

UNIT 2

①

Software Project Management

Management in simple words is defined as the art of getting work done through other people. But the modern management definition includes the terms 'efficiency' and 'effectiveness'.

Project Management involves four types of activities, broadly :-

- (i) Planning
- (ii) Monitoring
- (iii) Controlling
- (iv) Terminal Analysis

Management is not an exact science. The situations varies according to project characteristics. The project manager has functions like writing project proposals, cost estimations, staffing, scheduling, resource allocation software configuration management, interaction with clients.

Project Planning

The first and foremost component of Project Management is effective Planning. Lack of Planning is the primary cause of failures such as poor quality, cost over-run, late delivery and high maintenance cost for software. Before the project begins, the manager and software team must estimate the work to be done, resources that will be required and the time that will elapse from start to finish, is known as Planning. Project Planning the single most important activity which makes base for monitor and control.

Objectives Of Project Planning

The project planning is aimed to achieve following goals: —

- (i) The estimate of resources are made and documented.
- (ii) The cost estimation is done in advance so that the project could be monitored later on, to remain within those costs.
- (iii) The time scheduling and the commitments made according to schedules are documented.

①

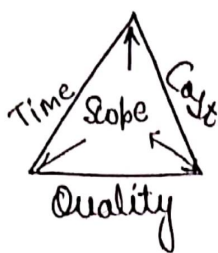
Why study Software Engineering?

- To acquire skills to develop large programs.
- A

Software Project Management

- * The job pattern of an IT company engaged in software development can be seen. Split into two parts:-
 - * Software Creation
 - * Software Project Management
- * A project is well-defined task, which is a collection of several operations done in order to achieve a goal (for example software development and delivery).
- * A project can be characterized as:
 - Every project may have a unique and distinct goal.
 - Project is not a routine activity or day to day operation.
 - Project comes with a start and end time.
 - Project needs adequate resources in terms of line of power, manpower, finance material, and knowledge-bank.
- * Software Project :-
 - A software project is the complete procedure of software development from requirement gathering to testing and maintenance carried out according to the execution methodologies.
- * Needs of Software Project Management :-
 - Software is to be an intelligence product. Software development is a kind of all new stream in world business and is very little experience. In building, software products are tailor made.

②



→ Software project Management is essential to incorporate user requirement along with budget and time constraints.

* Software Project Manager :-

- A Software project Manager is a person who undertakes the responsibility of executing the software project.
- Software project manager is thoroughly aware of all the phases of SDLC.

* Managing People :-

- Acts as project leader
- Liaison with Stakeholder
- Managing human resources.
- Setting up reporting hierarchy.

* Software Management Activities :-

- ① Project Planning
- ② Scope Management
- ③ Project Estimation

① Project Planning :-

→ Software project planning is task which is performed before the production of software actually starts.

② Scope Management :-

- define the scope
- decide its verification and control

UNIT-II

Software Project Management

Introduction :-

Developing computer software is a complex process, particularly if it involves many people working over a relatively long time.

→ Software project management is an essential part of software engineering & most important aspect of software development.

→ Effective project management helps to ensure.



(i) software will meet worker/customer expectation.

(ii) software will be delivered within target & time constraints

* Project Management :-

→ Project Management is the application of processes, methods, skills, knowledge & experience to achieve specific project objectives within agreed parameters.

→ Project Management has to find deliverables that are constrained by a finite timeline & budget.

→ Project Management includes →

(i) developing a project plan - which involves defining & conforming the project goals & objectives.

(ii) How those goals & objectives will be achieved.

What is Project Management :-

- Project Management is a way to lead the team and use all the resources in a manage way to achieve the requirement Results which we want at the end of the project.
- The focus of the project Management is to achieve all of the project goals under the given constraints.
- constraints are given time frame, budget, Requirement and all these things.

What is Software Project :-

- A Software project is complete procedure of software development from the Requirement gathering to final stage of developing a Software and maintain it.
- It is also important to keep the Software project under the define constraints.

* Software Project Management :-

- Software Project Management is a well defined, disciplined field, and an using which we plan, Manage and supervise the Software Project.
- Software Project Management field mainly deal with three Step./things
 - (1) Time
 - (2) Cost
 - (3) Quality

W5HH Principle of Project Plan.

"Boehm" 1996 suggested that answering seven simple questions we can define the project objectives, schedule and milestones, responsibilities, resources required and technical approach taken. This leads to a systematic project plan.

- ① **WHY** the system is developed?
- ② **What** will be done?
- ③ **When** it will be completed?
- ④ **Who** has the responsibility for each particular task?
- ⑤ **Where** are the persons located in team organisation?
- ⑥ **How** will the job be done?
- ⑦ **How** much of each resource is needed?

Phases of Project Management! →

There are four phases of project management.

- (i) Project planning
- (ii) Monitoring
- (iii) Control through scheduling
- (iv) Terminal Analysis.

① Project Planning!

The first and foremost component of project management is effective planning. Lack of planning is the primary cause of failures such as a poor quality, cost overrun, late delivery and high maintenance cost for software. Before the project begins the manager and software team must estimate the work to be done, resources that will be required and time that will elapse from start

to finish, is known as planning.

Objective of the project planning:-

The project planning is aimed to achieve following goals:-

- i) The estimate of resources are made and documented.
- ii) Cost estimation.
- iii) Time scheduling and the commitments made acc. to schedule.

Planning Activities for development process:-

The planning activity for the development process can be broadly grouped into four group, to be followed in sequential step as follows:-

- ① Defining Problem / Scope of Software
 - Desc. of present situation
 - Goals and scope of new system
 - meeting who is going to work
 - Cost-benefit.
- ② Solution strategies and feasibility
 - Technical
 - Resource feasibility
 - Financial
 - Time
- ③ Resource Allocation
 - Human Resources
 - People in organisation
 - their organisational position (mgr, slw)
 - Skill level & their speciality (DB/NW/APP)
 - Reusable slw
 - past project already running in mkt, Reuse module
 - Environmental
 - Test data used by past experienced
- ④ Cost Estimation.
 - HW, OS, tools, power line equipments and some machine tools.
 - Make estimate based on similar past project by exp
 - divide the whole project into small pieces, cost easily be counted by LOC
 - use one of the empirical model.

② Monitoring:- monitoring covers all the activity to track and review the slw accomplishment and an insidability of the progress of project. Monitoring suggests weather the objective are being met or not and if not the necessary action are taken by "control" activity of next phase. Milestones are review points in development phase of life cycle of slw. Where the managers and team member check the documents. work. Product and their quality to predict final software status in terms of quality & to predict final software cost and complete time scheduled.

The activity in monitoring includes:

- i) Knowing the s/w product size (Loc Based and PP Based) and if diff. from planned it should be controlled or the (SRS) be changed.
- ii) Resource utilization is monitored and the reallocation is done if under utilized.
- iii) Technical aspects and time schedules are reviewed from time to time.
- iv) Re-planning
- v) Commitments are changed and documented if necessary and communicated / discussed with top mgmt to take the "Control Action".
- vi) Approved changes are finally communicated to s/w engineering team members.

③ Control Through Scheduling:-

Control phase includes all the corrective measures taken during development phase, if the cost, schedule and quality objectives are not met as "monitored".

The corrective action could be:

- a) Adding manpower
- b) Requirement scrubbing
- c) Focusing incrementality in order of importance only.
- d) Using better solution for development.
- e) Using chart GANTT and PERT.

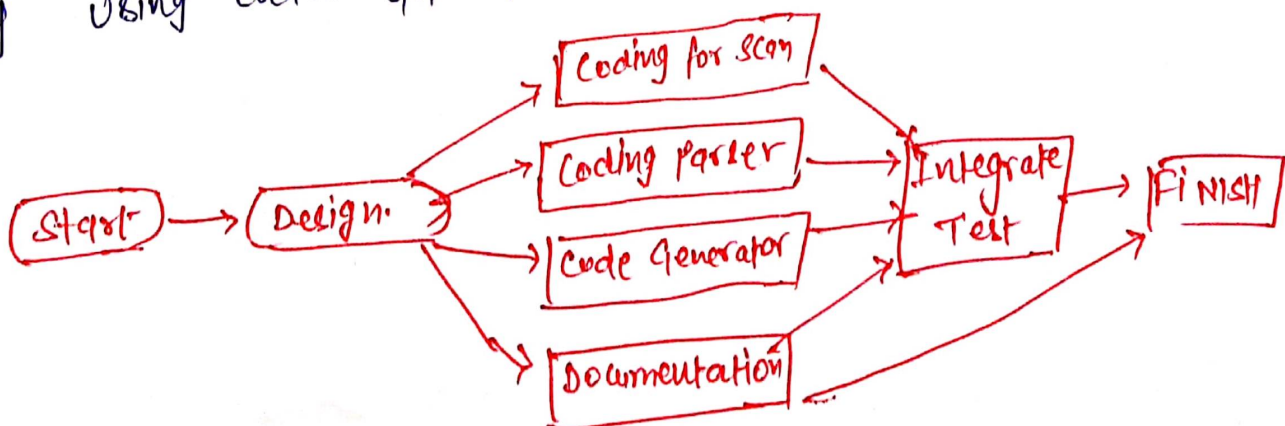


Fig. PERT chart for Compiler s/w.

④ Terminal Analysis.

Terminal analysis is performed at the end of the development process. This helps in predictions and estimations for future projects. The analyst in this phase prepares data in quantitative and qualitative form of the project just developed.

UNIT- IV

Software planning and Project Mgmt

Management in simple words is defined as the art of getting work done through other people with efficiency and effectiveness. Management is a necessary activity when computer based systems are built and developed.

Software managers plan and schedule the software projects to devise economical and timely software solution. Project management has four types of activities broadly:

- ① Planning
- ② Monitoring
- ③ Controlling
- ④ Terminal Analysis.

mgmt is not an exact science. The situations varies according to project characteristics. The project manager has functions like writing project proposals, cost estimations, staffing, scheduling, resource allocation, software configuration mgmt, interaction with clients.

Mgmt levels:-

The mgmt of the software can be done at four levels.

1. A good manager should pay attention to all these levels:
4 p's.

- a) People
- ☒ b) Product
- ☒ c) Process
- d) Project

People: A manager should take care of the people associated with software team, because they are the pillars for development. Proper recruiting, selection, training, job satisfaction, morale boosting team culture development should be done. This factor is most important becoz we know software is a labour intensive activity. So people must be organised into effective, motivated and coordinated teams.

Product: Product is any software that is built on the request of the others. Before the project can be planned the customer and developer must meet to define product objectives and its scope or coverage. The alternative solution and their technical constraints should be identified by the managers, i.e. the overall goals.

Process: A manager should decide what the processes in software development are to be followed. What are the tasks, milestones and quality assurance points. The framework and umbrella activity are decided.

Project: To control the complexity of software and IT development process, the project as a whole should be managed. The cost and time limits should be planned, followed and managed to achieve the success.

Planning Activities for Development Process

The Planning activities for the Development Process can be broadly groups into four steps they are :-

- (i) Defining Problem / Scope of Software
- (ii) Solution Strategies and feasibility.
- (iii) Resource allocation :-
 - Human Resources
 - Reusable Software
 - Environmental
- (iv) Cost Estimation.

Defining Problem / Software Scope.

The first step in planning a project is to define and documented, the assumptions, goals and constraints of the project. A clear statement of goals of project are necessary to guide engineer in daily decisions. A definitive problem statement should include :

- A description of present situation.
- Goals and scope of the new system.

The most common used technique for knowing the description of present situation, is to conduct a preliminary meeting or interview of customer by developers. The context-free

questions should be asked, such as:-

- Who has requested the work?
- Who is going to use it?
- What is the cost-benefit of it?

Solutions Strategies and Feasibility

After various solutions have been enumerated, they should be checked against constraints. Software feasibility is usually checked against four solid facts:-

- (i) Technical:
- (ii) Resource Feasibility:-
- (iii) Financial:
- (iv) Time:

To check these feasibilities, even a small prototype can be built.

Resources Allocation

The next task is to estimate the resources needed to perform the software development efforts. Each resource is defined in tabular form as description, availability schedule, time requirement. There are generally three forms of resources for Software Develop

3

(i) Human Resources :- Software being labour-intensive, we must know the

- People in organisation
- their organisational position (Manager, Software, etc.)
- Skill level.
- their specialty

(ii) Reusable Software Resources :- The past projects already running in organisation, known as Legacy software can be reused as a part of new projects. Test data used by past experienced teams could also be used.

(iii) Environmental Resources :- The hardware, operating system, the tools and even the office, power line equipments or some machine tools can also be listed.

Monitoring

Monitoring cover all the activities to track and review the software accomplishment and an inside visibility of the progress of projects. Monitoring suggests whether the objectives are being met or not and if not, the necessary actions are taken

by 'Control' activity of next phase. The activities in monitoring includes :-

- (i) Knowing the software product size and if different from planned it should be controlled or the requirements be changed.
- (ii) Resources utilization is monitored and the reallocation is done if under utilized.
- (iii) The technical aspects and time schedules are reviewed from time to time.
- (iv) The re-planning is documented as per measured performance.
- (v) Commitments are changed and documented if necessary and communicated / discussed with top management to take the "Control" action.
- (vi) Approved changes are finally communicated to the software engineering team members.

Cost Estimations

The cost estimations draws the boundaries for all resources to be acquired. There are three methods of estimating a reliable cost of projects:

(4)

- (i) Make estimates based on similar past projects by experience.
- (ii) Divide the whole projects into small pieces, whose cost could be easily estimated by counting LOC or functionality etc
- (iii) Use one of the empirical Models

For totally new projects second case i.e., dividing or decomposing is a stepwise simple method for estimating cost of complex projects.

Control Through Scheduling

Control phase includes all the corrective measures taken during development phase, if the cost, schedule and quality objectives are not met as "Monitored".

The corrective actions should be :-

- (i) Adding Manpower
- (ii) Requirement Scrubbing
- (iii) Focusing incrementality in order of importance only, during the spiral model life cycle, also cuts down the cost, schedule and efforts.

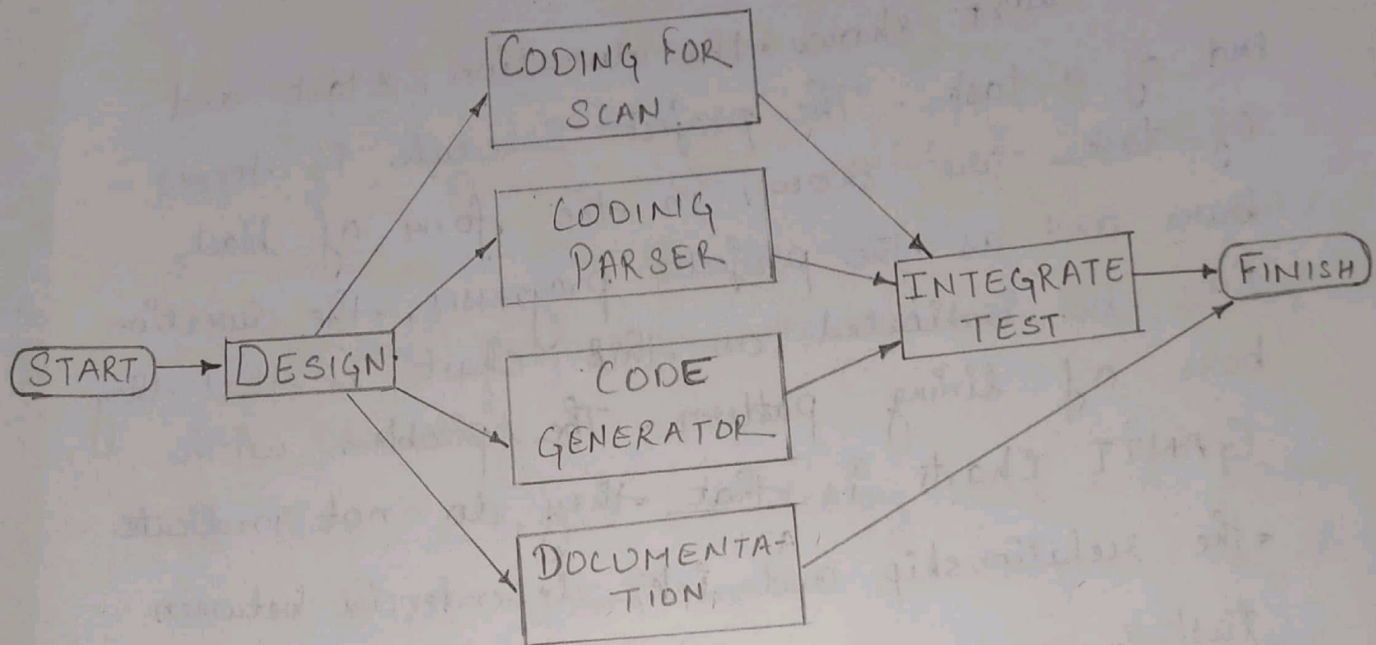
- (iv) Using better tools for development
- (v) Using charts like GANTT and PERT charts for monitoring and marking the boxes and arrows on completion.

TIME \ TASKS	JAN	FEB	MAR	APR
ALGORITHMS				
FLOWCHART				
CODING				
TESTING				

GANTT CHART used for scheduling

TIME \ TASKS	JAN	FEB	MAR	APR
ALGORITHMS				
FLOWCHART				
CODING				
TESTING				

GANTT Chart for monitoring and control.



PERT chart for compiler software

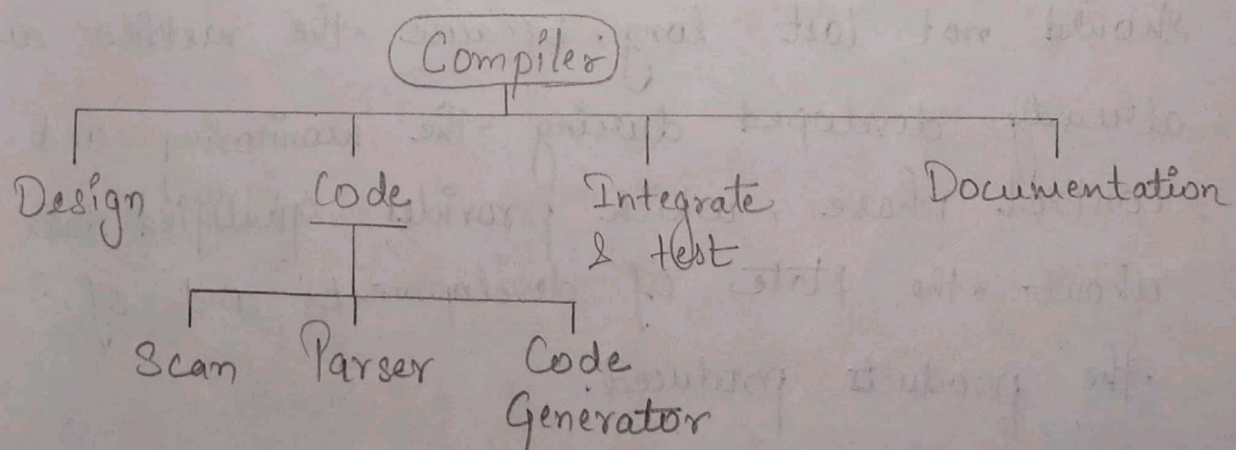
Terminal Analysis

Terminal Analysis is performed at the end of development process. This helps in predictions and estimations for future projects. But this phase should not last long, because the metrics are already developed during the monitoring and control phase, which provides qualified data about the state of development and of the products produced.

5

GANTT chart shows the duration, start and end of a task. The project schedule in terms of task are shown in the form of black bars and as the project progresses, the duration can be indicated on this chart is that they bars of lining pattern. The problem with GANTT charts is that they do not indicate the relationship and inter dependencies between tasks.

PERT and CPM are another project scheduling methods, which take care of interdependencies of tasks. A work Breakdown structure (WBS) shown and can be broken into the activities of PERT chart.



Tasks broken into leafs of fine details

Personal Software Process (PSP)

- PSP is a private metric, which is measured by an individual himself so that he can improve his personal performance to measure his quality and productivity.
- The PSP is based on ideology that each person is different from another and everyone has its own methods which brings out the best out of him. These methods are best known to him only.
- Public metrics on the other hand are known to all the team members and are measures of collective team efforts or characteristics.

Principles of Designing A Good Metric

- For designing a good metrics, a goal/questions/metrics approach is taken.
- But there are few principles, which should be kept in mind while designing a metrics:—

- (i) Software Metrics should be used to analyse software qualities and not to evaluate people.
- (ii) Metrics should neither be used for appraising the individuals nor to threaten individual or team.
- (iii) If it is so, the designer will try to maximise the measured quantity rather than concentrating on the quality.
- (iv) The quality evaluation should be applied to both products as well as the process, because process causes the products.
- (v) Quality measures should be applied to all the intermediate steps, products and not only to the end products.
- (vi) Set the goals clearly by working with experienced teams, because metrics must be defined on well-thought-out goal.

Qualities Of Good Metric

- (i) Simplicity and Precision
- (ii) Validity
- (iii) Objective queries
- (iv) Robustness and Consistency

Types Of Metrics

There are many ways in which the metrics could be classified, but for this text, the metrics broadly can be categorised as:—

(a) Size / Quantity / Productivity based metrics

- Text based
- LOC based
- Function Point based.
- Token Count.

(b) Design based metrics.

- (i) data structure
- (ii) Information flow Information flow metric.
- (iii) Object Oriented design.

(c). Quality based metrics :

- Defects metrics
- Maintainability metrics
- Usability metrics
- Reliability metrics.

Text Based Metrics.

- This is the simplest but not of much use, which measures the productivity.
- The text used in programming, irrespective of its usefulness, is the text based metrics. The size in this may be the numbers of bytes a program occupies on disk, no. of pages etc.
- It can be imagined that a software engineer who uses heavy comment lines or spaces in between may be ~~adjudged~~ adjudged as productive one.

LOC Based Metric (Line of Code)

This is actually the size based metrics in true sense. The quality of software delivered is adjudged by the size of the software.

A table of following form could be used to maintain the record.

Projects	LOC	Efforts	Errors	Costs	People
Inventory	12,021	63	183	90,000	6
Payroll	3,027	21	132	24,000	3
Financial	15,510	124	363	96,000	12
Accounting					

This is very simple to use and so it is popular but this is also not accepted as the best way at the same time. In Line of Code (LOC) the source instructions are counted

and the excluded lines are: —

- Comments Lines
- Header Lines
- The line of an "included" file which is used more than once.

IN
B
N
S
I

— Drivers or support software is not included.

Two most common code size based metrics are :—

- (a). Delivered Source Instructions (DSI)
- (b). Non-Commented Source Statement (NCSS)

Advantages of LOC Size Metric:

1. Easy of measuring it.

Using this metric is better than using nothing at all.

Large Literature is available for estimating models that use KDSI as input such as:—

- Errors per KDSI
- Rupees per KDSI
- LOC per person-month, etc.

* Lines of Code (LOC)

- Basic idea is to estimate size.
- NO general agreement about what is line of code.
- Some include data declarations, comments or ~~code~~ unexecutable statements others exclude them.

→ Productivity = $LOC / (\text{Person} - \text{Month})$

→ Measuring by line of code is like measuring a building by no. of bricks used.

```
#include <stdio.h>
```

```
void main() {
```

```
int a = 5;
```

```
cout << a;
```

```
a = a + 1;
```

```
cout << a;
```

```
}
```

} // header
// comments
// lines
// comment
// lines

- Lines of code can predict programming time, but fail to tell
- anything about efficiency, reliability

* What is LOC?

- Declaration, Actual including logic and computation.

* What is NOT a LOC?

- BLANK lines

↳ Indirectly to readability of code

- COMMENTS

↳ Indirectly to help in code understanding as well as during maintenance

↳ Not a LOC because

① Do not contribute to any kind of functionality.

② Measured by developers to give false notion about productivity.

→ Advantage by using LOC for size estimation

↳ Very easy to count and calculate from the developed code.

→ Disadvantage of using LOC for size estimation,

b/c LOC is language + technology dependent

(ii) What constitutes an LOC

↓ varies from orgⁿ to orgⁿ

Program 1

```
for (int i = 1; i < 10; i++)  
{  
    cout << i;  
}
```

program
(2)

int a ; // declaration
Code comment

// declaration } comment line

int a; {code}

(1)

- ① In using LOC for size estimation,
- ② way easy to count & calculate from developed code.

② Disadvantages of LOC :-

- language dependent
- organization to organization

9

10

Drawbacks of LOC Size Metric:-

- (i) LOC gives a numerical value which characterises only one specific view.
- (ii) The metric is language dependent and not application dependent.
- (iii) The metric has no relation to the quality and efficiency.
- (iv) There is no standard, some project managers even count comment lines considering the fact that documentation enhances the maintainability, readability and is a way of structured programming.
- (v) Can be measured only after code has been fully developed.

Functions Point (FP) Based Metric

- It characterises the complexity of software and so FP can be used to forecast how long and how many people will be required to develop it.
- FP based metrics assumes that there are five items that indicate the complexity of an application and weighted sum of these items is what we call Functions Point of that software.
- The five items are:—
 - (i) Number of external Inputs (I)
 - (ii) Number of external Outputs (O)
 - (iii) Number of Users Inquiries (Q)
 - (iv) Number of Files (F)
 - (v) Number of External Interface (E)
- Input refers to data provided to the software.
- Output is the reports, screens and error message.

- Inquiry is the on-line (small) interactive input which produces immediate (on-line) output.
- Each logical files that may be a part of database is counted.
- Interface means the data sent to and from another system.

After Counting all these values weights are combined with these numbers as in the tables :-

wt		Weight				
		LOW	AVERAGE	Complex		
Input	$I \times 2$	3	4	6	=	
Output	$O \times 2$	4	5	7	=	
Query	$Q \times 2$	3	4	6	=	
File	$F \times 2$	7	10	15	=	
External	$EI \times 2$	5	7	10	=	

Total Count = $\Sigma 2$

Alan Albrecht, who developed this metric, names this total count as UFP (Unadjusted Function Point). But the final Function Point (FP) is arrived at by considering answers

to 14 questions, which gives Complexity Adjustment Factor (CAF)

$$FP = \text{Total Count} * [0.65 + \sum Fi / 100]$$

where $\sum Fi$ is the sum total of 14 response having range of values as :-

0 - Not Applicable / Important

1 - Insignificant

2 - Moderate

3 - Average

4 - Significant

5 - Absolutely Essential.

Once the FP is calculated, the metric can be used in many ways:

- FP / PM $\rightarrow$ function point / Person-month
- Rupees spend / FP.
- Error rate / FP.

(11)

Advantages Of FP :-

1. It is not language dependent.
2. FP gives size measures but independent of Code size.
3. FP have gained widespread acceptance for data processing applications.
4. FP metric is not based on internal structure of software so it can be computed at the time of preliminary design in advance :-

Disadvantages Of FP :-

1. NOT Consistent,
2. Difficult to compute as complicated processing.
3. The behavioural aspects which is major in engineering and embedded system, is ignored.
4. CAF is sometimes no good, as it fails to give useful values.

Efforts

Another quantity given by Halstead is efforts 'E'

$$E = V/L$$

where E is a number of mental discriminations required to implement program.

Ex:- Consider the C program, Calculate its length and volume.

Sol.

```
main ()
{
    int a,b,c;
    scanf ("%d %d", &a, &b);
    c = a;
    a = b;
    b = c;
    printf ("Swapped values\n%d %d", a, b);
}
```

Heuristic / Algorithmic Model (COCOMO)

- Constructive Cost Estimation Model (COCOMO) is an algorithmic / heuristic model suggested by Boehm (1981)
- The heuristic models have strong mathematical base and its parameters are modelled using mathematical expressions.
- Boehm in his classic book on Software Engineering (1981) introduced 'COCOMO', which instantaneously became one of the most widely used estimation model.
- Later on it was modified in 1996 to COCOMO-II.
- COCOMO starts with an initial estimate of development efforts from single variable from Kilo Deliverived Lines of Code (KDLOC)
- Determines 15 multiplying factors from desired attributes of the software.

12

The unique operations

$\eta_1 = \text{main}(), \{ \}, \text{int}, \&\text{conf}, \&, ", "; ", ", =, \text{printf}.$

Unique Operands

$\eta_2 = a, b, c, \&a, \&b, "/.d /.d ", "swapped values "/.d /.d "$

$\therefore \eta_1 = 10, \quad \eta_2 = 7.$

$$\therefore \text{Estimated length} = 10 \left(\frac{\log_{10}^{10}}{\log_{10}^2} \right) + 7 \times \left(\frac{\log_{10}^7}{\log_{10}^2} \right)$$

$$= \frac{10 \times 1}{0.3010} + 7 \left(\frac{0.845}{0.3010} \right) = \underline{\underline{52.876}} \text{ Ans.}$$

$$\text{Volume} = \text{Length} * \log_2 (10+7)$$

$$= 52.876 \times \left(\frac{\log_{10}^{17}}{\log_{10}^2} \right)$$

$$= 52.876 \times 4.0878$$

$$\approx \underline{\underline{216.}} \text{ Ans}$$

- Modify the initial estimate by multiplying these factors.
- Compare this estimate with that given by Delphi or other technique and rectify if variable variations.

Boehm broadly classified all the software products into three categories accordingly to their complexity:-

- (a) Organic
- (b) Semi-detached
- (c) Embedded.
 - Basic models
 - Intermediate
 - Detailed Submodel.

(1) Basic Model :- The simplest and rough estimate which can be done by a calculator, is basic model.

... .. linking is blocked in⁽⁴⁾

→ The equations are as follows:—

$$\text{Efforts, } E_i = a (K D L O C)^b$$

$$\text{Development Time, } \Rightarrow T_D = c (E_i)^d$$

Type of s/w	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

(ii) Intermediate Model:—

COCOMO ~~models~~ uses 15 different attributes which scale and multiply the nominal development efforts. These attributes are called cost driver attribute and determine the effort multipliers as shown. The cost drivers are divided into four groups.

→ The development time versus the product size in KLOC is plotted in fig. from fig it can be observed that the development time is sub linear function of the size of the product

* Example :- A 60 KLOC program can be development in approximately 18 months, regardless of it is of organic, semi detached, or embedded type.

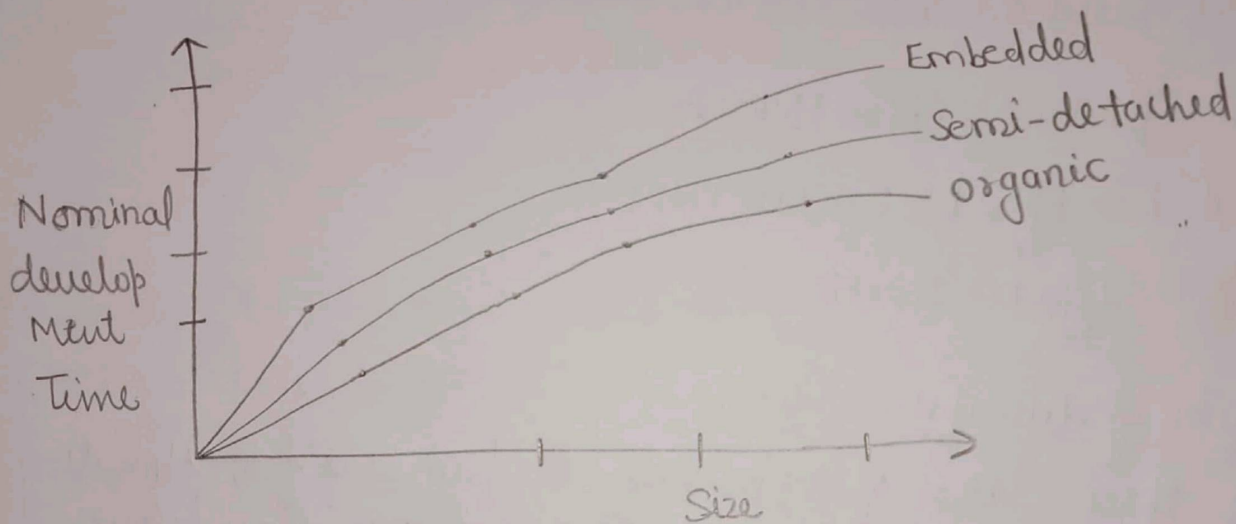


Fig:- Development time versus size

Example :- 1 Suppose a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three Model i.e., organic, semi-detached & embedded.

Solution :- The basic CoCoMo equation takes the form:

$$\text{Effort} = a_1 * (\text{KLOC})^{a_2} \text{ PM}$$

$$T_{dev} = b_1 * (\text{efforts})^{b_2} \text{ Months}$$

$$\text{Estimated Size of project} = 400 \text{ KLOC}$$

(3)

* T_{dev} :- T_{dev} is the estimated time to develop the S/W, expressed in months.

* Effort :- Effort is the total effort required to develop the S/W product expressed in person months (PMs).

* Estimation of development effort :-

For the three classes of S/W products, the formulas for estimating the effort based on code size are shown below :-

Organic :- $\text{Effort} = 2.4 (\text{KLOC})^{1.05} \text{ PM}$

Semi-detached :- $\text{Effort} = 3.0 (\text{KLOC})^{1.12} \text{ PM}$

Embedded :- $\text{Effort} = 3.6 (\text{KLOC})^{1.20} \text{ PM}$

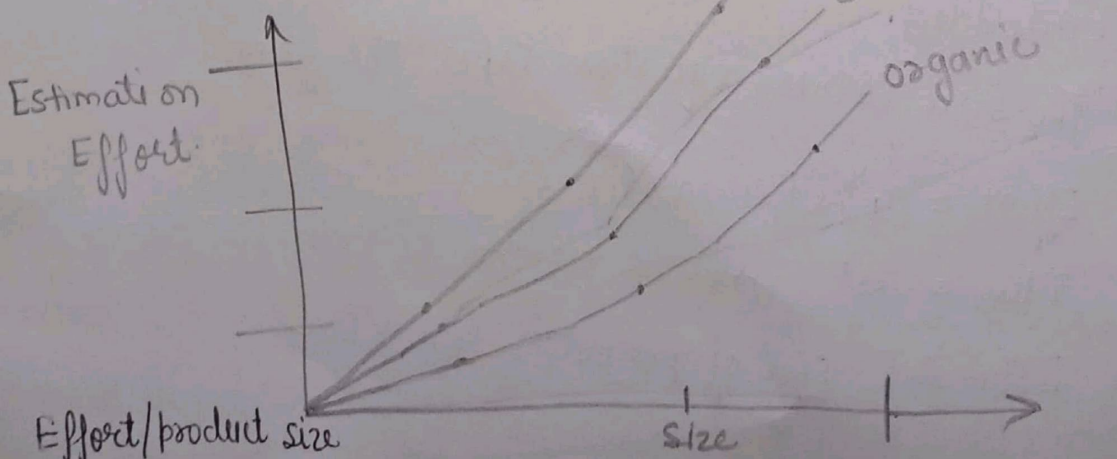
* Estimation of development Time :-

For the three classes of S/W products, the formulas for estimating the development time based on the effort are given below :-

Organic :- $T_{\text{dev}} = 2.5 (\text{Effort})^{0.38} \text{ Months}$

Semi-detached :- $T_{\text{dev}} = 2.5 (\text{Effort})^{0.35} \text{ Months}$

Embedded :- $T_{\text{dev}} = 2.5 (\text{Effort})^{0.32} \text{ Months}$



→ The project classified as semi-detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and

→ examples:- Developing a new operating System (OS), a data base Management System (DBMS).

* Embedded:- A development Model project is treated to be of an embedded type, if the software being developed is strongly coupled to complex h/w. or if the strength regulation on the operational method

examples:- ATM, AIR Traffic control.

According to Boehm, S/W cost Estimation should be done through three stages:-

- 1) Basic Model
- 2) Intermediate Model
- 3) Detailed Model

*1. Basic CoCo Model :- The basic CoCo Model provides an accurate size of the project parameters.

The following expression give the basic CoCo Model:-

$$\text{effort} = a_1 * (\text{KLOC}) a_2 \text{ PM}$$

$$T_{dev} = b_1 * (\text{efforts}) b_2 \text{ Months}$$

where KLOC is the estimated size of the S/W product indicate in kilo lines of code.

a_1, a_2, b_1, b_2 are constraints for each group of S/W products.

* CoCoMo Model:-

①

- CoCoMo (Constructive Cost Model) is a regression Model based on LOC, number of line of code.
- It is a procedural cost estimate Model for software project and often used as a process of reliability.
- The key parameters which define the quality of any software products, which are also an outcome of CoCoMo are primarily Effort & Schedule.

→ Effort :- Amount of labor that will be required to complete a task. It is measured in person months units.

* Schedule :- It is measured in the units of time such as weeks, months.

* Boehm's definition of organic, semidetached, and embedded System :-

→ Organic :- A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved.

→ Semi-detached :- A software project is said to be a semi-detached type if the vital characteristics such as team size, experience type if the vital characteristics knowledge of the various programming environment lies in between that of organic and embedded.

(6)

* Intermediate Model:- The basic Model CoCoModel considers that the effort is only a function of the number of lines of code and some constants calculated according to the various SW system.

The basic CoCoModel by using a set of 15 cost drivers based on various attributes of SW Engg.

* Classification of cost drivers and their Attributes:-

(i) Product attributes:-

- Requirement SW reliability extent.
- Size of the application database.
- The complexity of the product.

Hardware Requirement:-

- Run time performance constraints
- Memory constraints.
- Requirement turnabout time

* Personal Attributes:-

- Analyst Capability
- SW engineering capability

* Project Attributes:-

- use of SW tools
- Application of SW engineering Methods
- Required development schedule.

5

(1) Organic Mode :-

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

$$D = 2.5 * (1295.31) 0.38 = 38.07 \text{ PM}$$

(2) Semidetached Mode :-

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

$$D = 2.5 * (2462.79) 0.35 = 38.45 \text{ PM}$$

(3) Embedded Mode :-

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

$$D = 2.5 * (4772.8) 0.32 = 38 \text{ PM}$$

* Q 2 :- Example 2 :- A project size of 200 KLOC is to be developed. Software development team has average

experience on similar type of projects. The project schedule is not very tight. Calculate the Effort, development time, average staff size, and Productivity of the project.

Solution :- The Semidetached Mode is the most appropriate mode, keeping in view the size, schedule and experience of development time.

Hence

$$E = 3.0 (200) 1.12 = 1133.12 \text{ PM}$$
$$D = 2.5 (1133.12) 0.35 = 29.3 \text{ PM}$$

$$\text{Average Staff Size (SS)} = \frac{E}{D} \text{ Persons}$$
$$= \frac{1133.12}{29.3} = 38.67 \text{ Persons}$$

$$\text{Productivity} = \frac{\text{KLOC}}{E} = \frac{200}{1133.12} = 0.1765 \text{ KLOC/PM}$$

Multiplier	Range of Values
Product attributes	
Required Reliability	0.75 to 1.40
Database size	0.94 to 1.16
Product complexity	0.70 to 1.65
Compute attributes	
Execution time constraint	1.00 to 1.66
Main storage constraint	1.00 to 1.56
Virtual machine volatility	1.29 to 0.82
Computer turnaround time.	0.87 to 1.15
Personnel attributes	
Analyst Capability	1.46 to 0.71
Programmer Capability	1.42 to 0.70
Applications experience	1.29 to 0.82
Virtual machine experience	1.21 to 0.90
Programming language experience	1.14 to 0.95
Project attributes	
Use of modern Programming practices	1.24 to 0.82
Uses of software tools	1.24 to 0.82
Required development schedule	1.23 to 1.10

The multiplied results of all the 15 factors give the Effort Adjustment Factors (EAF).
And final Effort is calculated as

$$E_f = EAF \times E_i$$

(iii) Detailed Model:-

The detailed Model goes further by introducing two more factors:-

- (a). Phase sensitive effort multipliers.
- (b) Three levels (Module, subsystem, system level).

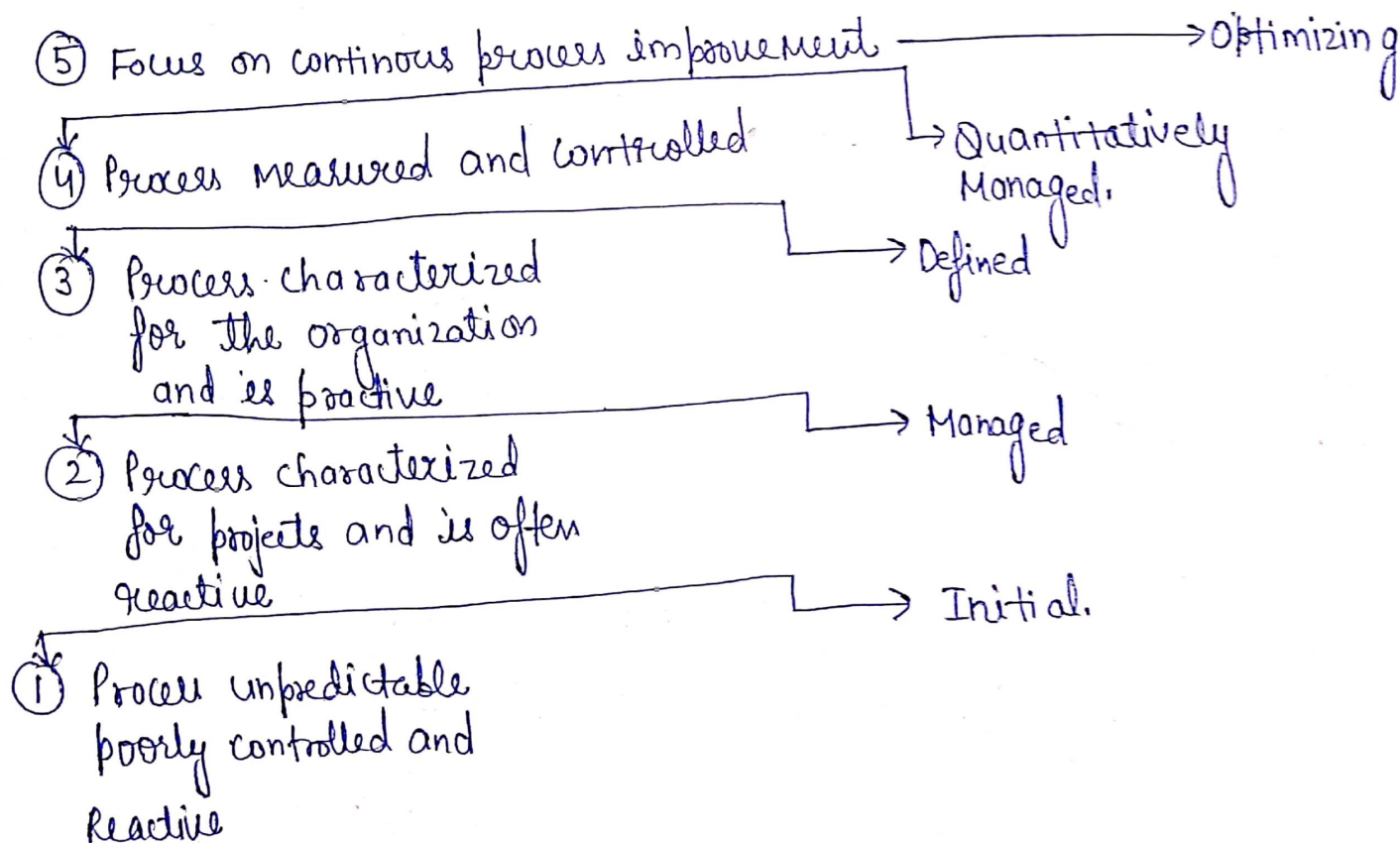
CMM (Capability Maturity Model)

①

* A Capability Maturity Model (CMM) is a Reference Model of Mature practices in a specified discipline, used to perform improve and appraise a group's capability to perform that discipline". SEI.

→ CMM contains a list of key process areas that includes Software engineering, project management, and process improvement divided into different levels.

CMMI Maturity levels :-



→ Overview of SEI-CMM :-

(2)

Software Engineering Institution ⇒ developed Capability Maturity

Helped organization to improve the quality of SW they developed.

* Adoption of SEI CMM has significant business benefits.

* 2 ways.

Capability evaluation

SW process assessment

↓
Provides a way to assess the SW process capability.

↓
It is used by an organization with the objective to improve the process capability.

* Level 1 :- Initial :-

→ A software development organization this level is characterized by adhoc activities.

* Level 2 :- Repeatable :-

→ At this level, the basic project management practice such as tracking cost & schedule are established.

* Level 3 :- Defined :- At this level processes for both management & development activities are changed defined & documented.