Project Planning: Step wise Project Planning

- Project planning is a crucial part of project management focused on creating a detailed plan that outlines the steps and resources necessary to achieve the project's objectives, including identifying the project's scope, establishing a timeline, assigning tasks and resources, and budgeting for the project.

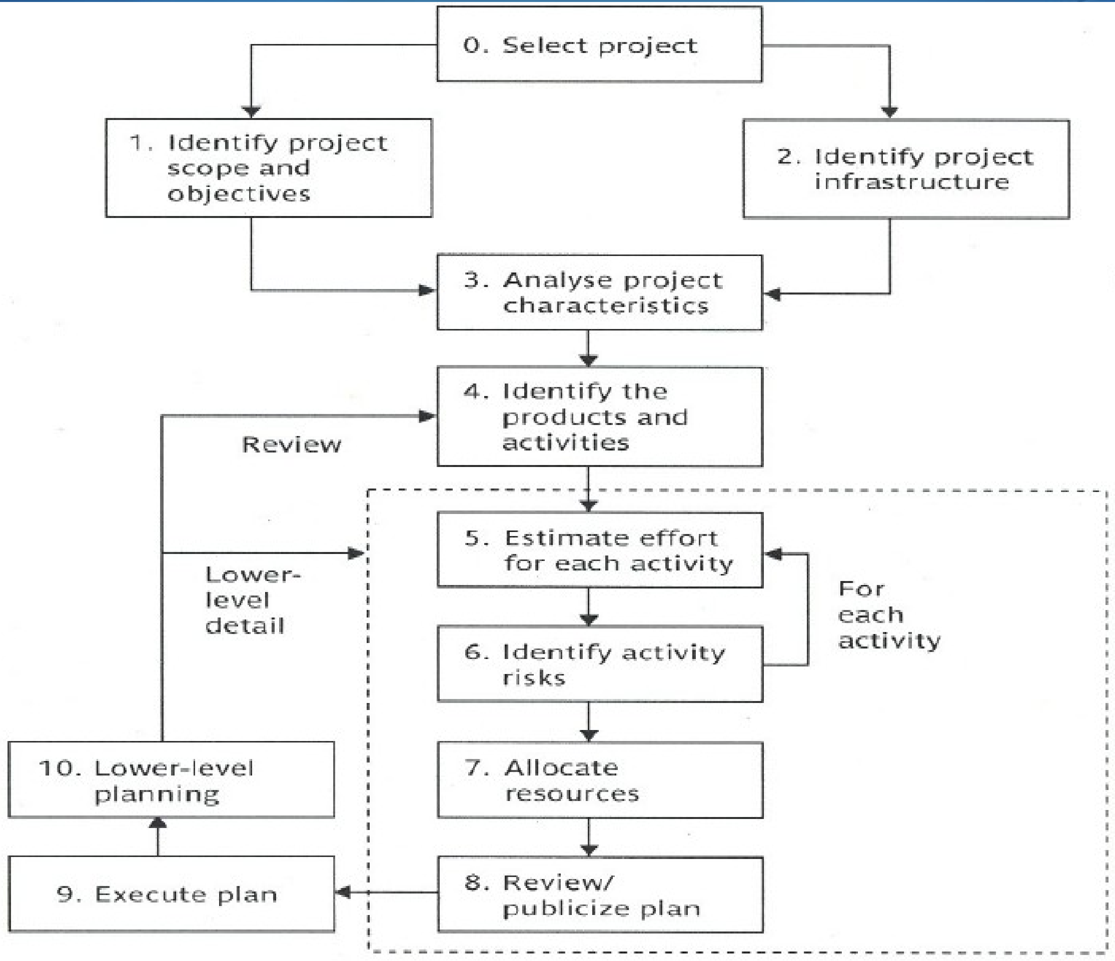
Planning Activities Steps

- Step 0: Select project
- Step 1: Identify project scope and objectives
- Step 2: Identify project infrastructure
- Step 3: Analyze project characteristics
- Step 4: Identify project products and activities
- Step 5: Estimate effort for each activity

Planning Activities Steps (cont.)

- Step 6: Identify activity risks
- Step 7: Allocate resources
- Step 8: Review/publicize plan
- Step 9: Execute plan
- Step 10: Execute lower levels of planning

Project Planning: Step wise Project Planning


0. Select project
1. Identify project scope and objectives
2. Identify project infrastructure
3. Analyse project characteristics
4. Identify the products and activities
Review
5. Estimate effort for each activity
Lower-level detail
6. Identify activity risks
For each activity
7. Allocate resources
8. Review/ publicize plan
9. Execute plan
10. Lower-level planning

Planning Activities Steps

- Step 0: Select project

  Some process must decide to initiate this project

Step 1: Identify project scope and objectives

Step 1.1: Identify objectives and practical measures of the effectiveness in meeting those objectives

Step 1.2: Establish a project authority

Step 1.3: Stakeholder analysis – identify all stakeholders in the project and their interests

Step 1.4: Modify objectives in the light of stakeholder analysis

Step 1.5: Establish methods of communication with all parties

Planning Activities Steps

Step 2: Identify project infrastructure

2.1. Identify relationship between the project and strategic planning(Strategic planning is a process in which an organization's leaders define their vision for the future and identify their organization's goals and objectives.)

2.2. Identify installation standards and procedures

2.3. Identify project team organization

Planning Activities Steps

Step 2: Identify project infrastructure

2.2. Identify installation standards and procedures

Any organization that develops software should define their development procedures .

As a minimum, the normal stages in the software life cycle to be carried out should be documented along with the products created at each stage.

2.3. Identify project team organization

Project leaders, especially in the case of large projects, might have some control over the way that their project team is to be organized.

Planning Activities Steps

Step 3: Analyze project characteristics

Step 3.1: Distinguish the project as either objective- or product-driven

Step 3.2: Analyse other project characteristics (including quality-based ones)
For example, is an information system to be developed or a process control system,

Will the system be safety critical, where human life could be threatened by a malfunction?

Step 3.3: Identify high-level project risks

Step 4: Identify Project Products and Activities

- **4.1 Identify and describe project products**
  - Identify all the products related to the project
  - Account for the required activities.

Products can be:

- Deliverables.
- Technical products (e.g. training material, operating instructions).
- Products for management and quality (e.g. planning documents).

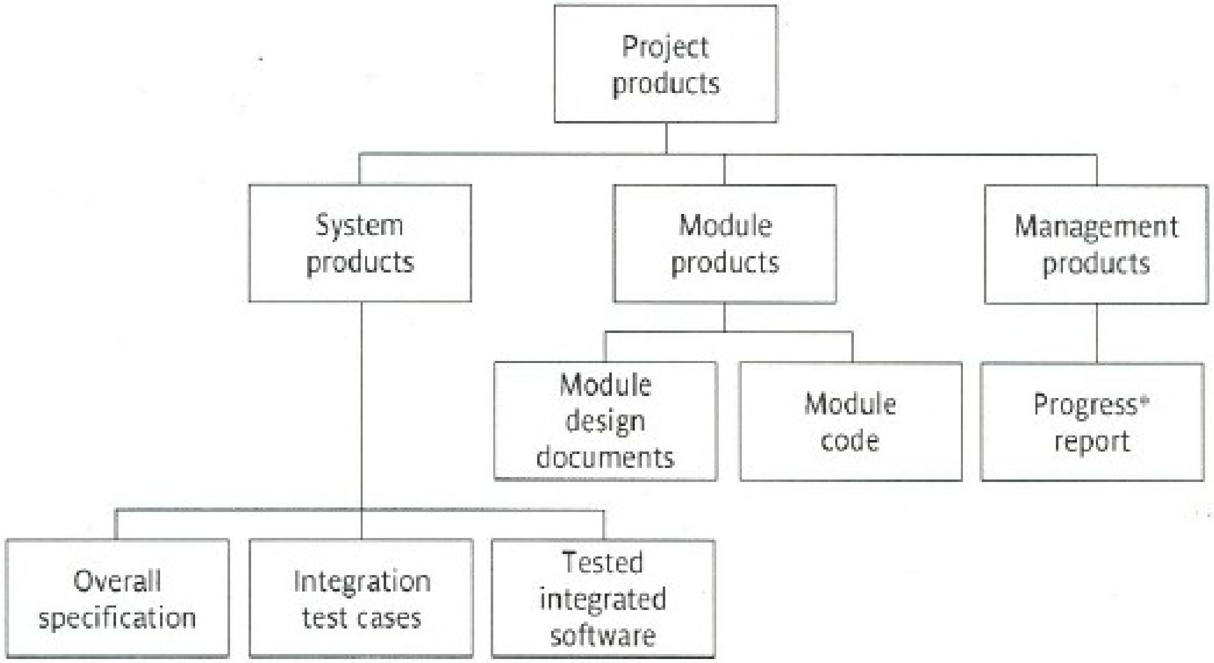
Step 4: (cont.)

- **Activity**
  - training
  - Testing
  - Designing
  - Documenting.

- **Products will perform a hierarchy:**
  - Main products will have a set of components which in turn will have sub-components and so on.

  - These relationships can be documented in a PBS(product breakdown structure.)

Product Breakdown Structure



Step 4: (cont.)

- **4.2 Document generic product flows.**
  - To document the relative order of the products.
  - Some products will need other products to exist first

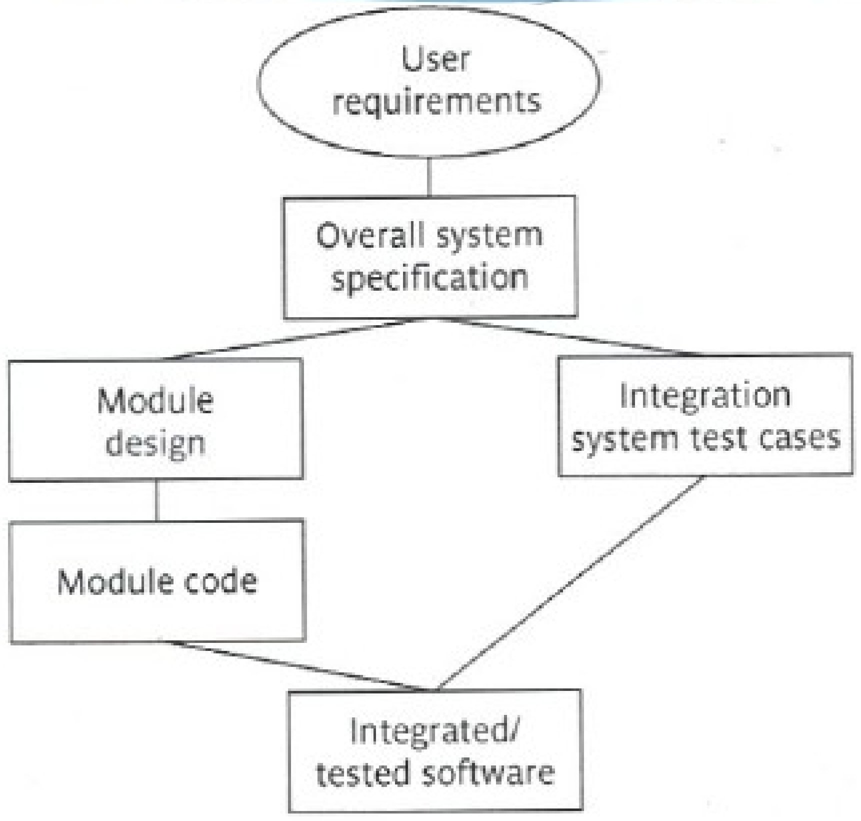
before they can be created.

E.g. the program code to be written need the program

design, the program design needs the program
specification

  - This can be portrayed in a **PFD (**Product Flow Diagram).

Product Flow Diagram


User requirements
Overall system specification
Module design
Integration system test cases
Module code
Integrated/ tested software

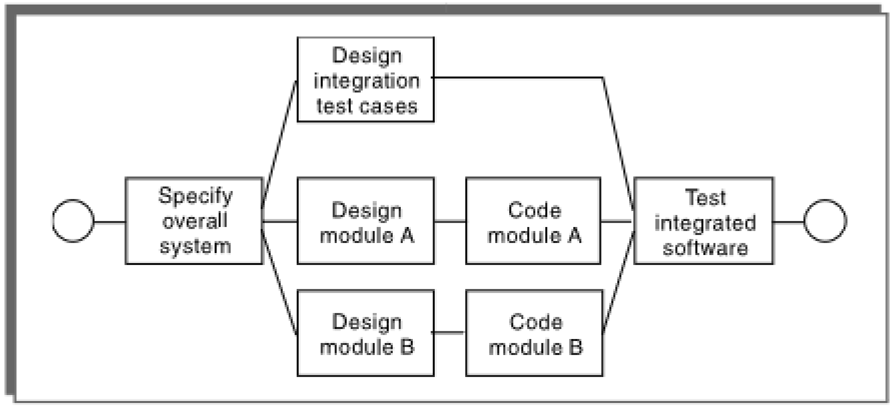
Step 4: (cont.)

- **4.3 Recognize product instances.**
  - We should try to identify each instances of the product

  e.g. there could be only two modules related to the ”module product”

- **4.4 Produce an ideal activity network**
- **Activity network shows :**
  - the tasks that have to be carried out and, their order of execution for the creation of a product.
- Activity networks don’t take account of resource constraints.

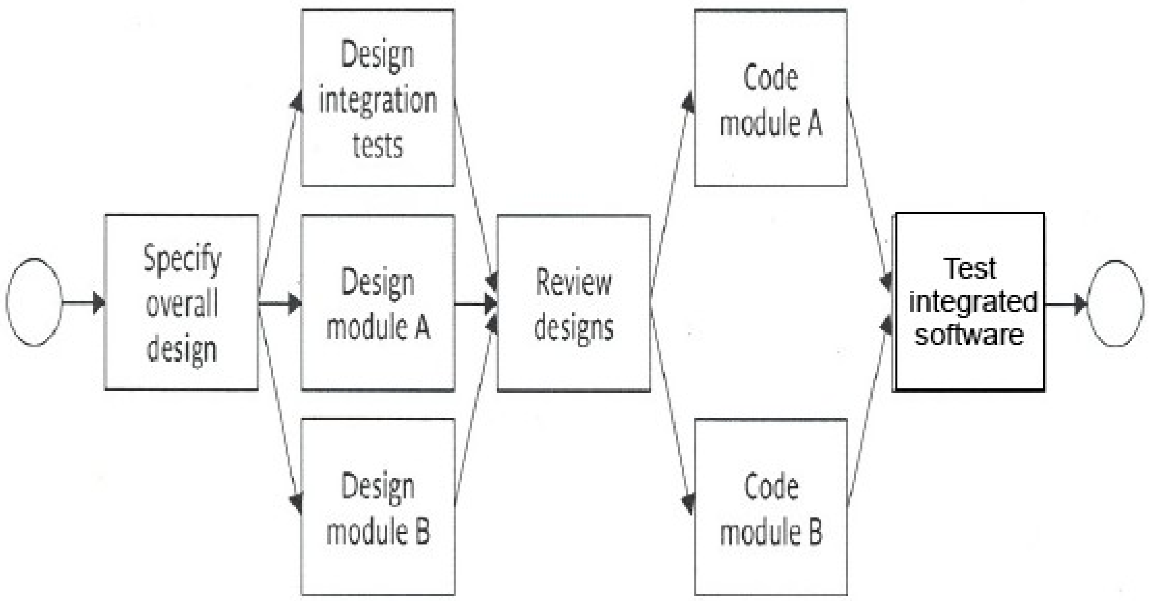
Activity Network Diagram



25.3.5 An example of an activity network

Step 4: (cont.)

- **4.5 Modify the ideal to take into account need for stages and checkpoints. Checkpoints:**
  - To check compatibility of products of previous activities



Step 5: Estimate Effort for Each Activity

- **5.1 Carry out bottom-up estimates.**
  - need to estimate staff effort, time for each activity, and other resources

- **5.2 Revise plan to create controllable activities**.
  - Long activity: break it down.
  - Short activity: combine multiple small activities into one.

Step 6: Identify Activity Risks

- **6.1 Identify and quantify the risks of each activity.**
  - Any plan is based on an assumption if the assumption is not correct this creates a risk to the plan.

For each risk identify:

- Importance, Likelihood and Damage.

Step 6: (cont.)

- **6.2 Plan risk reduction and contingency measures where appropriate**

  For the identified risks:
  - If (they didn’t happen) yet then try to:
    - Avoid risks.
    - Reduce some of them.
  - If a risk materializes (happens) then
    - Use a contingency plan.
- **6.3 Adjust overall plans and estimates to take account of risks.**

  e.g. add new activities that will reduce risks.

Step 7: Allocate Resources (Staffing)

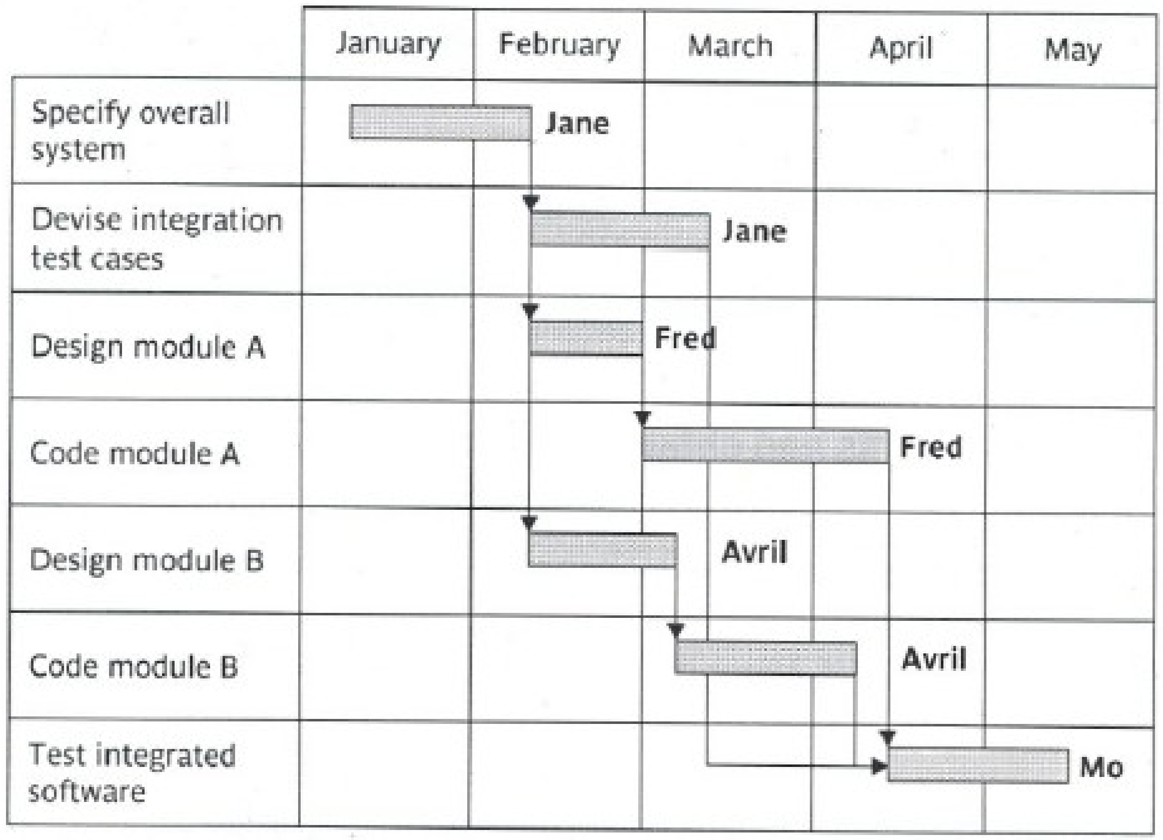
- **7.1 Identify and allocate resources**
  - type of staff needed for each activity.
  - staff availability are identified.
  - staff are provisionally allocated to task.
- **7.2 Revise plans and estimates to take into account resource constraints.**
  - staffing constraints.

Gantt charts: unlike the activity network they give a clear

picture of when activities will take place and highlights which

ones will be executed at the same time.

Gantt Chart



January
February
March
April
May
Specify overall system
Jane
Devise integration test cases
Jane
Design module A
Fred
Code module A
Fred
Design module B
Avril
Code module B
Avril
Test integrated software
Mo

Step 8: Review/publicize Plan

- **8.1 Review quality aspects of the project plan.**

  - Each activity should have quality criteria.
  - Quality criteria : are quality checks that have to be passed

  before the product can be ’signed off’ as completed.

- **8.2 Document plans and obtain agreement.**
  - all parties understand and agree to the commitments in the plan

Step 9/10: Execute plan/Lower-level planning

- Plans will need to be more detailed for each activity as it becomes due

- Detailed planning of later stages needs to be delayed because more information will be available nearer their start.

Click to add Text

- But it is necessary to make provisional plans for more distant tasks.

Software effort estimation

Software effort estimation

It involves predicting the amount of time and effort required to complete a particular task or project.

Effort estimation is a crucial aspect of project management, playing a significant role in setting realistic timelines and allocating resources efficiently.

Some problems with estimating

Nature of software.
-Complexity and invisibility of software.
-Over-estimating small tasks and
-Under-estimating large ones.
-Different objectives of people in an organization
-Managers may wish to reduce estimated costs in order to win support for a-cceptance of a project proposal.

Over- and under-estimating

- An over-estimate is likely to cause project to take longer than it would otherwise
- This can be explained by the application of two laws:
  - **Parkinson's Law:** 'Work expands to fill the time available'
    - Thus, e.g. for an easy task over estimating the duration required to complete it will cause some staff to work less hard to fill the time.
  - **Brook's Law:** putting more people on a late job makes it later
    - If there is an over- estimate of the effort required, this could lead to more staff being allocated than needed and managerial overheads being increased.

Software effort estimation methods

- Expert judgement- just guessing?
- Estimating by Analogy
- Bottom-up approach
- Top down approach

Expert judgment

- One or more experts in both software development and the application domain use their experience to predict software costs..
- Advantages: Relatively cheap estimation method. Can be accurate if experts have direct experience of similar systems
- Disadvantages: Very inaccurate if there are no experts!

Estimation by Analogy

- It is also called case-based reasoning.
- For a new project the estimator identifies the previous completed projects that have similar characteristics to it
- The new project is referred to as the target project or target case

Estimation by Analogy

- The completed projects are referred to as the source projects or source case.
- The effort recorded for the matching source case is used as the base estimate for the target project.
- The estimator calculates an estimate for the new project by adjusting the (base estimate) based on the differences that exist between the two projects.

Estimation by Analogy

- There are software tools that automate this process by selecting the nearest project cases to the new project.
- Some software tools perform that by measuring the
  - Euclidean distance between cases (projects).
  - The Euclidean distance is calculated as follows:

distance= square-root of ((target_parameter$_1$- source_parameter$_1$)^2 .... + (target_parameter$_n$ - source_parameter$_n$)^2)

Estimation by Analogy Example

- Assume that cases are matched on the basis of two parameters, the number of inputs and the number of outputs.
- The new project (target case) requires 7 inputs and 15 output
- You are looking into two past cases (source cases) to find a better analogy with the target project:
  - Project A: has 8 inputs and 17 outputs.
  - Project B: has 5 inputs and 10 outputs.

Which is a more closer match for the new project A or project B?

Answer

- Distance between new project and project A:
  - Square-root of ((7-8) 2 + (15-17) 2)= 2.24
  - Distance between new project and project B:
  - Square-root of ((7-5) 2 + (15-10) 2)= 5.39

Project A is a better match because it has less distance than project B to the new project

Bottom-up estimating

1. Break project into smaller and smaller components

2. Stop when you get to what one person can do in one/two weeks

3. Estimate costs for the lowest level activities

4. At each higher level calculate estimate by adding estimates for lower levels

Bottom-up estimating

A procedural code-oriented approach

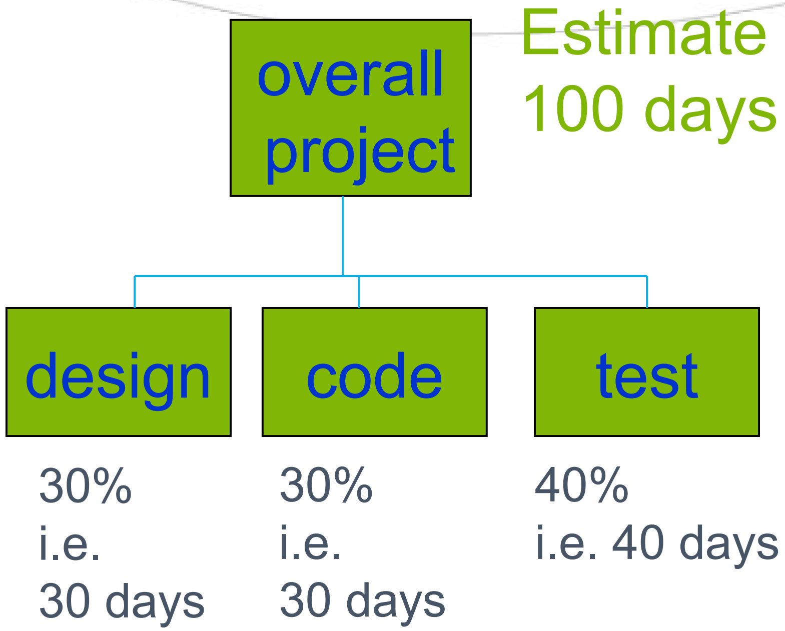
(a) Predict the number and type of software modules in the final system

(b) Estimate the SLOC of each identified module

(c) Estimate the work content, taking into account complexity and technical difficulty.(The practice is to multiply the SLOC estimate by a factor for complexity and technical difficulty.)

(d) Calculate the work-days effort Historical data can be used to provide ratios to convert weighted SLOC to effort

Top-down estimates



- Produce overall estimate using effort driver(s)
- distribute proportions of overall estimate to components

Top-down Estimation

- It is associated with parametric or algorithmic models.
- A formula for a parametric model:
  - Effort = (System Size) * (Productivity Rate)
  - The model of forecasting the SW development effort has two components
  - System size is a method of assessing the amount of work
  - Productivity rate is a method of assessing the rate of work at which the task can be done

Top-down Estimation

- Example:

System Size = 3 KLOC.

Productivity Rate = 40 days per KLOC.

Effort = (System Size) * (Productivity Rate)

Effort = 3* 40 =120 Days.

Top-down Estimation

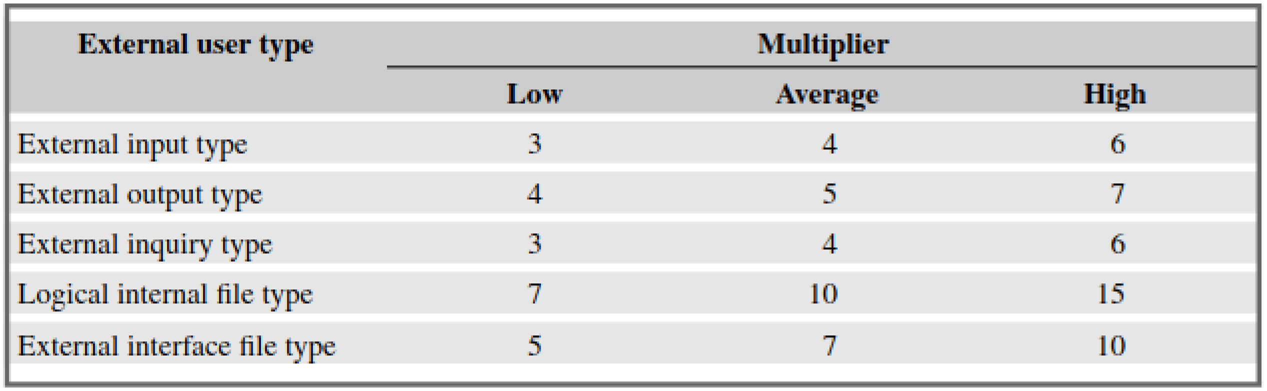
- Other parametric models:
  - **Function points** is concerned more with task sizes.
  - **COCOMO** is concerned more with productivity rate.

FUNCTION POINT ANALYSIS

FPA is used to make estimate of the software project, including its testing in terms of functionality or function size of the software product.

FPs of an application is found out by counting the number and types of functions used in the applications. Various functions used in an application can be put under five types, as shown in Table:

TABLE 5.2 Albrecht complexity multipliers



External user type	Multiplier		
	Low	Average	High
External input type	3	4	6
External output type	4	5	7
External inquiry type	3	4	6
Logical internal file type	7	10	15
External interface file type	5	7	10

Function Point Analysis: In this method, the number and type of functions supported by the software are utilized to find FPC(function point count). The steps in function point analysis are:

Count the number of functions of each proposed type.

Compute the Unadjusted Function Points(UFP).

Find Total Degree of Influence(TDI).

Compute Value Adjustment Factor(VAF).

Find the Function Point Count(FPC).

Find Total Degree of Influence: Use the ’14 general characteristics’ of a system to find the degree of influence of each of them. The sum of all 14 degrees of influence will give the TDI. The 14 general characteristics are: Data Communications, Distributed Data Processing, Performance, Heavily Used Configuration, Transaction Rate, On-Line Data Entry, End-user Efficiency, Online Update, Complex Processing Reusability, Installation Ease, Operational Ease, Multiple Sites and Facilitate Change.

Each of the above characteristics is evaluated on a scale of 0-5.

```
0 - No Influence
1 - Incidental
2 - Moderate
3 - Average
4 - Significant
5 - Essential
```

Compute Value Adjustment Factor(VAF): Use the following formula to calculate VAF

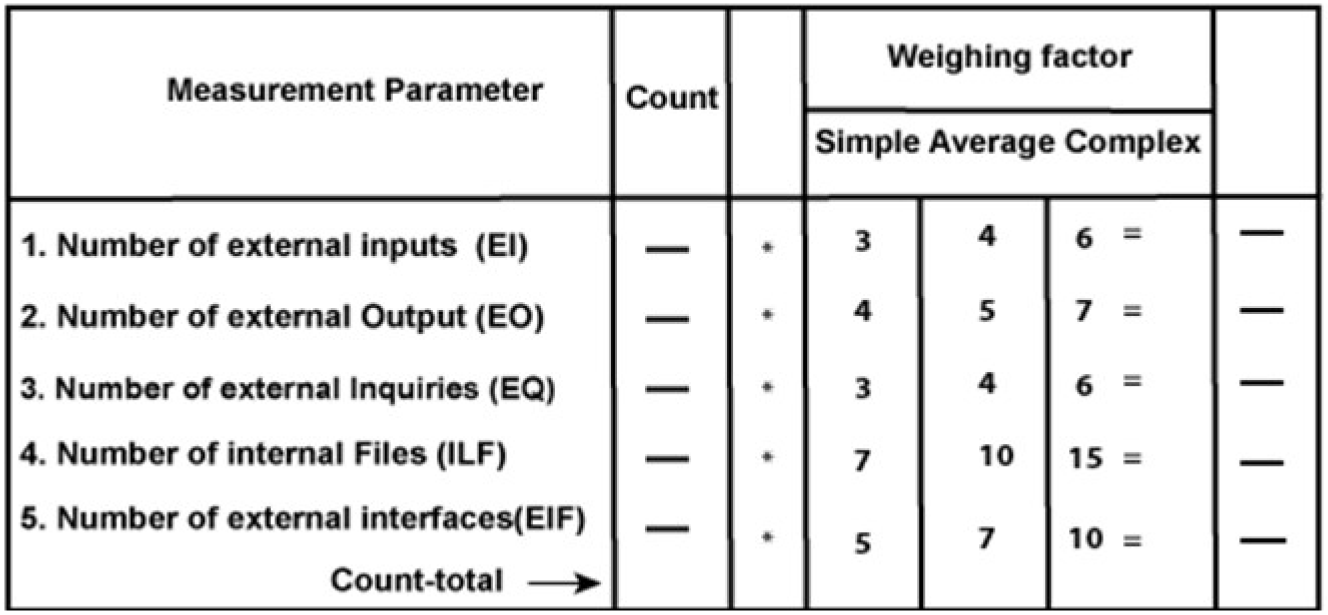
VAF = 0.65 +(0.01*TDI)

Find the Function Point Count: Use the following formula to calculate FPC

FPC = UFP * VAF

All the parameters mentioned above are assigned some weights that have been experimentally determined and are shown in Table

Computing FPs



Measurement Parameter	Count		Weighing factor			
			Simple	Average	Complex	
1. Number of external inputs (EI)	—	*	3	4	6 =	—
2. Number of external Output (EO)	—	*	4	5	7 =	—
3. Number of external Inquiries (EQ)	—	*	3	4	6 =	—
4. Number of internal Files (ILF)	—	*	7	10	15 =	—
5. Number of external interfaces(EIF)	—	*	5	7	10 =	—
Count-total ⟶						

Compute the function point

Number of user inputs = 24

Number of user outputs = 46

Number of inquiries = 8

Number of files = 4

Number of external interfaces = 2

Various processing complexity factors are: 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.

Solution:

Measurement Parameter	Count		Weighing factor
1. Number of external inputs (EI)	24	*	4 = 96
2. Number of external outputs (EO)	46	*	4 = 184
3. Number of external inquiries (EQ)	8	*	6 = 48
4. Number of internal files (ILF)	4	*	10 = 40
5. Number of external interfaces (EIF) Count-total →	2	*	5 = 10 378

so sum of all fi (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43

FP = UFP* VAF

VAF= [0.65 + 0.01 *∑(fi)]

= 378 * [0.65 + 0.01 * 43]

= 378 * [0.65 + 0.43]

= 378 * 1.08 = 408

If team productivity= 24 FP/person-week,then
Effort =FP/productivity=408/24=17 person-week

COCOMO

- COCOMO (CONSTRUCTIVE COST MODEL)

  -First published by Dr. Barry Boehm, 1981

- Interactive cost estimation software package that models the cost, effort and schedule for a new software development activity.
  - Can be used on new systems or upgrades

- Derived from statistical regression of data from a base of 63 past projects.

COCOMO Modes

Boehm’s definition of organic, semidetached, and embedded systems

- 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 in the past and also the team members have a less experience regarding the problem.
- Semi-detached – A software project is said to be a Semi-detached type if the vital characteristics such as team-size, experience, knowledge of the various programming environment lie in between that of organic and Embedded.

COCOMO Modes

Boehm’s definition of organic, semidetached, and embedded systems

- Embedded – A software project with requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.

  All the above system types utilize different values of the constants used in Effort Calculations

COCOMO Models

- Basic Model
  - Used for early rough, estimates of project cost, performance, and schedule
- Intermediate Model
  - Uses Effort Adjustment Factor (EAF) fm 15 cost drivers
- Detailed Model
  - Uses different Effort Multipliers for each phase of project (everybody uses intermediate model)

Basic Model Effort Equation (COCOMO 81)

- $\text{Effort}=A(\text{size})^{\text{exponent}}$
  - A is a constant based on the developmental mode
    - organic = 2.4
    - semi = 3.0
    - embedded = 3.6
  - Size = 1000s Source Lines of Code (KSLOC)
  - Exponent is constant given mode
    - organic = 1.05
    - semi = 1.12
    - embedded = 1.20

Early design model

Estimates can be made after the requirements have been agreed.

● Based on a standard formula for algorithmic models

• PM = A × Size B × M where

PM is the effort in person-months,

• M = PERS × RCPX × RUSE × PDIF × PREX × FCIL × SCED; •

A = 2.94 in initial calibration, Size in KLOC, B varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.

Multipliers

• Multipliers reflect the capability of the developers, the non-functional requirements, the familiarity with the development platform, etc. •

RCPX - product reliability and complexity;

• RUSE - the reuse required;

• PDIF - platform difficulty;

• PREX - personnel experience;

• PERS - personnel capability;

• SCED - required schedule;

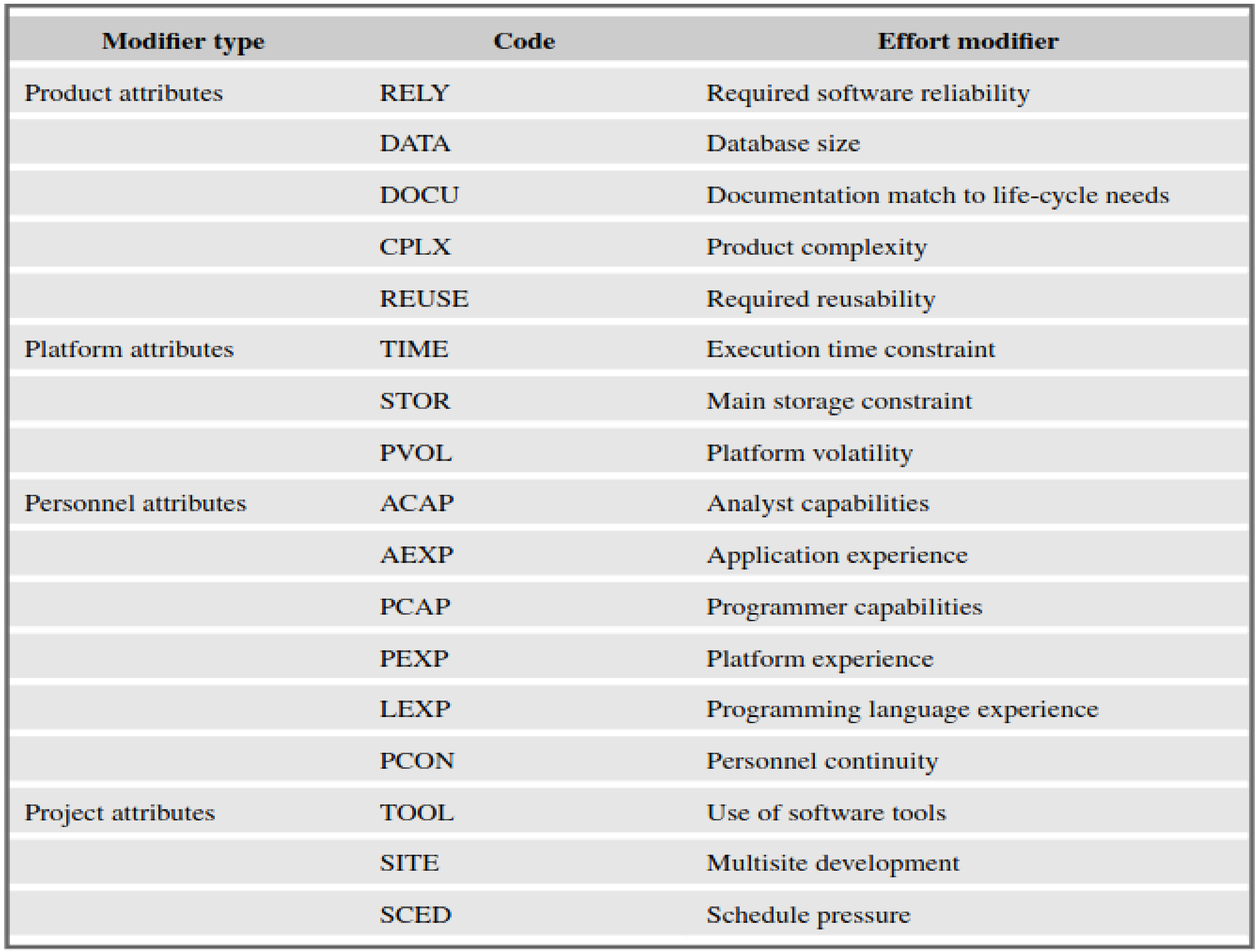
• FCIL - the team support facilities.

TABLE 5.6 COCOMO II Early design effort multipliers

Code	Effort modifier	Extra low	Very low	Low	Nominal	High	Very high	Extra high
RCPX	Product reliability and complexity	0.49	0.60	0.83	1.00	1.33	1.91	2.72
RUSE	Required reusability			0.95	1.00	1.07	1.15	1.24
PDIF	Platform difficulty			0.87	1.00	1.29	1.81	2.61
PERS	Personnel capability	2.12	1.62	1.26	1.00	0.83	0.63	0.50
PREX	Personnel experience	1.59	1.33	1.12	1.00	0.87	0.74	0.62
FCIL	Facilities available	1.43	1.30	1.10	1.00	0.87	0.73	0.62
SCED	Schedule pressure		1.43	1.14	1.00	1.00	1.00	

– Post-architecture model

TABLE 5.7 COCOMO II Post architecture effort multipliers



Modifier type	Code	Effort modifier
Product attributes	RELY	Required software reliability
	DATA	Database size
	DOCU	Documentation match to life-cycle needs
	CPLX	Product complexity
	REUSE	Required reusability
Platform attributes	TIME	Execution time constraint
	STOR	Main storage constraint
	PVOL	Platform volatility
Personnel attributes	ACAP	Analyst capabilities
	AEXP	Application experience
	PCAP	Programmer capabilities
	PEXP	Platform experience
	LEXP	Programming language experience
	PCON	Personnel continuity
Project attributes	TOOL	Use of software tools
	SITE	Multisite development
	SCED	Schedule pressure

– Cost Estimation

Project cost can be obtained by multiplying the estimated effort (in man-month, from the effort estimate) with the manpower cost per month.

However, in addition to manpower cost, a project would incur several other types of costs which we shall refer to as the overhead costs.

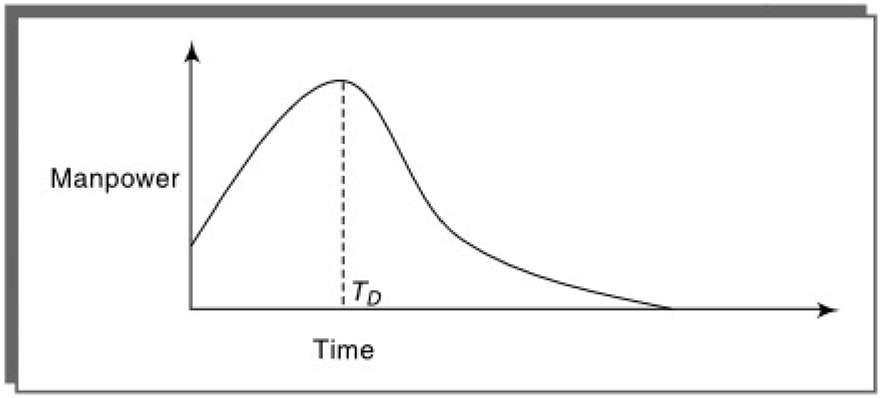
The overhead costs would include the costs of hardware and software required for the project and the company overheads for administration, offi ce space, etc. Depending on the expected values of the overhead costs, the projectmanager has to suitably scale up the cost estimated by using the COCOMO formula.

– Staffing Pattern

After the effort required to complete a software project has been estimated, the staffing requirement for the project can be determined

We use Norden’s and Putnam’s results.

Norden concluded that the staffing pattern for any R&D project can be approximated by the Rayleigh distribution curve shown in Figure



– Putnam’s work

Putnam suggested that starting from a small number of developers, there should be a staff build-up and after a peak size has been achieved, staff reduction is required.

However, the staff build-up should not be carried out in large instalments. Experience shows that a very rapid build-up of project staff any time during the project development correlates with schedule slippage.

Effect of Schedule Compression

It is quite common for a project manager to encounter client requests to deliver products faster, that is, to compress the delivery schedule.

It is therefore important to understand the impact of schedule compression on project cost. Putnam studied the effect of schedule compression on the development effort and expressed it in the form of the following equation:

$$pm_{new} = pm_{org} \times \left(\frac{td_{org}}{td_{new}} \right)^4$$

Where pm_{new} is the new effort, pm_{org} is the originally estimated effort and td_{org} is the originally estimated time for project completion and td_{new} is the compressed schedule.