SOFTWARE PROJECT PLANNING

- After finalisation of SRS, the another step to estimate size, Cost & development time of Project.

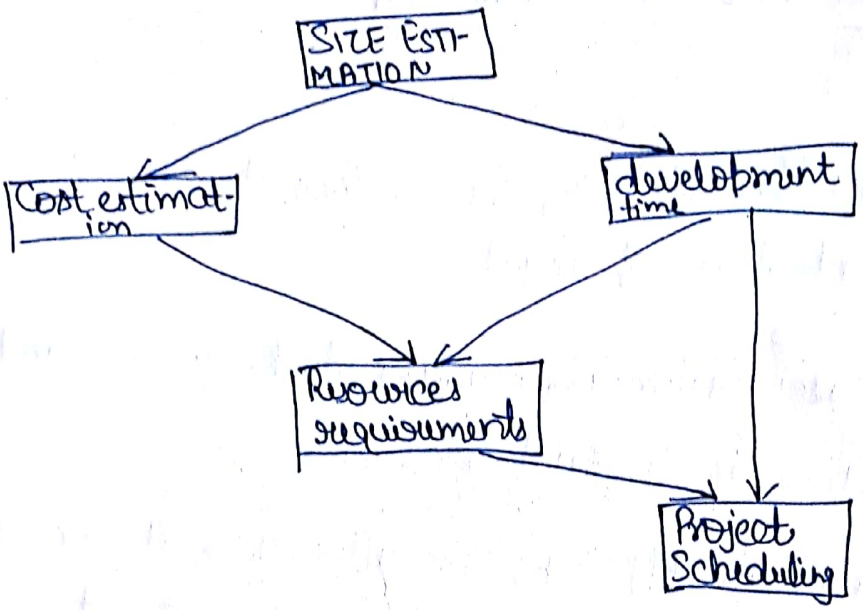
- In many cases customer may like to know cost and development time even prior to finalisation of SRS.

- Whether we estimate b/f SRS or after SRS, it would be always be very criticial and crucial decision for Project.

Project Planning must incorporate major issues like: size, Cost estimation, scheduling, Project monitoring and reviews, Personnel selection and evaluation & risk management.

In Order to Conduct a succesful S/w Project, we must understand:

i) Scope of work to be done
ii) risk to be incurred
iii) resources required
iv) task to be accomplished
v) Cost to be expended
vi) schedule to be followed



ACTIVITIES DURING S/W PROJECT PLANNING

- Firstly, activity is to estimate size of Project.
- Size is the key Parameter for estimation of other activities
- Size is input to all costing models for
  (i) estimation of cost
  (ii) development time
  (iii) schedule for Project

- Resources req. are estimated on basis of cost and development time.

- Project scheduling is useful for controlling and monitoring Progress of Project.
- This (P.S) is dependent on resource & dev. time

SIZE ESTIMATION :- Some Pgs. are written in C, Some in Pascal, others in Fortran, others in assembly lang.

- Some Pgs. are well documented, other Pgs. are written in a quick & dirty fashion with no comments at all.

There is one char. that all Pgs. share - they all have size.

Size can be estimated by :-

i) LOC (Line of Count)

ii) Function Count

LOC

- Simplest measure of Problem size is line of code.
- Simple to use
- This metric measures no. of source instructions required to solve a Problem.
- While counting the source instructions, lines used for commenting the code and header lines are ignored.
- include declarations, statements.
- Determining LOC count at end of a Project is a very simple job.

- To estimate LOC at beginning of a Project is simple.
- To count at beginning of Project, Project manager usually divide Problem into modules, and each module into submodule and so on, until size of different leaf-level modules can be Predicted.
- By using estimation of lowest level modules, Project managers arrive at total size estimation.

LOC as a measure of Problem size has several shortcomings:

1.) LOC gives a numerical value of a Problem size that can vary widely with individual coding style - different Programmers lay out their code in different way.
- e.g: one Programmer might write several source instructions on a single line whereas another might split a single instruction across several line.

2.) LOC measure correlates Poorly with quality and efficiency of code. A larger size code doesn't necessarily imply better quality or higher efficiency.

3.) A good Problem size measure should consider overall complexity of Problem and effort needed to solve it i.e should consider effort needed to specify

design, code, test etc. But LOC focuses on coding activity alone.

- very difficult to accurately estimate LOC in final Product from Problem specification. LOC count can be accurately computed only after code has been fully developed.
∴ LOC is of little use to Project managers during Project Planning, since Project Planning is carried out much bft development activity is started.

```
1   int sort (int x[], int n)
2   {
3       int i, j, save, im1;
4   /* fn. sorts array x in ascending order */
5   If (n<2) return 1;
6.  for (i=2 ; i<=n ; i++)
7.  {
8.  im1 = i-1;
9.  for (j=1 ; j<=im ; j++)
10. if (x[i]<x[j])
11. {
12. save = x[i];
13. x[i] = x[j];
14. x[j] = save;
15. }
16. }
17 return 0;
18. }
```

If count as simple no. of lines then 18 LOC

But most researches says LOC should not include comments or blank lines.

So 17 LOC.

If main interest for specific functionality,

So only executable statements will includes

exe. st = 5-17 (17-5)

So LOC = 13

So " LOC is any line of Pg. text that is not a comment or blank line, regardless of no. of sts. or fragments of sts. on line.

This includes:

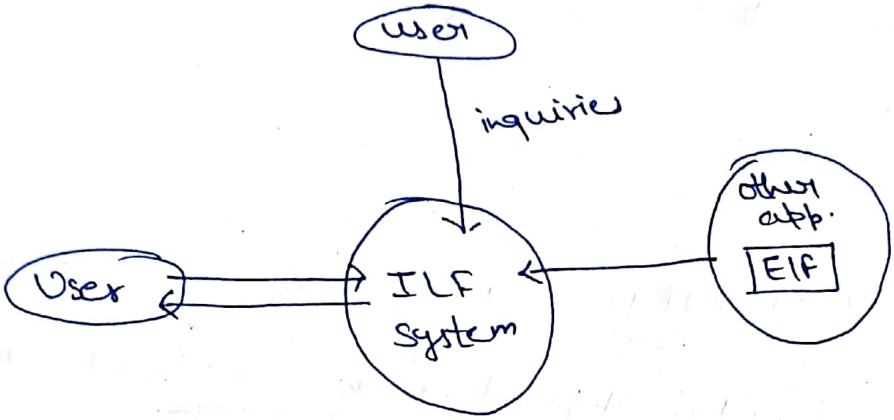
* all line containing:
  i) header
  ii) declaration
  iii) executable
  iv) non-exe- sts.

So 17 LOC.

Adv:
1.) Simple to measure

Disadv:
1.) Language dependent
2.) Bad S/W design may cause excessive line

FPA functional units:-



ILF : Internal logical files
ELF : External " "

5 fun. units are divided in two categories:

(i) DATA FUNCTION TYPES : (i) ILF
(ii) ELF

ILF = data + control int. maintained within system

ELF = data + control int. referenced by system but maintain within another sys.

(ii.) ~~Feat~~ Transactional Function types : (i) EI : external input
(ii.) EO " output
(iii) EQ " query

FEATURES :- (1.) FPA Independent of language, tools.

2.) FP can be estimated from req. specification or design specification

3.) User can't understand it easily.

4.) It characterise only one specific view of size, namely length, takes no account for functionality or complexity

FUNCTION COUNT :- Basic idea behind this size of s/w Product is directly dependent on no. and type of different fns. it Performs.

- Function Point measures functionality from user Pt. of view i.e on basis of what user requests and receive in return from system.

Principal of FPA (Function Point Analysis) : system is decomposed into functional units:

1.) INPUTS : int. entering system (data)

2.) OUTPUTS : int leaving system (reports, screens)

3.) Enquiries :- no. of enquiries (request for instant (queries) access to int.)

4.) Internal logical files : int. held within system

5.) External interface files : int. held by other system that is used by system being analyzed.

data structure logical files

Data files on tape, disk etc.

COUNTING FP (Function Points) :

Five Fun units ranked to their complexity i.e low, average or high

Functional units	Weighting Factor		
	low	average	High
EI	3	4	6
EO	4	5	7
EQ	3	4	6
ILF	7	10	15
ELF	5	7	10

all are predefined.

	Count Comp.	Complexity total
EI =	low × 3 =	—
	Avg × 4 =	—
	High × 6 =	—

Functional Unit total.

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

↓ unadjusted Functional Points

i = row
j = column

w_{ij} = entry of ith row jth col.

z_{ij} = count of no. of Functional units of Type i that have been classified as having complexity corresponding to column j.

$FP = UFP * CAF$

↓

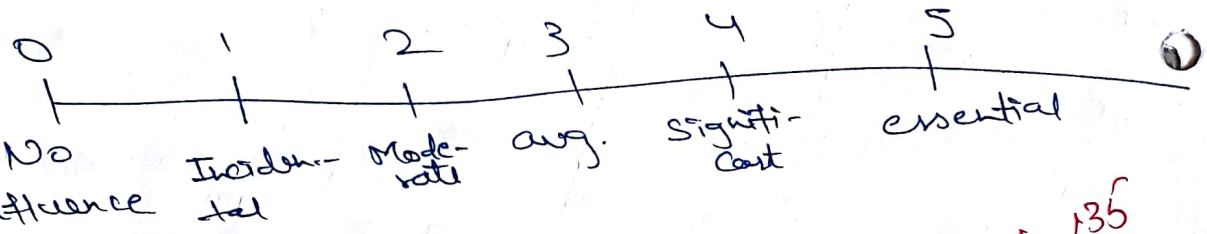
Complexity adjustment Factor and equal to

$[0.65 + 0.01 \times \Sigma F_i]$

$F_i (i = 1 \text{ to } 14)$ are degree of influence.

No I am a Software Engg.

Rate each factor on a scale 0 to 5



Page 135 Questions 1 to 14.

Project with functional units.

No. of user input = 50

" output = 40

" enquiries = 35

" files = 06

" external interfaces = 04

assume CAF and weighting factor are average.

FP = ?

$$UFP = \sum_{i=1}^{5} \sum_{j=1}^{3} Z_{ij} W_{ij}$$

$= 50 \times 4 + 40 \times 5 + 35 \times 4 + 6 \times 10 + 4 \times 7$

$= 200 + 200 + 140 + 60 + 28$

$= 628$

$CAF = (0.65 + 0.01 \times \Sigma F_i)$

$= 0.65 + 0.01 (14 \times 3)$ avg.

$= 0.65 + 0.42$

$= 1.07$

$42 \times \frac{61}{100}$

$FP = UFP \times CAF$

$= 628 \times 1.07 = 672.$

#
10 low EI
12 high EO
20 low ILF
15 high EIF
12 avg. EQ

CAF = 1.10

UFP = $10 \times 3 + 12 \times 7 + 20 \times 7 + 15 \times 10 + 12 \times 4$

$= 30 + 84 + 140 + 150 + 48$

$= 452$

$FP = UFP \times CAF$

$= ? \times 1.10$

$= 452 \times 1.10$

$= 497.2$

$0.01 (14 \times 2)$
$0.01 (28)$
0.28
0.65
0.93

Eg - 4.3 do yourself.

COST ESTIMATION :-

- for any new S/w Project, it is necessary to know how much will it Cost to develop and how much development time it will take.

- No. of estimation techniques have been developed and are having foll. attributes in common:

  * Project scope must be established in advance
  * Project is broken into small pieces which are estimated individually
  * S/w metrices are used as a basis from which estimates are made.

To achieve reliable Cost and schedule estimates, a no. of option arises:

* delay estimate until late in Project (we can achieve 100% accurate estimates after Project is Complete)
* Use simple decomposition techniques to generate Project Cost and schedule estimate.
* develop empirical models for estimation
* acquire one or more automated estimation tools.

Models :- Model is concerned with representation of process to be estimated.

- A Model may be static or dynamic
- In static model, a unique variable (say size) is taken as a key element for calculating all other (say cost, time).
- In dynamic model, all variables are interdependent and there is no basic variable as in static model.

Single-Variable models :- When a model makes use of a single basic variable to calculate all others it is said to be single-variable model.

Multivariable models :- In this several variables are needed to describe S/W development process, and selected equations combine these variables to give estimate of time and cost.

Predictors :- variable, single or multiple, that are input to model to predict behaviour of a S/W development are called "predictors".

* STATIC, Single Variable Models: Methods using this model use an equation to estimate desired values such as cost, time and effort.

They all depend on same variable used as Predictor (say size).

$$C = aL^b \quad \text{---} \quad (1)$$

C = Cost

L = Size (Line of Code)

a & b = Constants

eg:

SEL model:- developed by University of Maryland for estimating its own s/w Products.

$$E = 1.4L^{0.93} \quad \text{---} \quad (2)$$

$$DOC = 30.4L^{0.90} \quad \text{---} \quad (3)$$

$$D = 4.6L^{0.26} \quad \text{---} \quad (4)$$

Effort (E in Person-months)

documentation (DOC, in no. of Pages)

duration (D, in months)

Line of Code (L, in thousands of lines) used as a Predictor.

Static, Multi-Variable model:

Walson and Felix model developed at IBM provides a relationship blw delivered lines of source code (L thousand of lines) and effort E (E in person-month)

$$E = 5.2 L^{0.91} \quad \text{---} \quad (5)$$

$$D = 4.1 L^{0.36} \quad \text{---} \quad (6)$$

↓

duration of development (D in months)

$$I = \sum_{i=1}^{29} W_i X_i$$

Productivity index uses 29 variable

W_i = factor weight for i^{th} variable

$X_i = \{-1, 0, +1\}$

estimator gives X_i one of the values $-1, 0, +1$ depending on whether variable decreases, has no effect or increase productivity respectively.

Compare Walson-Felix model with SEL model on a S/w development expected to involve 8 Person-years of effort:

(a) Calculate no. of lines of sources code that can be Produced.

(b) Calculate duration of development

(c) Calculate Productivity in LOC/PY

d.) Calculate average manning.

Solution :- amount of manpower involved = $8PY = 8 \times 12$
$= 96$ P-M
↓
Person-Month

(a.) $E = 1.4 L^{0.93}$ [SEL Model]

$E = 5.2 L^{0.91}$ [Multivariate Model]

$L = (E/a)^{1/b}$

$L(\underset{\text{Model}}{SEL}) = (96/1.4)^{1/0.93} = 94264 \text{ LOC}$

$L(\underset{\text{Model}}{W-F}) = (96/5.2)^{1/0.91} = 24632 \text{ LOC}$

(b.) $D(SEL) = 4.6 L^{0.26}$

$= 4.6 (94.264)^{0.26}$

$= 15$ months

$D(W-F) = 4.1 L^{0.36}$

$= 4.1 (24.632)^{0.36}$

$= 13$ months

(c.) Productivity = $\frac{\text{Line of Code Produced}}{L/E}$ Per Person/month(year)

$P(SEL) = \frac{94264}{8} = 11783$ LOC/Person-year

$P(W-F) = \frac{24632}{8} = 3079$ LOC/Person-year

(d) Average manning = avg. no of Person required Per month in Project

$$M(SEL) = \frac{96 \text{ P-M}}{15 \text{ M}} = 6.4 \text{ Persons}$$

15 → duration

$$M(WOF) = \frac{96 \text{ P-M}}{13 \text{ M}} = 7.4 \text{ Persons}$$

Mode	Project Size	Nature of Project	Innovation	deadline of Project	development environment
Organic	Typically 2-50 KLOC	Small size Project experienced develop ers in familiar env. for eg: Pay roll, inventory Project	little	Not tight	Familiar & In house
Semi-detached	50-300 KLOC	Medium size Project, Medium size team, Avg. Previous experience on similar Projects. Eg: utility systems like Compilers, editors etc	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	large Project, real time systems, Complex interfaces, very little previous experience. Eg: ATMs, Air traffic Control	Significant	Tight	Complex hardware/ customer interfaces required

COMPARISON OF 3 COCOMO MODEL

COCOMO Model : (Constructive Cost model)

used for small to medium sized s/w Project.

3 modes of s/w development are considered in this model :

- organic
- semi-detached
- embedded

ORGANIC MODE : - small team of experienced developers develops s/w in a very familiar environment.

- Size of s/w development ranges from small (a few KLOC) to medium (a few tens of KLOC).

- In other 2 modes size varies from small to very large.

Embedded mode :- Project has tight constraints, which might be related to target processor and its interface with associated hardware.

Semidetached :- intermediate mode b/w organic & embedded mode.

Basic COCOMO eq. takes form:

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (E)^{d_b}$$

E = effort applied in Person-Month

D = development time in months

a_b, b_b, c_b, d_b are given in table.

Project	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

Average staff size (SS) = $\frac{E}{D}$ Persons

When Project size is known, Productivity level may be calculated as:

$$P = \frac{KLOC}{E} \quad KLOC/PM,$$

Suppose that a project was estimated to be 400 KL Calculate effort and development time for each of three models.

COCOMO eq. take the form:

$$E = a_b (KLOC)^{b_b}$$

$$D = c_b (KLOC)^{d_b}$$

estimated size = 400 KLOC

(i) Organic model-

$$E = 2.4 (400)^{1.05} = 1295.31 \text{ PM}$$

$$D = 2.5 (1295.31)^{0.38} = 38.07 \text{ M}$$

(ii) Semidetached model-

$$E = 3.0 (400)^{1.12} = 2462.79 \text{ PM}$$

$$D = 2.5 (2462.79)^{0.35} = 38.45 \text{ M}$$

(iii) Embedded model-

$$E = 3.6 (400)^{1.20} = 4772.81 \text{ PM}$$

$$D = 2.5 (4772.8)^{0.32} = 38 \text{ M}$$

effort calculate for embedded mode is app. 4 times organic mode. " " semidetached " 2 times of organic mode. But development time is app. same for all 3 modes.

system and

Intermediate model :- Basic model allowed for a quick and rough estimate, but it resulted in a lack of accuracy.

- Boehm introduced additional set of 15 Predictors called "Cost drivers".

- Cost drivers are used to adjust nomial cost of a project to actual project environment, hence increasing the accuracy of estimate.

Cost drivers are grouped into four categories:

1) Product attributes:
   (a) Required S/W reliability (RELY)
   (b) database size (DATA)
   (c) Product Complexity (CPLX)

2) Computer attributes:
   (a) Execution time constraint (TIME)
   (b) Main storage constraint (STOR)
   (c) Virtual machine volatility (VIRT)
   (d) Computer turnaround time (TURN)

3) Personnel attributes:
   (a) Analyst capability (ACAP)
   (b) Application experience (AEXP)
   (c) Programmer capability (PCAP)
   (d) Virtual machine experience (VEXP)
   (e) Programming language experience (LEXP)

equations used for this model are:

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

values of a_i, b_i, c_i, d_i are:

Project	a_i	b_i	C_i	d_i
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Detailed COCOMO Model; detailed model introduced a more capabilities:

1.) Phase - sensitive effort multipliers:- Some Phases (design, Programing, integration/test) are more affected than others by factors defined by cost drivers.
- detailed model provides a set of Phase sensitive effort multipliers for each cost driver.
- this helps in determining manpower allocation for each Phase of Project.

2.) Three - level Product hierarchy:- Three Product levels are defined. These are module, subsystem and system levels.

Development Phase: S/w development is carried out four successive Phase:

1.) PLAN / REQUIREMENTS :- - first Phase of development cycle.
- requirement is analyzed, Product Plan is setup and full Product specification is generated.
- Phase Consume 6% to 8% of effort & 10% to 40% of development time.
- This Percentages depend not only on mode (organic, semidetached or embedded), but also on size.

2.) Product design: - this phase concerned with determination of Product architecture and specification of subsystem.
- Phase requires 16% to 18% of effort and 19% to 38% of development time.

3.) Programming: The third Phase of COCOMO development cycle is divided into two sub-Phase: detailed design and Code/unit test. This Phase requires from 48% to 68% of effort and last from 24% to 64% of development time.

4.) Integration / Test :- - this Phase occurs befor delivery.
- mainly Consist of Putting testing Parts together and then Testing final Product.

- Phase requires 16% to 34% of nomial effort (13) and can last from 18% to 34% of development time.

Principle of effort estimate :-

Size equivalent : As s/w might be partly developed from s/w already existing (i.e re-usable code), a full development is not always required.

In such cases, parts of design document (DD%), Code(C%) and integration (I%) to be modified are estimated. An adjustment factor, A, is calculated by means of foll. eq.:

$$A = 0.4\,DD + 0.3\,C + 0.3\,I \quad \text{— (1)}$$

Size equivalent is obtained by

$$S(\text{equivalent}) = (S \times A)/100 \quad \text{— (2)}$$

S = thousands of line of code (KLOC)

(for each phase)

$$E_p = \mu_p E \quad \text{— (3)}$$

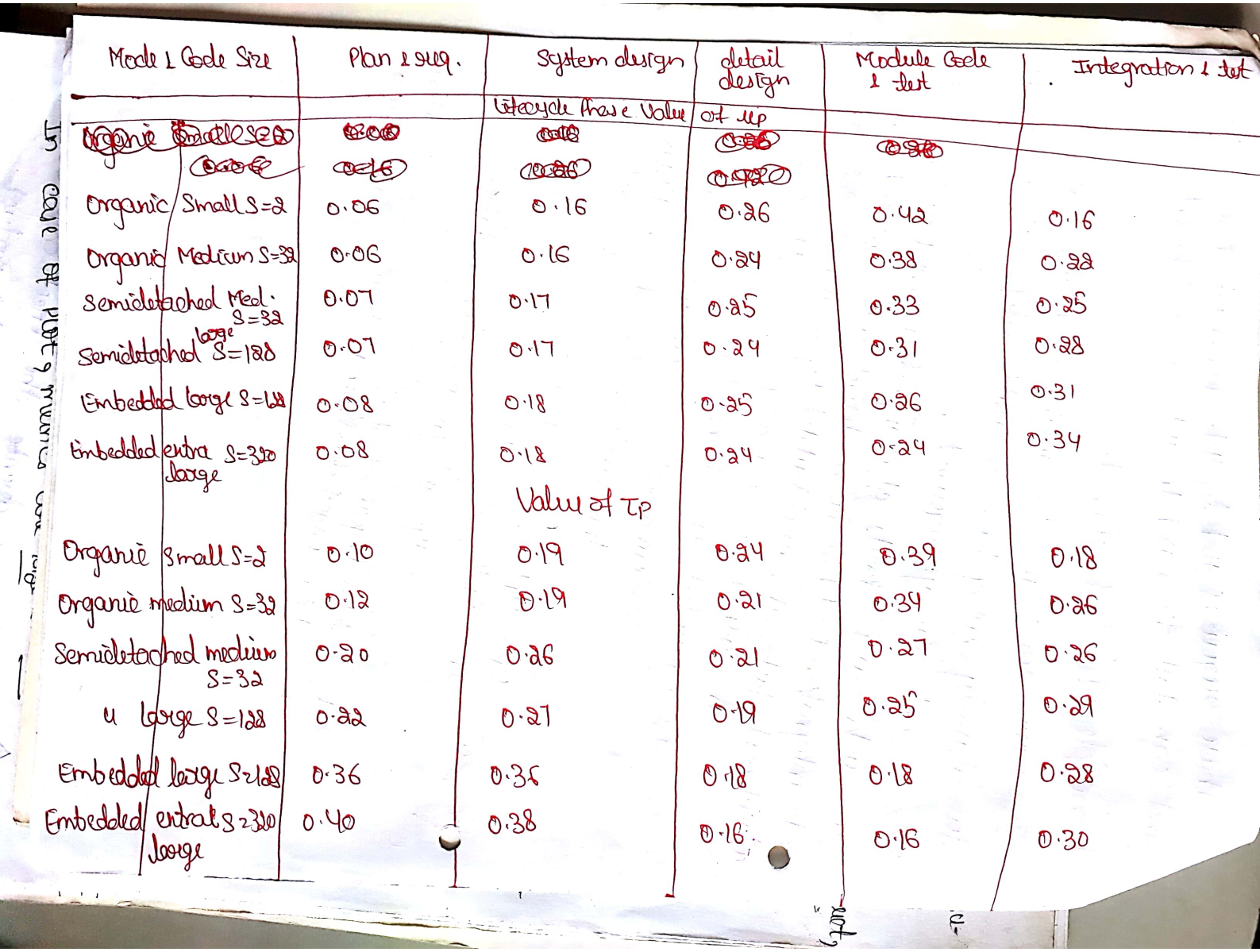
$$D_p = \tau_p D \quad \text{— (4)}$$

μ_p and τ_p are given in table.

E = total Project effort.

D = total Project development time.

total 5 Phases are assumed.



Mode L Code Size	Plan & Req.	System design	detail design	Module Code & test	Integration & test
		Lifecycle Phase Value of Ip			
Organic Small S=2	0.06	0.16	0.26	0.42	0.16
Organic Medium S=32	0.06	0.16	0.24	0.38	0.22
Semidetached Med. S=32	0.07	0.17	0.25	0.33	0.25
Semidetached large S=128	0.07	0.17	0.24	0.31	0.28
Embedded large S=128	0.08	0.18	0.25	0.26	0.31
Embedded extra large S=320	0.08	0.18	0.24	0.24	0.34
		Value of Tp			
Organic Small S=2	0.10	0.19	0.24	0.39	0.18
Organic medium S=32	0.12	0.19	0.21	0.34	0.26
Semidetached medium S=32	0.20	0.26	0.21	0.27	0.26
u large S=128	0.22	0.27	0.19	0.25	0.29
Embedded large S=128	0.36	0.36	0.18	0.18	0.28
Embedded extra large S=320	0.40	0.38	0.16	0.16	0.30

In case of Post & maintenance use

Plan & requirement + System design = requirement and Product design

Book numerical.

Cocomo - II :- is revised version of original COCOMO and is developed at University of Southern California under leadership of Dr. Barry Boehm.

Categories of applications/Projects :-

(1) End User Programming :- - category is applicable to small systems, developed by end user using application generators.

- application generators are spreadsheets, extended query system, report generators etc.
- end user may write small Pgs. using application generators.
- end user may not have sufficient knowledge about Computers and S/W engineering Practices.
- they may have in-depth knowledge about their business needs and Practices.
- this knowledge may motivate them to develop an app. using user friendly tools like MS-Excel, MS-Access etc.

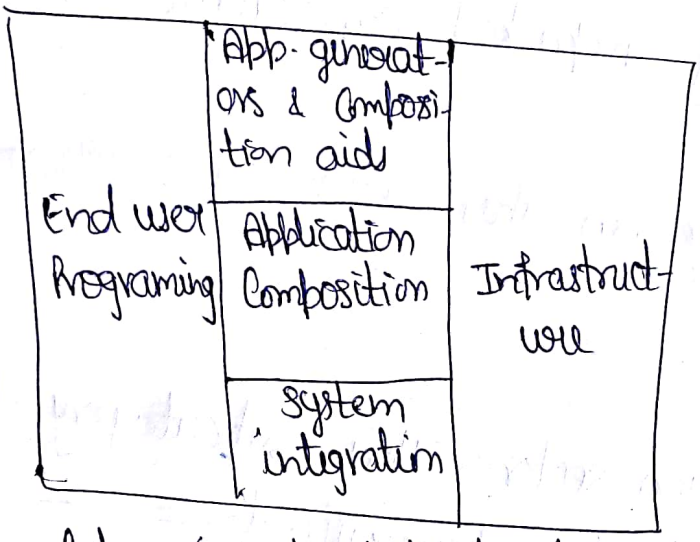


fig-(1)

Categories of application / Projects.

(ii) Infrastructure Sector :- this category is applicable to infrastructure development i.e s/w that provide infrastructure like O/S System, database mgt system etc.

Eg's are : DB2, UNIX, ORACLE.

- Infrastructure developers generally have good knowledge ~~about applications~~ of s/w development and s/w engineering practices.
- little knowledge about application.

(iii) Intermediate Sector :- this category is partitioned into 3 sub categories in fig-(1)

- s/w developers will need to have good knowledge of s/w development and s/w eng. practices.
- experience and expertise of infrastructure s/w and indept domain knowledge of one or more app.

Application generator & Composition aids: this subca will create largely prepacked capabilities for user programing.

- Firms operating in this sector are: Microsoft, Lotus, Oracle, IBM..

Application composition sector: This subcategory deals with applications which are too diversified

- Components will be GUI builders, database or object managers.

- These applications are complex, large, versatile, diversified etc.

System integration: - deals with large scale, highly embedded, systems.

Stages of COCOMO-II :- end user programing sector doesn't need a COCOMO-II model.

- its applications are normally developed in hours to days.

Cocomo - II includes 3 stage.

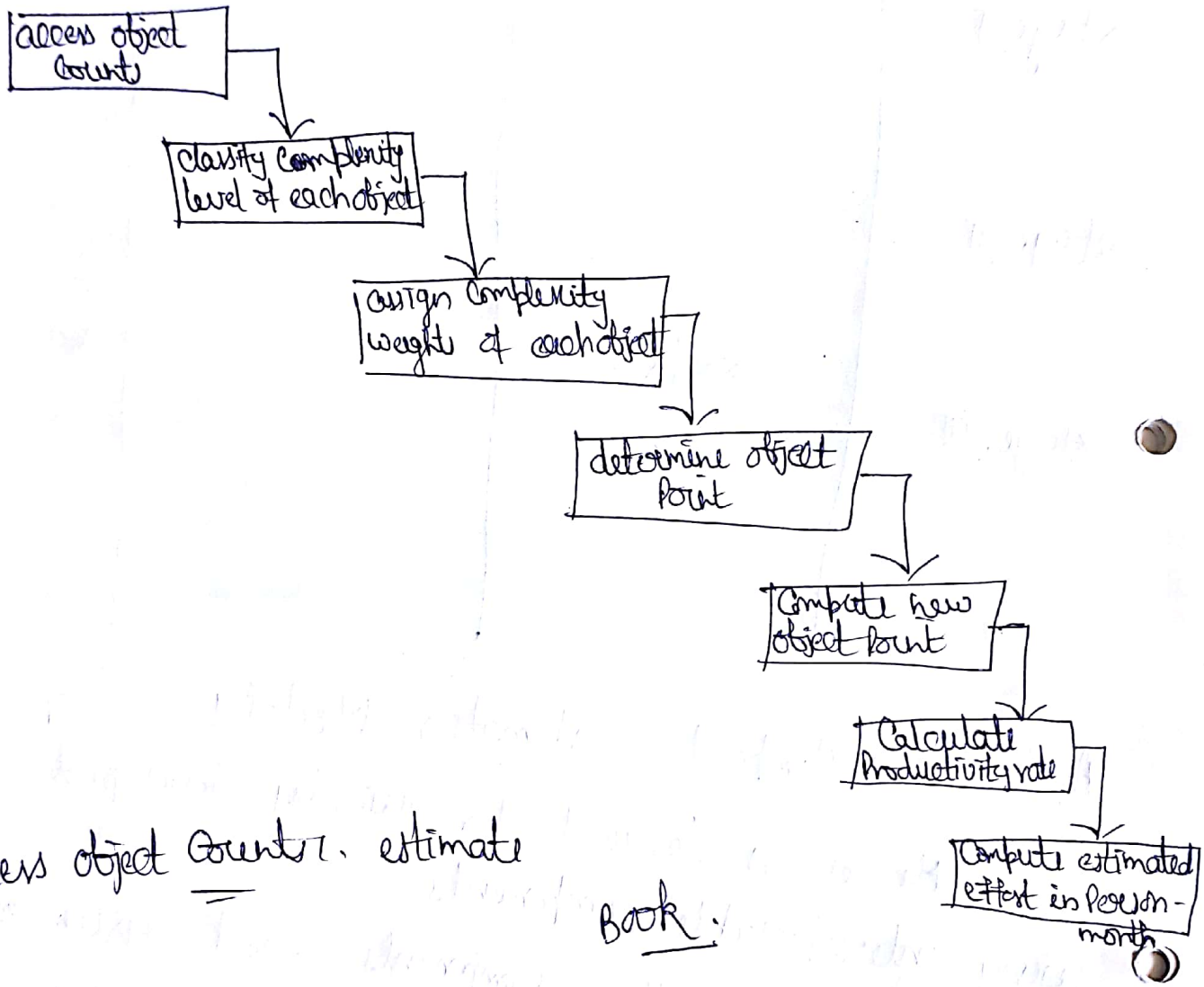
Stage No.	Model Name	App. for type of Project	App.
Stage I			
stage II			
stage III	Boot.		

Application Composition estimation Model :-

- Model is designed for quickly developed applications using interoperable components.
- egs are of these components based system are GUI builders, database or object manager.
- This model can also be used for Prototyping Phase of application generator development, infrastructure sector and system integration Projects.
- firstly, size is estimated using Object Point.
- Object Point are easy to identify & Count.

- Object in object points defines screens, reports and modules as object.

Steps Required for estimation of effort in Person-month:-



1.) Access object Counts. estimate

Book.

Early design model:- Boehm-II model the base eq. of the form:

$$PM_{nominal} = A * (size)^B$$

$PM_{nominal}$ = effort of Project in Person month

A = Constant representing nominal Productivity, set 2.5

B = Scale factor Size = size

- This model uses UFP (unadjusted function points) as measure of size.

- model is used in early stages of a s/w Project when very little may be known about size of Product to be developed.

- This model can be used in either Application Generator, System integration, or infrastructure development Sector.

If $B = 1.0$, there is linear relationship of effort and size.

If $B \neq 1$, there will be a non-linear relationship b/w size and effort.

If $B < 1.0$, rate of increase of effort decreases as size of Product increases.

If $B > 1.0$, rate of inc- of effort increases as size of product increases.

- Application Composition model assume $B = 1$
- but early design model assume value of $B > 1$

Scaling factors required for Calculation of value of B.

Book.

S/w metrics.

Metric is quantitative measure out of different measures that represent entity of interest.

A metric is a quantifiable measurement of s/w Product, Process or Project that is directly observed, calculated or Predicted.

S/w measurement is continuous process of defining, collecting and analyzing data on s/w development Process and its Product in order to understand and Control Process & its Products and to supply meaningful inf to improve that Process & its Product.

eg, a Plot Purchased for house Construction.

The length & breath are basic measures. They are measured using a measuring tape from one point to other.

So length & breadth are measures.

The Process of using tape is measurement.

Metrics are built using these measures.

In case of Plot, metrics are size & area.

Curve

More Sophisticated

S/w metrics can be used to indicate basic attributes of S/w. The indicative metrics are:

Size :- Line of Code, function Point Count.

Quality :- Reliability, accuracy, dependability are measured through measure like errors, failures & no. of runs b/w failures.

Execution time :- Process time of execution of a critical process.

S/w metrics can be divided into two categories based on type of measure:

(1.) Direct Measure :- It includes cost, effort, LOC, response time etc.

(2.) Indirect Measure :- include functionality, quality, complexity, efficiency, ease of use etc.

Categories of Metrics :- 3 categories:

(i.) Product metrics :- describe characteristics of Product as size, complexity, performance, efficiency etc.

(ii.) Process metrics :- describe the effectiveness and quality of Processes that Produce S/w Product.

eg are:-

* effort required in Process.
* time to Produce Product.
* no. of defects found during testing.

(iii.) Project metrics :- describe Project characteristics and execution.

* no. of s/w developers
* Cost & schedule
* Productivity.

TOKEN COUNT:- Size metrics (LOC & function Count) are discussed in Project Planning.

Major Problem with LOC is that it is not consistent bc some lines are more difficult to code than others. So LOC may change from developer to developer.

Tokens are classified as either operators or operands.

Size is defined as:

$$\eta = \eta_1 + \eta_2$$

η = vocabulary of a Pg.

η_1 = no. of unique operators.

η_2 = no. of unique operands.

Length of Pg. in terms of total no. of tokens used is:

$N = N_1 + N_2$ → total occurrences of operands

↓ Pg. length

→ total occurrences of operators

For machine lg. Pg. where each line consist of one operator and one operand, the Pg. length is:

$$N = 2 * LOC.$$

Another measure for size of Pg. is called "Volume".

$$V = N * \log_2 n$$

The unit of measurement of volume is common unit for size "bits".

Pg. level $L = V^*/V$.

V^* ↓ Potential volume V^*

V ↓ volume

value of L ranges btw 0 and 1, with $L=1$ representing a Pg. written at highest possible level (i.e with minimum size).

The inverse of Pg. level is termed difficulty.

$$D = 1/L$$

Effort is defined as: $E = V/L$

But $D = \frac{1}{L}$

$E = V * D$.

no. of conceptually unique input & output parameters

Potential Volume:

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

↑ two unique operators

Estimated Pg. length :-

Size of Pg. is total no. of tokens, referred to as Pg. length, N.

Estimated length eq. is denoted by $\hat{N}$ and is defined by :

$$\hat{N} = \eta_1 \log_2 \eta_1 + \eta_2 \log_2 \eta_2$$

In Fig. 4.2 (of chapter 4) has 14 unique operators & 10 unique operands.

$$\hat{N} = 14 \log_2 14 + 10 \log_2 10$$

$$= 53.34 + 33.22 = 86.56$$

operator :- (i.) any symbol or keyboard in a that specifies an algorithmic action is considered an operator.

(2.) Punctuation marks

(3) +, -, *, Command name such as "while", "for", "printf". (4.) :=, braces, parentheses & fn. name eof "end of file".

operands :- (i.) symbol used to represent data.

(ii.) variable, Constant & labels.

Pg. length can be estimated using Halstead's length estimator :-

$$N_j = \log_2(\eta_1!) + \log_2(\eta_2!)$$

Slight modification of Halstead length estimator i.e.

$$N_B = n_1 \log_2 n_2 + n_2 \log_2 n_1$$

The Halstead estimator was further modified by Caird & Agresti by substituting $\sqrt{n_1}$ for $\log_2 n_1$ and $\sqrt{n_2}$ for $\log_2 n_2$, thus

$$N_C = n_1 \sqrt{n_1} + n_2 \sqrt{n_2}$$

Counting Rules for C language :- Book Page 248.

Estimated Pg. level / Difficulty :-

To estimate the Pg. level,

$$\hat{L} = 2n_2 / (n_1 N_2)$$

Hence $\hat{D} = \frac{1}{\hat{L}} = \frac{n_1 N_2}{2 n_2}$

Effort & Time :-

Halstead proved that effort required to implement a Pg. increases as size of Pg. increases.

$E = V / \hat{L} =$

we know $\hat{D} = \frac{1}{\hat{L}}$ $\quad V = N * \log_2 n$

So $E = V * \hat{D}$

$= \frac{n_1 N_2 N \log_2 n}{2 n_2}$

Programming time $T = E/\beta$ (47)

and β is normally set to 18.

To

Language level:- several languages may be used on a regular basis for S/w development.

Halstead and researchers determined language level for various languages are shown in table 6.1

Eg:- 6.1 Book.

Potential Volume:-

$$V^* = (2 + \eta_2^*) \log_2 (2 + \eta_2^*)$$

η_2^* → no. of conceptually unique input & output parameters (either

2 → represent two unique operators for procedure call:
(i.) Procedure name
(ii.) grouping symbol that separates procedure name from its parameters.

$\lambda = L \times V^*$

$= L^2 V$.

...stem flow model)

Data Structure metrics

Need for s/w development and other activities are to process data.

Some data is input to a system, pg. or module;

Some data is output from a system, pg. or module:

pg.	Data Input	Internal data	data output
Payroll	name/ Social Security no/ Pay Rate/ no. of hours worked	withholding rates Overtime factors Insurance premium rates	Gross Pay withholding Net Pay

Book:

A count of amount of data input to, processed in and output from s/w is called "data structure metric".

eg:- imp. set of metric is A pg. has 25 input parameters, 35 internal data items & 10 output parameters.

Amount of data: Most Compilers and assemblers have an option to generate a cross-reference list, indicating line where a certain variable is declared and line or lines where it is referenced.

Such a list is useful in debugging and maintenance and can help determine amount of data in the Pg.

Eg: Fig 6-2, it input = work hours, pay rates, Computers gross pay, taxes & net pay.

C, Compiler Produces a cross-reference listing for this Pg. in Fig 6.3.

Method to determine amount of data:- is to count no. of entries in the cross-reference list.

* Exclude from count those variables that are defined but never used.

The definition of these variables may be made for future reference, but they don't effect the operation characteristics of Pg.

Such a count variables will be referred to as VARS.

In fig. 6.2 VARS = 7

Count of VARS depends on following definition :-

A variable is a string of alphanumeric characters that is defined by a developer and that is used to represent some value during either compilation or execution.

Fig. 6.3.

(1.) But items feof stdin are related to I/O.

So feof — 9
stdin — 09 } items not counted as VARS.

(2.) Constant — 13 of whose value 0.25 not counted as VARS.

Halstead introduced a metric that he referred to as η_2

η_2 = VARS + unique constants + labels

for Fig 4.2

6 variables = X, N, I, J, SAVE, IM1

1 label = SORT → treated as labels since it is label that is used by any other fig. or sub fig.

3 Constants = {1, 2, 0}

$\eta_2 = 6 + 3 + 1$

$= 10$

In fig. 6-2 (6)

7 variables { check, gross, hours, net, pay, rate, tax).

1 Constant = 0.25

no label

$$\eta_2 = 7+1 = 8$$

Halstead further defined metric ~~total~~ occurences of operands and named in N_2.

fig. 6.5 enclosed each operand occurence in brackets.

So $N_2 = 30$

The metrics VARS, η_2 and N_2 are most Popular data structure measures.

Usage of data within modules- In fig. 66 fg. "bubble" input two related integer arrays (a and b) of same size upto 100 elements each. fig. 6-6 = swap & sort

Live variables :- while Constructing fg. "bubble", developer created a variable "last".

Definition of live variable:-

1.) A variable is live from beginning of a procedure to end of procedure.

2.) A var. is live at a particular st. only if it is referenced a certain no. of sts. b/f or after that st.

3.) A var. is live from its first to its last reference within a procedure.

Average no. of LIVE variables :- (LV), which is sum of count of live variables divided by count of executable statements in a procedure.

Fig. 6.7 live variables for fg. in Fig. 6.6.

$$\text{average} = \frac{\text{Count of live variable}}{\text{Count of executable statements}} = \frac{124}{34} = 3.647$$

Variable Span :- Two variable can be alive for same no. of statements, but their use in a fg. can be markedly different.

Fig. 6.8 :- In this a and b are alive for same 40 statements (21-60), but "a" is referred to 3 times and "b" is mentioned only once.

A metric that captures some of essence of how often a variable is used in Pg. is called "span" Sp. (7)

In this a has 4 spans
and b has 2

So "a" is being used more than "b".

So this metric is no. of statements b/w two successive references of same variable.

for a Pg. that reference a variable in n sts, there are (n-1) spans for that variable.

* Size of a span indicates no. of statements that lies b/w successive uses of a variable.

MAKING PG-WIDE METRICS FROM INTRA-MODULE METRICS :-

Each of the metric discussed before, is intended to be used within a module.
But it is possible to extend each one into an inter-module metric.

$$\overline{LV}_{Program} = \frac{\sum_{i=1}^{m} \overline{LV_i}}{m}$$

$\overline{LV_i}$ = avg. live variable metric computed from ith module.

Scanned by CamScanner

average span size ($\overline{SP}$) for a pg. of n spans

$$\overline{SP}_{Program} = \frac{\sum_{i=1}^{n} \overline{SP_i}}{n}$$

PROGRAM WEAKNESS: A pg. consist of modules.

$\overline{LV}$ = avg. no. of live variable

γ = avg. life of variable

module weakness $WM = \overline{LV} * \gamma$

$$\text{Program weakness} = \frac{\left[\sum_{i=1}^{m} WM_i\right]}{m}$$

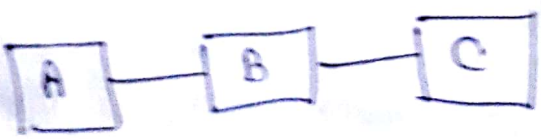
WM_i = weakness of ith module

WP = weakness of pg.

m = no. of modules in pg.

SHARING OF DATA AMONG MODULES :- A pg. contains several modules and shows coupling among modules. So it may be desirable to know amount of data being shared among modules.

Fig. (1)



Three modules form an imaginary program.

Complex ② Relationship b/w A & C.

data shared "Pipes" of data shared among modules.

for eg:- fig. 6, 3 subprograms A, B and C.

If we include information that represent amount of data "Passed" among subpgs, we could envision Pipes b/w subpgs.

diameter of each pipe could represent quantity or volume, of data sent from one subpg. to be used in other.

In fig. (2.), show same three subpgs. with pipes representing data shared b/w A and B & between B and C.

A-B shared data is implicitly less than B-C shared data.

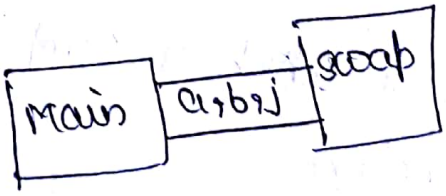


fig — (3.)

fig (3.) shows a pictorial representation of data shared between the main pg. of bubble and Procedure scoop.

"bigger the Pipe" in between any two modules, more Complex is their relationship.

INFORMATION FLOW METRICS.

Simple system consists of components, it is to work that these components do and how they are fitted together that influences Complexity of a system.

* If a Component has to do numerous discrete tasks, it is said to be lack 'Cohension'. assgn. 1

* If it passes inf. to, and/or accept inf. from many other Components within system, it is said to be highly coupled. assg. 2 or more.

* Component that are highly coupled and lack cohension tend to be less reliable and less maintainable.

COMPONENT :- any element identified by decomposing a (S/W) system into its constituent parts.

Cohension :- degree to which a component performs a single function. ✓

Coupling :- term used to describe degree of linkage b/w one Component to others in same system.

b/w this module ✓

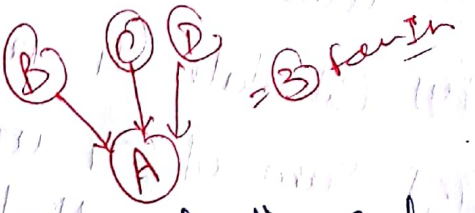
b/w module ✓

High Cohension & Low Coupling.

Intf. Flow metric model degree of Cohension and Coupling for a particular system Component. (8)

for

Basic Information Flow model :-



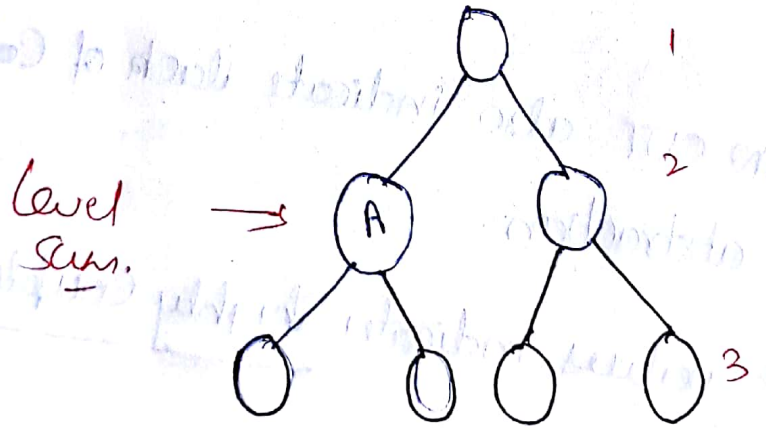
three measures are :-

(1.) 'FANIN' is simply a count of no. of other Component that can call, or Pass Control, to Component A. B call A

(2.) 'FAN OUT' is no. of Components that are called by Component A.

(3.) Measure the INFORMATION FLOW index of Component A, abbreviated as IF(A)

$$IF(A) = [FAN\ IN(A) \times FAN\ OUT(A)]^2 .$$



Steps to deriving simple IF metrics are :

(1.) Note level of each Component in system design.

(2.) for each Component, Count no. of calls to that Component - this is the FAN IN of that Component.

(3.) for each Component, Count no. of calls from that Component. for Component that call no other, assign a FAN-OUT value of one.

(4) Calculate IF value for each Component using above formula.

(5.) Sum of IF value for all Components within each level which is called as LEVEL SUM.

(6.) Sum the IF values for total system design which is called SYSTEM SUM.

(7.) for each level, rank the Components in that level/acc. to FAN IN, FAN OUT and IF values. Three line Plots should be Prepared for each level.

(8.) Plot LEVEL SUM value for each level using a line Plot.

High FAN IN values indicates Components that lack Cohesion.

High FAN OUT also indicate lack of Cohesion or mixed level of abstraction.

High IF values indicate highly Coupled Components.

SYSTEM SUM :- final item of inf. flow metric is SYSTEM SUM value. This give an overall Complexity rating for design in terms of IF metrics.

Difference b/w simple and sophisticated IF models lies in definition of FAN IN and FAN OUT.

For a Component A let!

a = no. of Components that call A.

b = } Book.
c =
d =

FAN IN(A) = a+b+c+d

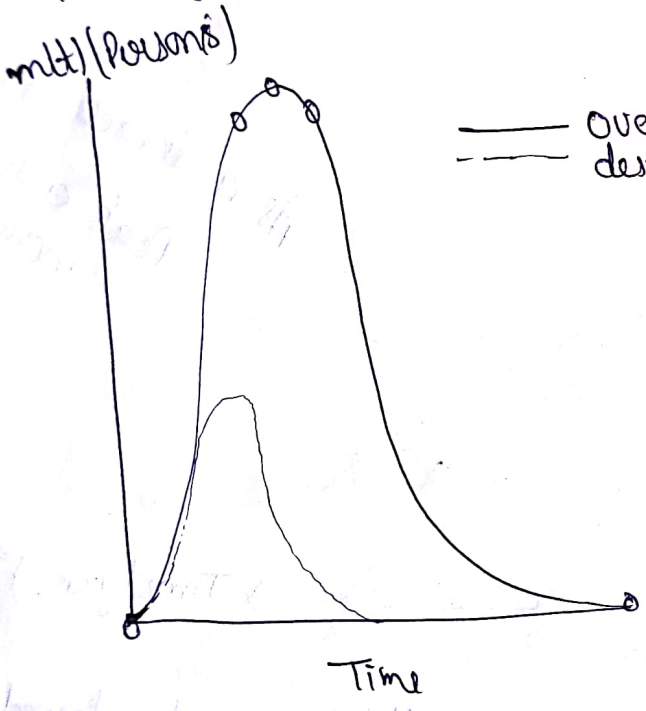
also FAN OUT(A) = e+f+g+h

Metric analysis :- Book.

Putnam Resource allocation model. ①

- Norden of IBM observed that Rayleigh curve can be used as an approximate model for a range of hardware development projects.

- This approach was later extended by Putnam to apply to S/w Projects.



Norden and Putnam observed that manpower curve to give peak, then exponentially trail off as a function of time.

Norden/Rayleigh Curve; Norden/Rayleigh eq. represents manpower measured in persons per unit time as a fn. of time. It is usually expressed in person-year/year (PY/YR).

$$m(t) = \frac{dy}{dt} = 2Kate^{-at^2} \quad \text{---- (1.)}$$

dy/dt = manpower utilization rate per unit time, 't' = elapsed time.

a = Parameter that affect shape of curve.

K = area under the curve in interval $[0, \infty]$.

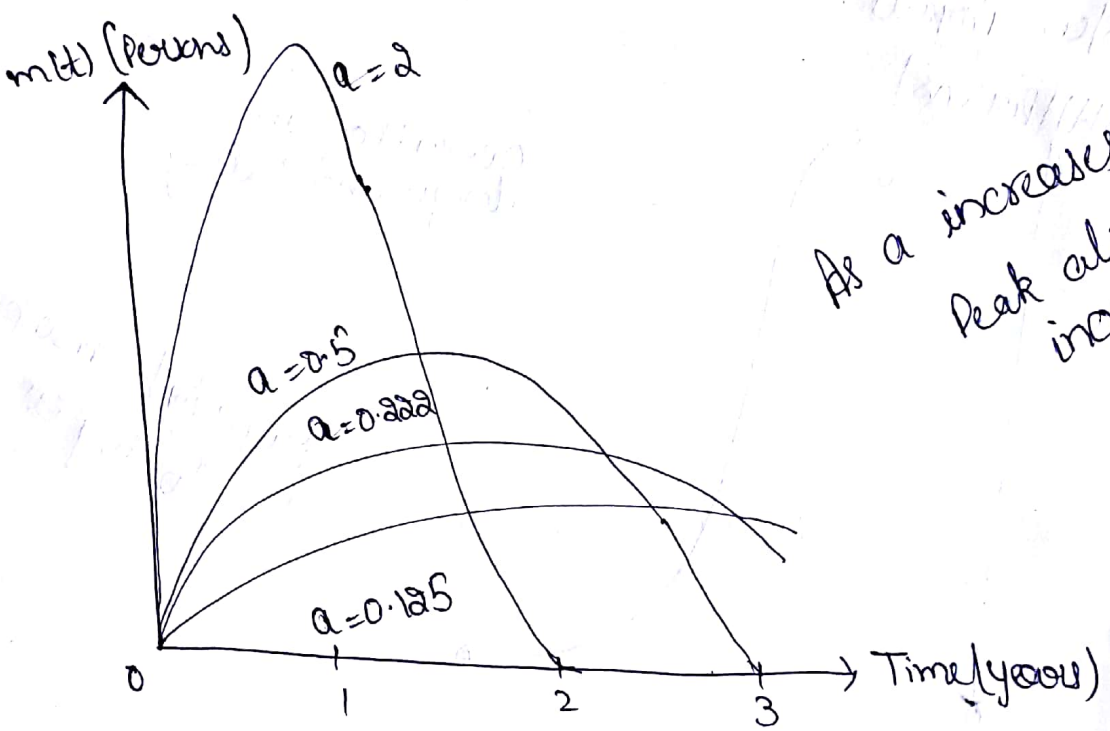
Integrate eq. (1) on interval $[0, t]$

$$y(t) = K[1 - e^{-at^2}] \quad \text{---(2)}$$

, $y(t)$ = cumulative manpower used upon time t.

$y(0) = 0$

$y(\infty) = K$



As a increases Peak also increases.

Relationship btw "td" and "a" can be found as:

$$\frac{d^2y}{dt^2} = 2Kae^{-at^2}[1 - 2at^2] = 0$$

$$1 - 2at^2 = 0 \qquad t_d^2 = \frac{1}{2a}$$

$$1 = 2at^2$$

$$1/2a = t_d^2$$

So $t_d^2 = 1/2a$ so $a = \frac{1}{2t_d^2}$

td = time where max. effort rate occurs.

Put td value in eq. (2)

$$E = y(t) = K\left(1 - e^{-\frac{a t_d^2}{2 t_d^2}}\right)$$

$$= K\left(1 - e^{-0.5}\right)$$

$$E = y(t) = 0.3935\,K$$

Put value of $a = \frac{1}{2t_d^2}$ in eq. (1)

(3) $$m(t) = \frac{dy}{dt} = 2Kate^{-at^2}$$

$$= \frac{2K}{2t_d^2} t e^{-\frac{t^2}{2t_d^2}}$$

$$= \frac{K}{t_d^2} t e^{-t^2/2t_d^2}$$

At time $t = t_d$, Peak manning $m(t_d)$ is obtained and denoted by m_0.

(4) $$m_0 = m_{t_d} = \frac{K}{t_d^2} t e^{-\frac{t_d^2}{2t_d^2}}$$

$$= \frac{K}{t_d\sqrt{e}}$$

K = total Project Cost (or effort) in Person-year.

t_d = delivery time in year.

m_0 = no. of Persons employed at Peak.

$K = 95$ PY [95 Person Project Cost]

Peak development time $t_d = 1 + \frac{9}{12} = 1 + 0.75$

$= 1.75$ years

Peak manning = ? $m_0 = \frac{K}{t_d\sqrt{e}}$

$$= \frac{95}{1.75 \times 1.648} = 32.94 = 33 \text{ persons.}$$

Average rate of s/w team build up =

$$\frac{mo}{td}$$

$$= \frac{33}{1.75} = 18.8 \text{ person/year.}$$

or

1.56 person/month.

K = 600 PY

td = 3 year 6 month. = 3.5 years

(a) mo = ? = Peak manning

$$mo = \frac{K}{td\sqrt{e}} = \frac{600}{3.5 \times 1.648} = 104 \text{ persons.}$$

(b) what is manpower cost after 1 year 2 months.

t = 1 year 2 month = 1.17 years

$$a = \frac{1}{2td^2} = \frac{1}{2 \times (3.5)^2} = 0.041$$

manpower cost $y(t) = k[1 - e^{-at^2}]$

$$y(1.17) = 600[1 - e^{-0.041(1.17)^2}]$$

$$= 32.6 \text{ PY.}$$

culty Metric)

$$m(t) = \frac{dy}{dt} = 2Kate^{-at^2} \quad \text{---(1)}$$

differentiate eq. (1)

$$m'(t) = \frac{d^2y}{dt^2} = 2Kae^{-at^2}(1-2at^2)$$

for $t = 0$

$$m'(0) = 2Ka = \frac{2K}{2t_d^2} \quad \text{---(2)} \qquad a = \frac{1}{2t_d^2}$$

Ratio $\frac{K}{t_d^2}$ is called difficulty & denoted by D

$$D = \frac{K}{t_d^2}$$

Person/year

This shows that project is more difficult to develop when manpower demand is high or when time is short.

$$m_0 = \frac{K}{t_d\sqrt{e}}$$

$$D = \frac{K}{t_d^2} = \frac{m_0 t_d\sqrt{e}}{t_d^2} = \frac{m_0\sqrt{e}}{t_d}$$

Thus difficult projects tend to have a higher peak manning for a given development time.

Manpower buildup :- D is depend upon K & td.

$$D = \frac{K}{t_d^2}$$

$$\frac{d}{dt_d}\left(\frac{K}{t_d^2}\right) = K t_d^{-2}$$

$$= -2K t_d^{-3}$$

$$D'(t_d) = \frac{-2K}{t_d^3} \text{ Person/year}^2 = \frac{-2K}{t_d^3}$$

$$D'(K) = \frac{1}{t_d^2} \text{ year}^{-2}$$

Putnam observed that project scale is increased the development time also increases to such extent that quantity K/t_d^3 remain constant around a value, which could be 8, 15 or 27.

This quantity is represented by Do and expressed as :

$$Do = \frac{K}{t_d^3} \text{ Person/year}^2.$$

4.14. $K = 600$ PY

$t_d = 3$ year 6 month $= 3.5$ years.

D : difficulty $= \frac{K}{t_d^2} = \frac{600}{(3.5)^2} = 49$ Person/ye

Manpower build up $Do = \frac{K}{t_d^3} = \frac{600}{(3.5)^3} = 14$ Person/ year^2

ductivity versus Difficulty:

Productivity = no. of lines of code developed per person-month.

Putnam has observed that Productivity is proportional to difficulty.

$$P \propto D^{\beta} \quad \text{—— (1)}$$

Average Productivity may defined as:

P = Line of Code Produced / Cumulative manpower used to Produce code.

$$P = S/E \quad \text{—— (2)}$$

S = Line of Code Produced

E = manpower used from $t=0$ to $t=td$

Putnam observed that

$$P = \phi D^{-2/3} \quad \text{—— (3)}$$

we eq. (2)

$$P = S/E$$

i.e $S = P E$

$= \phi D^{-2/3} \cdot E$

$= \phi D^{-2/3} (0.3935 K)$

$E = 0.3935 K.$

using eq. $D = \frac{K}{t_d^2}$ $E = 0.39$

$$S = \phi \left(\frac{K}{t_d^2}\right)^{-2/3} (0.3935 K)$$

~~$S = 0.3935 \phi K (0.3935)$~~

$$\boxed{S = 0.3935 \phi K^{1/3} t_d^{4/3}} \quad \text{---------- (4.1)}$$

Replace $= 0.3935 \phi$ by C.

C = Technology factor.

C = 610 to 57314.

Modify eq. (4.1)

$$S = C K^{4/3} t_d^{4/3}$$

$$C = S \cdot K^{-1/3} t_d^{-4/3}$$

Trade off b/w time versus cost: For developing a s/w of size = S.

using eq. $S = C K^{4/3} t_d^{4/3}$

$$\frac{S}{C} = K^{1/3} t_d^{4/3}$$

$$K^{4/3} = \frac{S}{C} \cdot \frac{1}{t_d^{4/3}}$$

$$K = \left(\frac{S}{C}\right)^3 \times \frac{1}{t_d^{4/3 \times 3}} \quad \text{or} \quad K = \left[\frac{S}{C}\right]^3 \times \frac{1}{t_d}$$

If $C = 5000$

$S = 50,000\ (0.C$

$$K = \frac{1}{t_d^4}\left(\frac{50,000}{5000}\right)^3$$

$$K = \frac{1}{t_d^4}(100)^3$$

Development sub-cycle:

$$m_{(t)} = \frac{dy}{dt} = 2Kate^{-at^2}$$

let $m_d(t)$ and $y_d(t)$ design manning.

$$m_d(t) = 2K_d bte^{-bt^2}$$ Put $K = K_d$, $a = b$

$$y(t) = K[1 - e^{-at^2}]$$

$y(t)$ = manpower used upon time t

$$y_d(t) = K_d[1 - e^{-at^2}]$$

for good Practical assumption to assume that development will be 95% Completed by time t_d.

$$\frac{y_d(t)}{K_d} = 1 - e^{-at^2} = 0.95$$

from Previous $a = \frac{1}{2t_d^2}$ so $b = \frac{1}{2t_{od}^2}$

t_{od} = time at which development curve exhibits a Peak manning.

t_{od} = development Peak manning

t_d = development time

$$t_{od} = t_d/\sqrt{6}$$

$t_{od} = \frac{t_d}{\sqrt{6}}$.

$t_d = t_{od}\sqrt{6}$.

$K_d = \frac{K}{6}$.

Now eq. $D = \frac{K}{t_d^2}$

eq. $m_d(t) = 2K_d b t e^{-bt^2}$ — (*)

diff. eq. (*)

$$\left(\frac{dm}{dt}\right)_0 = \frac{K}{t_d^2} = \frac{K_d}{t_{od}^2} = \left(\frac{dm_d}{dt}\right)_0$$

and acc. $t_{od} = \frac{t_d}{\sqrt{6}}$

$$K_d = \frac{K}{6}$$

we can write as

difficulty $\rightarrow D = \frac{K}{t_d^2} = \frac{K_d}{t_{od}^2}$ t_{od} for sub-cycle.

but

manpower build up $= D_o = \frac{K}{t_d^3} = \frac{K_d}{\sqrt{6}\, t_{od}^3}$

4.15]

(a.) duration t_d = 3 year 5 months
= 3.41 years

Now

K_d = 90 KY

$$\frac{y_d(t_d)}{K_d} = 0.95$$

manpower cost ← $y_d(t_d) = 0.95 \times K_d$
$= 0.95 \times 90$
$= 85.5$ PY.

(b.)

dev. peak time ← $t_{dd} = \frac{t_d}{\sqrt{6}} = \frac{3.41}{\sqrt{6}} = 1.39$ years

(c.) $K_d = y_d(t_d) / 0.95$
$= 0.95 \times 85.5 / 0.95 = 90$

$K = 6K_d$
$= 90 \times 6 = 540$ PY

difficulty ← $D = K / t_d^2 = 540 / (3.41)^2 = 46$ Person/year

$D_0 = \frac{K}{t_d^3} = 540/(3.41)^3 = 13.6$ Person/year^2.

↓
manpower build up