond), while (cond1 && cond2) } more testcase.

Case Study 1: Simple condtn

Declaration

```
while (cond)
do
    if (cond) then
        True blk
    else
        False blk
    endif
enddo
end.
```



$V(G) = 8-7+2 = 3$
$V(G) = 2+1 = 3$
$V(G) = 2+1 = 3$

logic coverage exhaustive

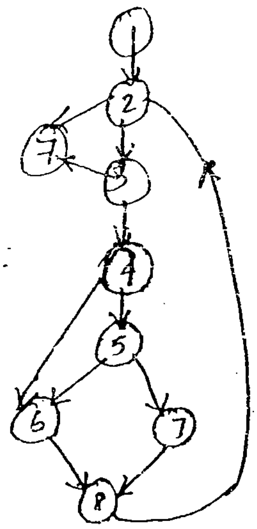
1-2-7
1-2-3-5-6-2-7
1-2-3-4-6-2-7

3 test cases.

Case study 2: Composite

Declaration

```
while (cond && cond)
do
    if (cond || cond) then
        True blk
    else
        False blk
    endif
enddo
end;
```



$V(G) = 12-9+2 = 5$
$V(G) = 4+1 = 5$
$V(G) = 4+1 = 5$

exhaustive.

1-2-9
1-2-3-9
1-2-3 4 6 8 2 9
1 2 3 4 6 8 2 3 9

1 2 3 4 5 6 8 2 3 9

1 2 3 4 5 6 8 2 9

1 2 3 4 5 7 8 2 9

1 2 3 4 5 ~~8~~ 7 8 2 3 9

<u>8 Test cases</u>.

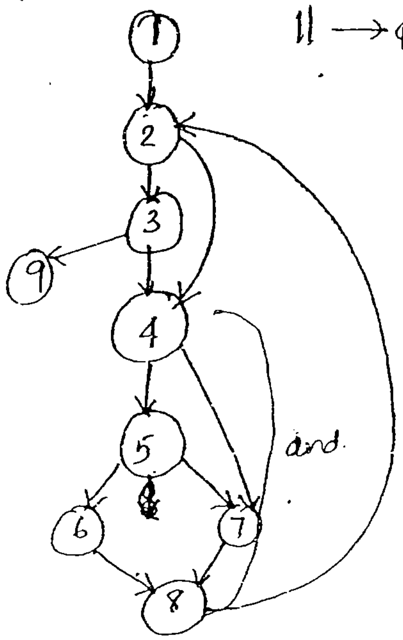
Case Study 8:

interchange the conditions.

&& → ||
|| → &&.

Declaration

```
While (cond || cond)
do
   If (cond && cond) then
        True blk.
   else
        False blk.
   endif
enddo
end;
```



$V(G) = 12 - 9 + 2 = 5$

$V(G) = 4 + 1 = 5$

$V(G) = 4 + 1 = 5$

indpn. paths:

1 – 2 – 3 – 9
1 – 2 – 4 – 7 – 8 – 2 – 3 – 9
1 – 2 – 3 – 4 – 7 – 8 – 2 – 3 – 9
1 – 2 – 4 – 5 – 7 – 8 – 2 – 3 – 9
1 – 2 – 3 – 4 – 5 – 7 – 8 – 2 – 3 – 9
1 – 2 – 4 – 5 – 6 – 8 – 2 – 3 – 9
1 – 2 – 3 – 4 – 5 – 6 – 8 – 2 – 3 – 9

<u>7 Test cases</u>

10 / rsday

Data Flow Testing - Data coverage Test cases

SE (1)

Datatype a, b, c, d ;

if all vars are used effectively, then the pgm is efficient.

efficient pgm

Performance reqm. Space reqm

wastage of m/m & efficiency ↓

case 1 : a, b, c.
case 2 : a, b, c, d.

case 3 : a, b, c, d, e

e is not declared, cpu displays that e is undeclared.

No. of test cases for a declaration is 2^{bit} representation.

case 1 : $2^{32} \cdot 2^{32} \cdot 2^{32} \longrightarrow 2^{96}$

case 2 : $2^{32} \cdot 2^{32} \cdot 2^{32} \cdot 2^{32}$ — 2^{128}

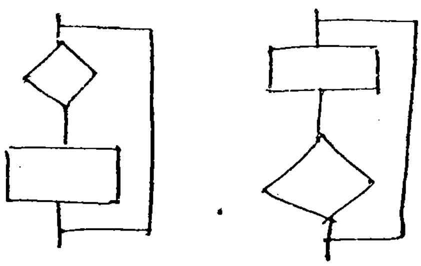
case 3 : $2^{32} \cdot 2^{32} \cdot 2^{32} \cdot 2^{32} \cdot 2^{32}$ — 2^{160} more than the declared vars.

Minimum 1 testcase for each declared resource. Here we need atleast 4 testcases.

Loop Testing : - logic coverage test cases.

→ Simple loop
→ Nested loop
→ Concatenated loop.
→ Unstructured loop. (generally we neglect unstructured loop)

Simple loop



```
for (i=0; i<=10 ; i++)
    pf ("%d", i);
```

```
for (i=0 ; i<=10 ; i++);
    pf . . .
```

```
for (i=0 ; i++)
    pf :
```

we prepare 1 testcase to evaluate all the condtns of the loop.

```
for (i=0 ; i<3 ; i++)
  for (j=0 ; j<3 ; j++)
      pf ("%d", A[I][j]);
```

a_{00}	a_{01}	a_{02}	a_{03} ②
a_{10}	a_{11}	a_{12}	(kigd) a_{13}
✓ a_{20}	a_{21}	a_{22}	a_{23}
① a_{30}	a_{31}	a_{32}	a_{33} ③

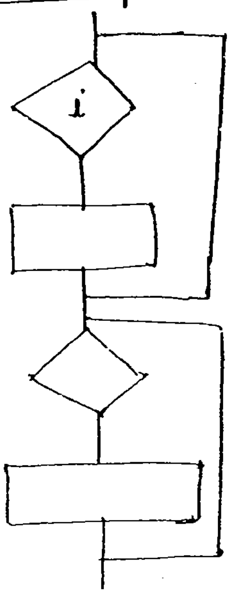
① ';' for i loop

② ';' for j loop

③ ';' for both

We prepare test cases for each.

Concatenated loop



```
for (i=0 ; i<=10 ; i++)
     pf ("%d", i);

for (i=11 ; i<=100 ; i++);
     pf ("%d", i);
```

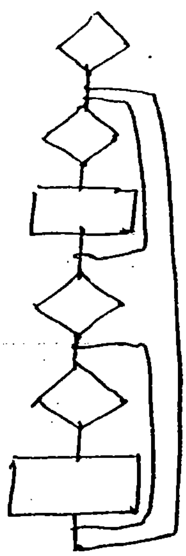
expected o/p	case 1:	case 2	case 3	case 4
0	11	0	11	0
:	11	1	101	1
10	:	2		2
11	76	10		:
:	:	101		10
100	100			11
				:
				Stack overflow

case 5:

Stack overflow (in first loop itself).

Unstructured loop

eg: goto. stmt



Black Box Testing - I/o driven testcases. SE (2)

* Equivalence Partitioning Testing.
* Boundary value Analysis (BVA)
* Logic Coverage (Criteria)
* Random Generation.
* Error Guessing.
* Comparison Testing.

Case Study: factorial number

```
int i ; fact=1;
for(i=1 ; i<=n ; i++)
    fact = fact * i;
```

± 32767

D2				D1							D3		
-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
invalid				valid							invalid		

(1)

(2) -3 -2 -1 $\boxed{0 \mid 1}$ 2 3 4 5 6 $\boxed{7 \mid 8}$ 9 10

boundary values must be checked.

(3) Logic coverage ─ Stmts / Branch / Path

every pgms must satisfy the 3 criteria. (they shd execute atleast once)

In flow graph ─ edges + node / predicate / regions

(4) Random generation: does not give bugs.
eg:- without grg boundary values, sm random i/ps values are used. Hence bugs mit not surface.

(5) Error Guessing - Alerts.
eg:- mandatory columns in online form left blank, then alert comes up.
When there is no enuf money to withdraw, ATM gives alert.
Alerts are given to the customers

Real time Sys
- No predicted o/p
- o/p

Application
- Predicted o/p
- predicted o/p

RTS
- hard RTS — strict RTS
- Soft RTS — small delay is tolerated

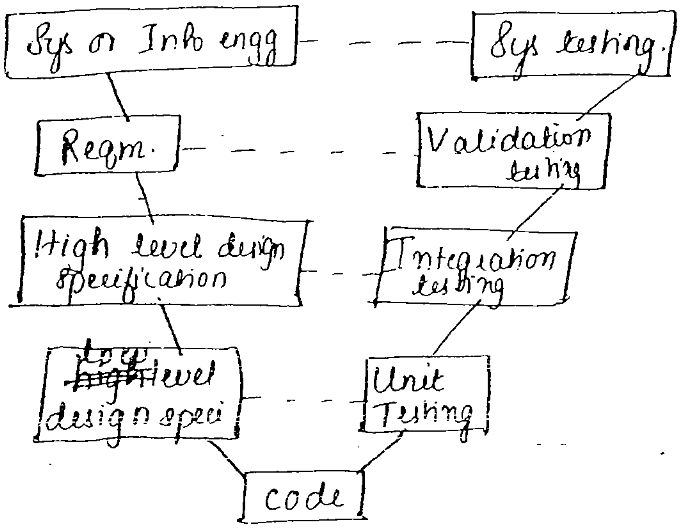
n teams

Testcases: $\left(\frac{1\ \text{version}}{n} \quad \frac{2\ \text{version}}{n} \quad \frac{n\ \text{version}}{n} \right)$

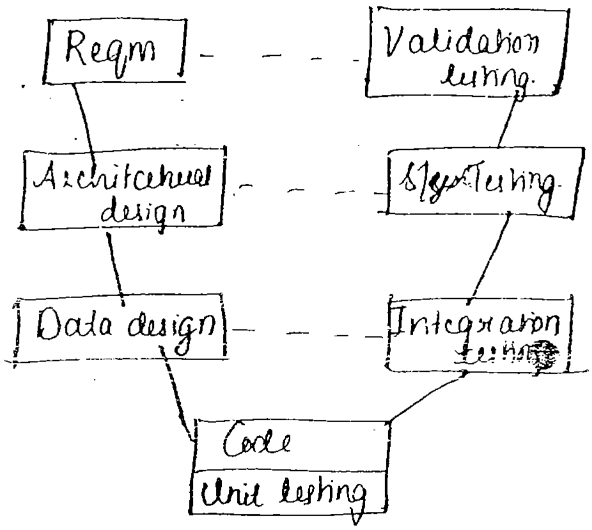
Testcases shd provide same o/p for all i/ps.

S/w Test Strategies -- Level of testing

V model

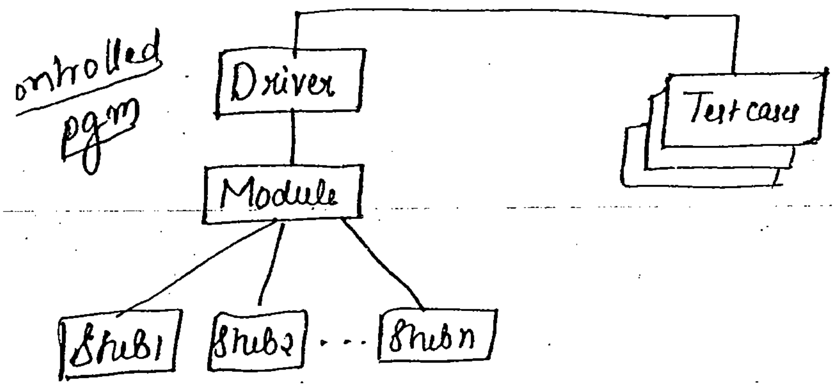


90 % companies



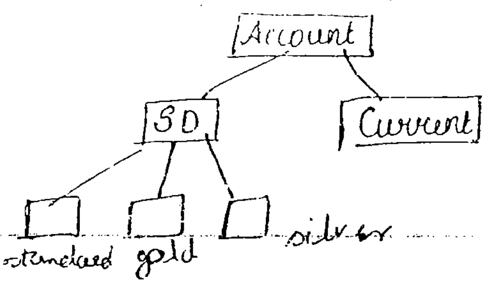
10 % companies

Unit Testing or Module Testing - Low level Design Specification.



- Interface Testing. (BB)
- Local Data Structures (WB)
- Independent paths (WB)
- Boundary condns. (BB)
- Error handling. (BB)

Stub contains atomic operation



Driver is the main pgm. Module is conntd to driver. Each stub contain atomic operation & each one independently executed.

If you find, some stubs are workg and some are not working so those should reevaluate; once the module performing upto mark, then apply Interface testing here inflow, outflow → Cohesion & coupling

Simple or complex interface

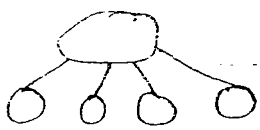
Local data structure → Declare (n)
Usage (n)

Independent path → going to exercise complete logic of path aspects.

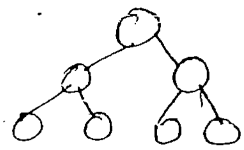
Integration Testing - High level Specification Design → External aspects → Architectural design

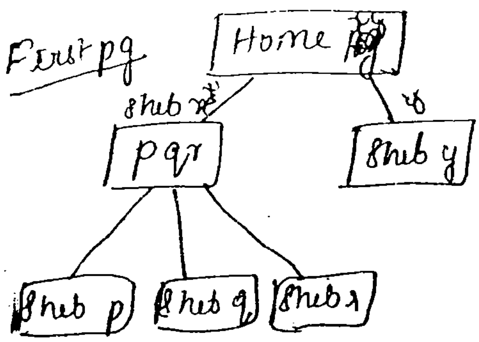
Non Incremental Integration - Big Bang. —(generally nt used)

Incremental Integration — Top down — web technologies (stub used)
— Bottom up — Traditional / classical (drivers used)


non incremental

~~Incremental~~ ~~Top down~~

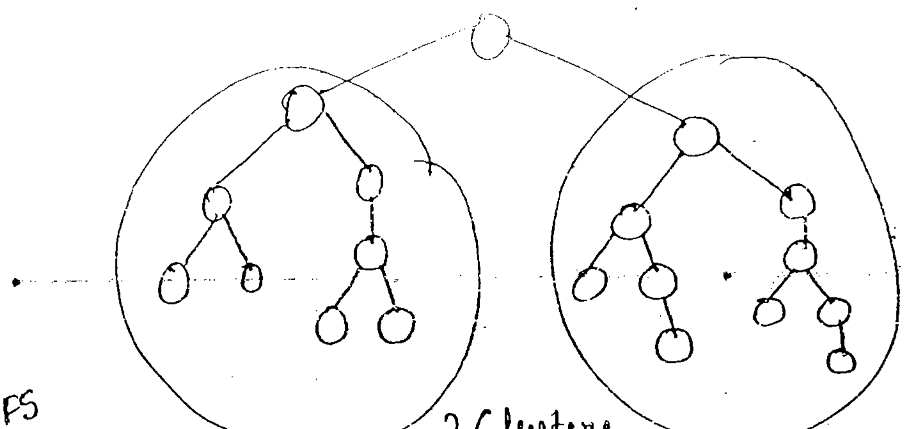

(Traditional / classical)



BFS (dummy page by OS as stub x & stub y)

pqr (immediate succ of webpage)

Everytime, testing, the old pg is also testing with new one → Regression testing



FS

iff level

2 Clusters

Modules of a proj

clusters are united by driver/dummy pgm.

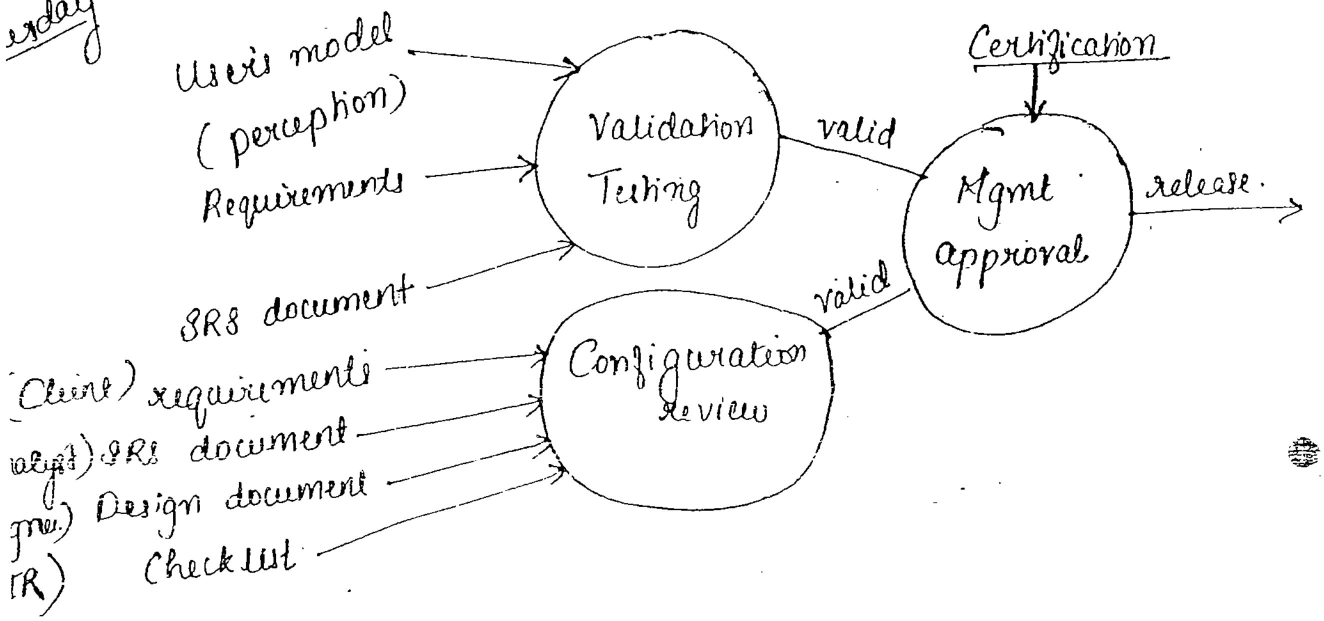
Module testing: driver & stubs.

Validation Testing - requirements
- developer
- customer

} func. requirements needs.

7/10

Friday



* Developer — α testing - simulation environment — virtuality testing.

* Client — β testing - live environment - reality testing.

User's model is done in live environment.

Here, we check the functional requirements.

System Testing - System or Info engg - non func. requirements

~~Generic testing~~
- Recovery Testing
- Security Testing.
- Stress Testing.

Recovery testing - failure and its impact.
They provide simple trouble shooting
Security Testing - (Safety - integrity)

$$\text{Integrity} = 1 - \underset{\uparrow \text{more}}{\underline{\text{security}}} \times 1 - \underset{\uparrow \text{less}}{\underline{\text{threat}}}$$

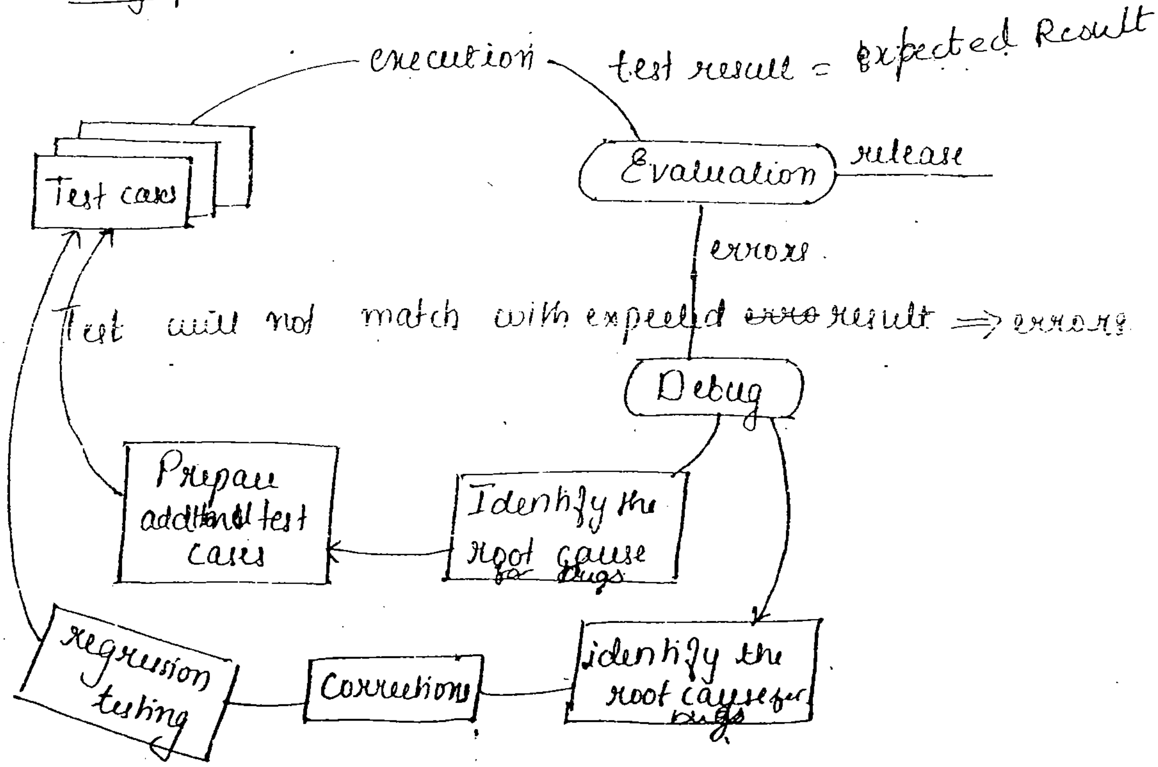
also gives privacy (like pwd & uname)

Stress testing: extend to which the system can work comfortably in terms of load (Threshold point)

Performance testing: efficiency.
- Space complexity
- Time complexity.

Bugs and Debugging:

Debug process.



Coding: debug

1 : 10

Test cases to remove the bugs will be 10 times that of coding.

Techniques:

→ ~~Brute force technique~~

90% → cause elimination — induction / deduction

10% → Back tracking. - never provides list of bugs. Only one bug at a time.

Maintenance:

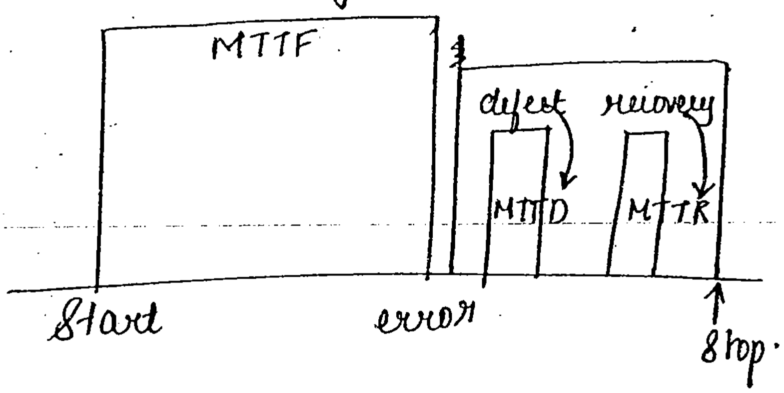
Person → Development, Maintenance

→ SMI

→ MTTF availability. (Mean time to failure)

Maintenance (1:3) 60%

* Bug removal or corrective maintenance. (changes internal aspects) — 21%
* Adaption (or) adaptive maintenance. (external) — 25%
* Enhancement (or) Perfective maintenance. (internal) — 50%
* Re-engineering (or) Primitive maintenance. — 4%



MTBF = MTTF + MTTR

(meantime b/w failure)

$$\boxed{\text{MTTF availability} = \frac{MTTF}{MTTF + MTTR}}$$ by default

$$\text{MTBF availability} = \frac{MTBF}{MTBF + MTTR}$$

Q: Ans XYZ s/w company released a pdt to client, it works for 2 yrs, then the first failure occured. The maintenance team has taken 3 days to resolve the problem. Calculate meantime b/w failure & availability & non availability of the pdt.

convert into hours $2 \times 365 \times 24$ — 17520 hrs.

no of hrs to correct it: 3×24 — 72 hrs.

$$MTBF = 17520 + 72$$
$$= 17592$$

$$\text{availability} = \frac{17520}{17592} = 0.995 \qquad \frac{MTTF}{MTTF + MTTR}$$

$$\text{non availability} = 1 - 0.995$$
$$= \underline{0.005}$$

(shd never be >1% !!)

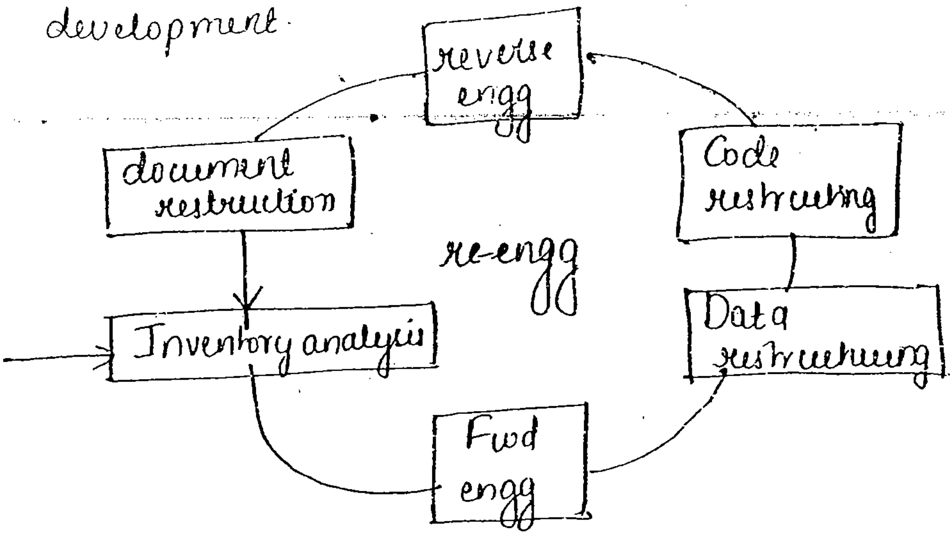
$$SMI = [M_T - [f_a + f_d + f_c]] / M_T$$

a - added
d - deleted
c - changed
T - total
M - module

$$SMI = [30 - (3 + 2 + 4)] / 30$$
$$= 0.7$$

Reengineering:

Cost of redevelopment is very high compared to development.



old Technology ←

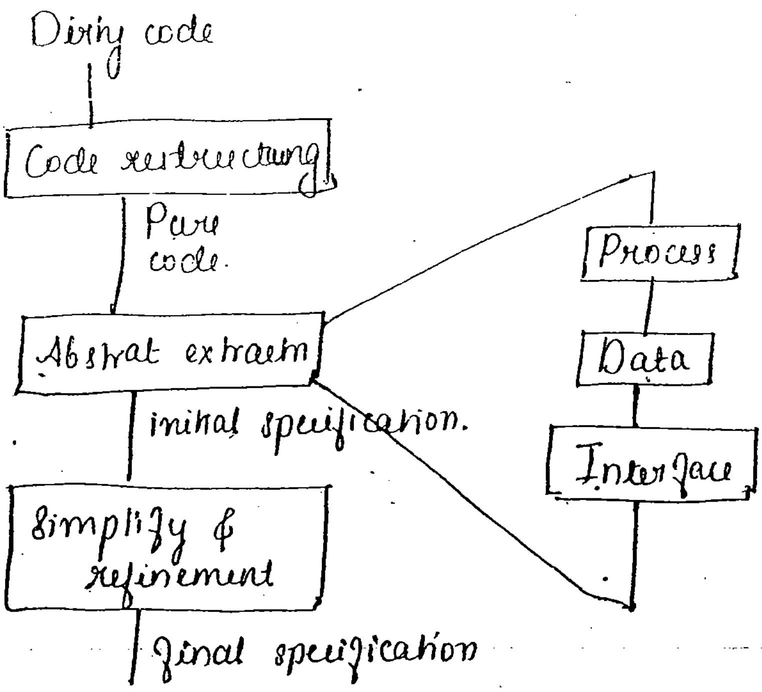
Algol (dirty code)

Specification ← Analysis ← Design ← Code

refine & simplify

final specification — Analysis — Design — Code — test — release

↑ objective.



Maintenance: (COCOMO maintenance)

development team > maintenance team

since maintenance is not that often.

ACT = Annual cost Tariff

$$ACT = \frac{\text{Add + modify}}{\text{Total size}} = \frac{2+3}{30} = 0.16.$$

$$M_E = 1.0\,(ACT)\,D_E \quad \text{Person-month.}$$

maintenance effort

$$D_E \text{ or } E = a_b\,Kloc)^{bb}$$

$$M_D = 1.0\,(ACT)\,D_D$$

maintenance duration

$$D_D = C_b\,(E)^{db}$$

$$M_N = 1.0\,(ACT)\,(M_E/M_D) \text{ persons.}$$

no. of persons for maintenance.

Organic mode

$$M_E = 1.0\,(0.16)\,(85.3) = 13.6 \text{ person month}$$

$$D_E = 2.4\,(30)^{1.05} = 85.3.$$

$$M_D = 1.0\,(0.16)\,(\cancel{12.45}\ 13.5) = \cancel{1.9}\ 2.16 \text{ month.}$$

$$D_D = 2.3\,(85.3)^{0.38} = \cancel{12.45}\ 13.5 \text{ month}$$

$$M_N = 1.0\,(0.16)\left(\frac{13.6}{2.16}\right) = 1.\cancel{14}\,07 \simeq \boxed{1 \text{ person}}$$

$$D_N = \frac{85.3}{12.45} = 6.8 \simeq 6 \text{ persons}$$

Semidetached mode

$$M_E = 1.0\,(0.16)\,(135.36) = 21.65$$

$$D_E = 3.0\,(30)^{1.12} = 135.36$$

$$M_D = 1.0\,(0.16)\,(13.9) = 2.2$$

$$D_D = 2.5\,(135.36)^{0.35} = 13.9 \text{ months}$$

$$M_N = 1.0\,(0.16)\left(\frac{21.65}{2.2}\right) = 1.5 \simeq 2 \text{ persons}$$

$$D_N = \left(\frac{135.36}{13.9}\right) = 9.79 \simeq 10 \text{ persons}$$

Embedded mode

$$M_E = 1.0\,(0.16)\,(213.2) = 34.11 \text{ pm}$$

$$D_E = 3.6 \times (30)^{1.2} = 213.2 \text{ pm}$$

$$M_D = 1.0\,(0.16)\,(13.9) = 2.2$$

$$D_D = 2.5\,(213.2)^{0.32} = 13.9 \text{ month}$$

$$M_N = 1.0\,(0.16)\,(34.11/2.2) = 2.48 \simeq 2 \text{ person}$$

$$D_N = \left(\frac{213.2}{13.9}\right) = 15.3 \simeq 15$$

	OM	SM	EM
Developmt	6	10	15
maintenane	1	2	2

4. $ACT = \frac{2.13}{15} = 0.33$

$M_E = 1.0(0.33)(62.27) = \underline{\underline{20.5 \text{ pm}}}$

$D_E = 3.0(15)^{1.12} = 62.27$

continued

5. $M_D = 1.0(0.33)(10.6) = \underline{\underline{3.5}}$ months

$D_D = 2.5(62.27)^{0.35} = 10.6\ M$

$M_N = 1.0(0.33)\left(\frac{20.5}{3.5}\right) = 1.93 \approx 2$ persons.

6. $ACT = 0.033.$ (1/100)

$M_E = 1.0(0.033)(821) = 27.09$ PM

$D_E = 3.0(150)^{1.12} = 821$ pm

continued from 6.

7. $M_D = (1.0)(0.033)(26.15) = 0.863$ mnths.

$D_D = (2.5)(821)^{0.35} = 26.15$ mnths

$M_N = \frac{27.09}{0.86} \times 0.033 \times 1.0 = 1.07 \approx \boxed{1 \text{ person}}$

Meenu Mathew
PM7