MODULE:IV

SOFTWARE PROJECT MANAGEMENT

INTRODUCTION

- Many software projects **fail**:

  due to **faulty** project management practices
- It is important to learn

  -different **aspects of software project management.**

INTRODUCTION…

- Goal of software project management:

  …enable a group of engineers to work efficiently towards successful completion of a software project.

Responsibility of Project Managers

- Project proposal writing (justification, timeline, resource requirement)
- Project cost estimation
- Scheduling
- Project staffing
- Project monitoring and control
- Risk management
- Managerial report writing and presentations etc.

- A project manager's activities are varied.

  -can be broadly classified into:

  project **planning,**

  project **monitoring** and **control** activities.

Project Planning

- **Once a project is found to be feasible, project managers undertake project planning.**
- it is the job in between the phase **feasibility study** and **requirement analysis and specification.**

Project Planning Activities

- **Estimation:**

  -Effort, cost, resource, and project duration
- **Project scheduling:**

  - Timeline
- **Staff organization:**

  -Staffing plans
- **Risk handling:**

  -Identification, Analysis, and Abatement (reduction) procedures
- **Miscellaneous plans:**

  -Quality assurance plan, Configuration management plan, etc.

Project Planning…

Requires utmost care and attention ---

commitments to **unrealistic** time and resource estimates result in:

- irritating delays.
- customer dissatisfaction
- adverse affect on team morale
- poor quality work
- project failure.

Sliding Window Planning

- Involves project planning over several stages:
  --protects managers from making big commitments too early.
- The information base gradually improves as the project progresses through different phases
- After the completion of every phase, the project managers can plan each subsequent phase more accurately and with increasing levels of confidence.

SPMP Document…

- After planning is complete:

Document the plans in a Software Project Management Plan(SPMP) document.

Organization of SPMP Document

- **Introduction (Objectives ,Major Functions, Performance Issues, Management and Technical Constraints)**
- **Project Estimates (Historical Data, Estimation Techniques, Effort, Cost, and Project Duration Estimates)**
- **Project Resources Plan (People, Hardware and Software)**
- **Schedules (Work Breakdown Structure, Gantt Chart Representation, PERT Chart Representation)**
- **Risk Management Plan (Risk Analysis, Risk Identification, Risk Estimation, Abatement Procedures)**
- **Project Tracking and Control Plan**
- **Miscellaneous Plans(Process Tailoring, Quality Assurance)**

Software Estimations…

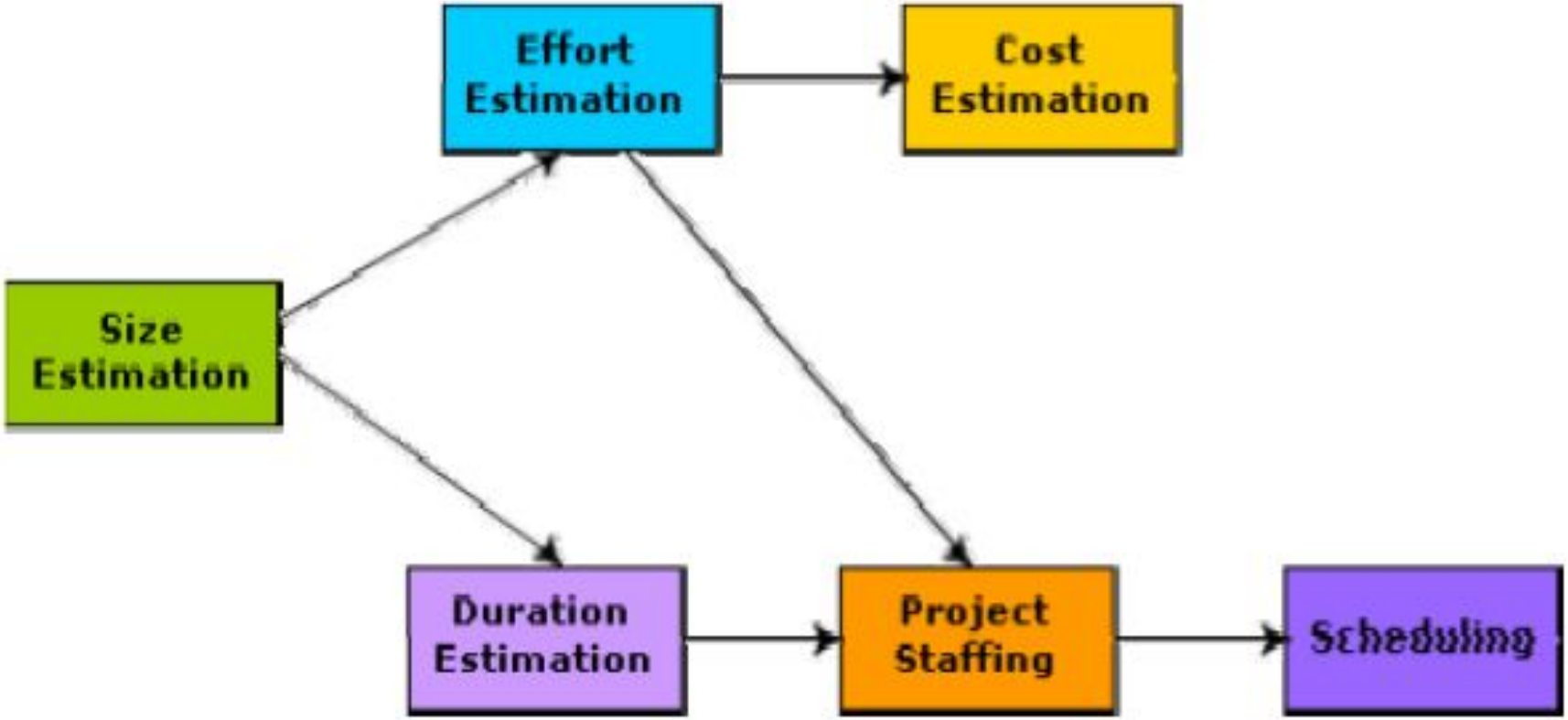


Fig. 11.1: Precedence ordering among planning activities

The 4 P's(Management Spectrum)

- **People** — the most important element of a successful project
- **Product** — the software to be built
- **Process** — the set of framework activities and software engineering tasks to get the job done
- **Project** — all work required to make the product a reality.

Organization Structure

- **Functional Organization:**

-Engineers are organized into functional groups, e.g.

specification, design, coding, testing, maintenance, etc.

-Engineers from functional groups get assigned to different projects

Advantages of Functional Organization

- Specialization
- Ease of staffing
- Good documentation is produced
- different phases are carried out by different teams of engineers.
- Helps identify errors earlier

Disadvantages

- lack of communication between the functional groups within an organization, making the organization slow and inflexible.
- Job rotation is not there.

Organization Structure

- **Project Organization**
- Engineers get assigned to a project for the entire duration of the project
- Same set of engineers carry out all the phases
- **Advantages:**

-Engineers save time on learning details of every project.

-Leads to job rotation

Team Structures

- Problems of different **complexities** and **sizes** require different team structures:

  -Chief-programmer team

  -Democratic team

  -Mixed organization

Team Structures…

Chief Programmer Team

- A **senior engineer** provides technical leadership:
- **partitions** the task among the team members.
- **verifies** and **integrates** the products developed by the members.

Team Structures…

Chief Programmer Team

- Works well when

  -the task is well understood

  -also within the intellectual grasp of a single individual,
- It ensures early completion of small process.
- The chief programmer should also have the capability of motivating teams.

Team Structures…

Chief Programmer Team

- Chief programmer team is subject to **single point failure**:

  -too much **responsibility** and **authority** is assigned to the chief programmer.

Team Structures…

Democratic Teams

- Suitable for:
  -**small projects** requiring less than five or six engineers
  -**research-oriented projects**
- A manager provides **administrative leadership**:
  -at different times different members of the group provide **technical leadership**.

Team Structures…

- Democratic organization provides

  -**higher morale** and **job satisfaction** to the engineers

  - therefore leads to **less employee turnover**.

  -Suitable for **less understood problems**,

  -a **group of engineers** can invent better solutions than a **single individual**.

Team Structures…

Democratic Teams

- Disadvantage:

  -team members may waste a lot time arguing about trivial points in absence of any authority in the team.

Team Structures…

Mixed Control Team Organization

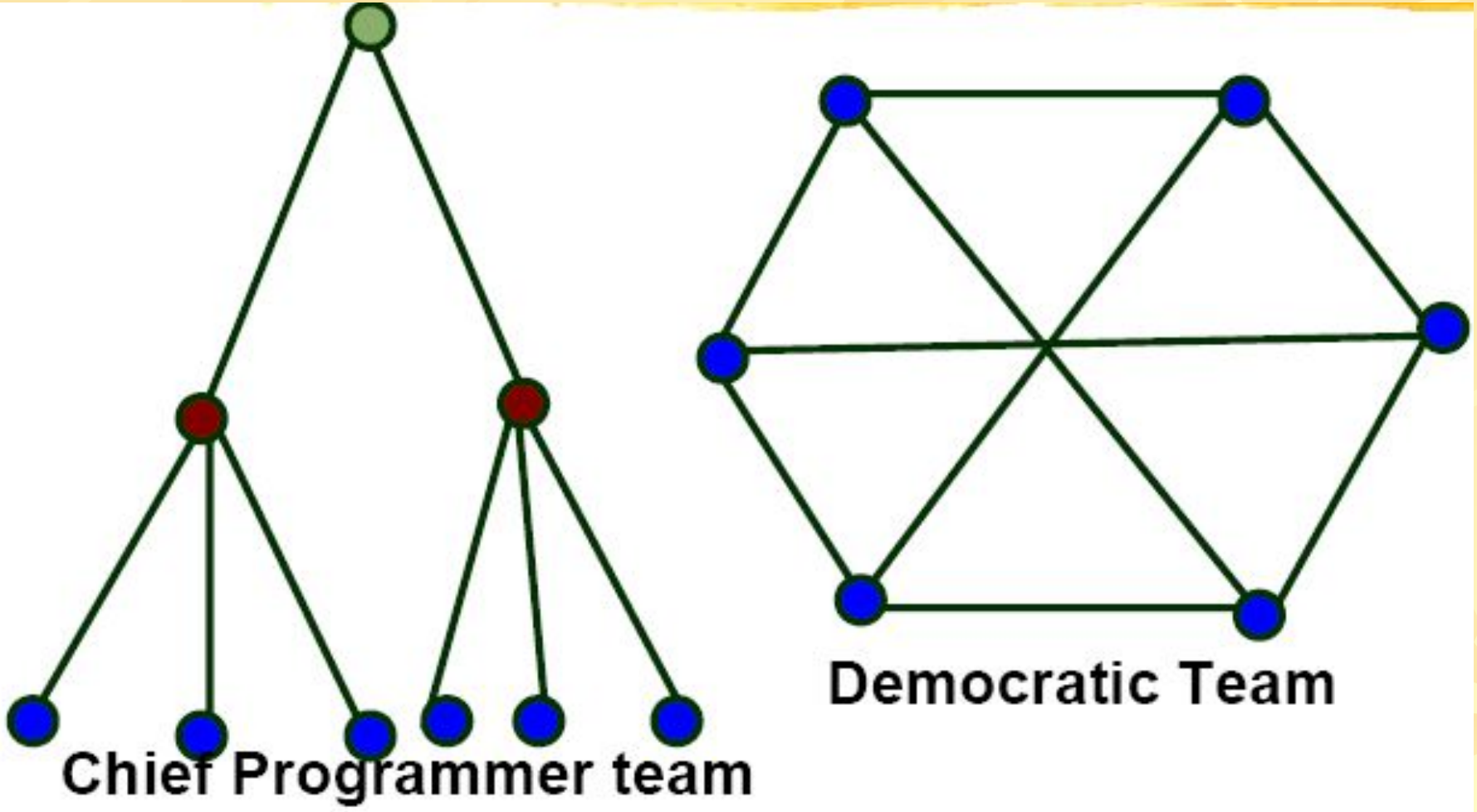
- Draws upon ideas from both:

  -democratic organization and

  -chief-programmer team organization.
- Communication is limited

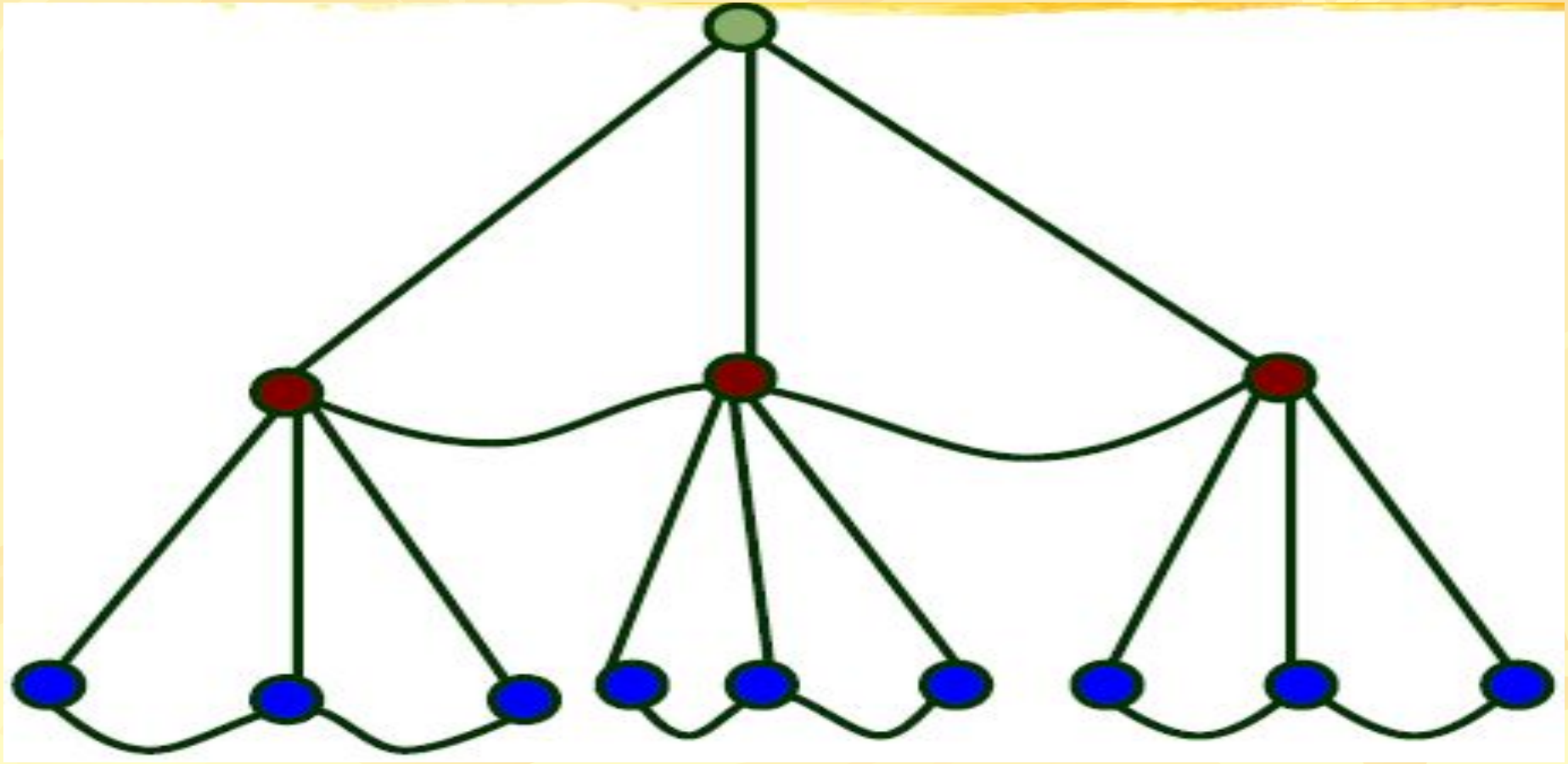
  -to a small group that is most likely to benefit from it.
- Suitable for large organizations.

Team Structures…

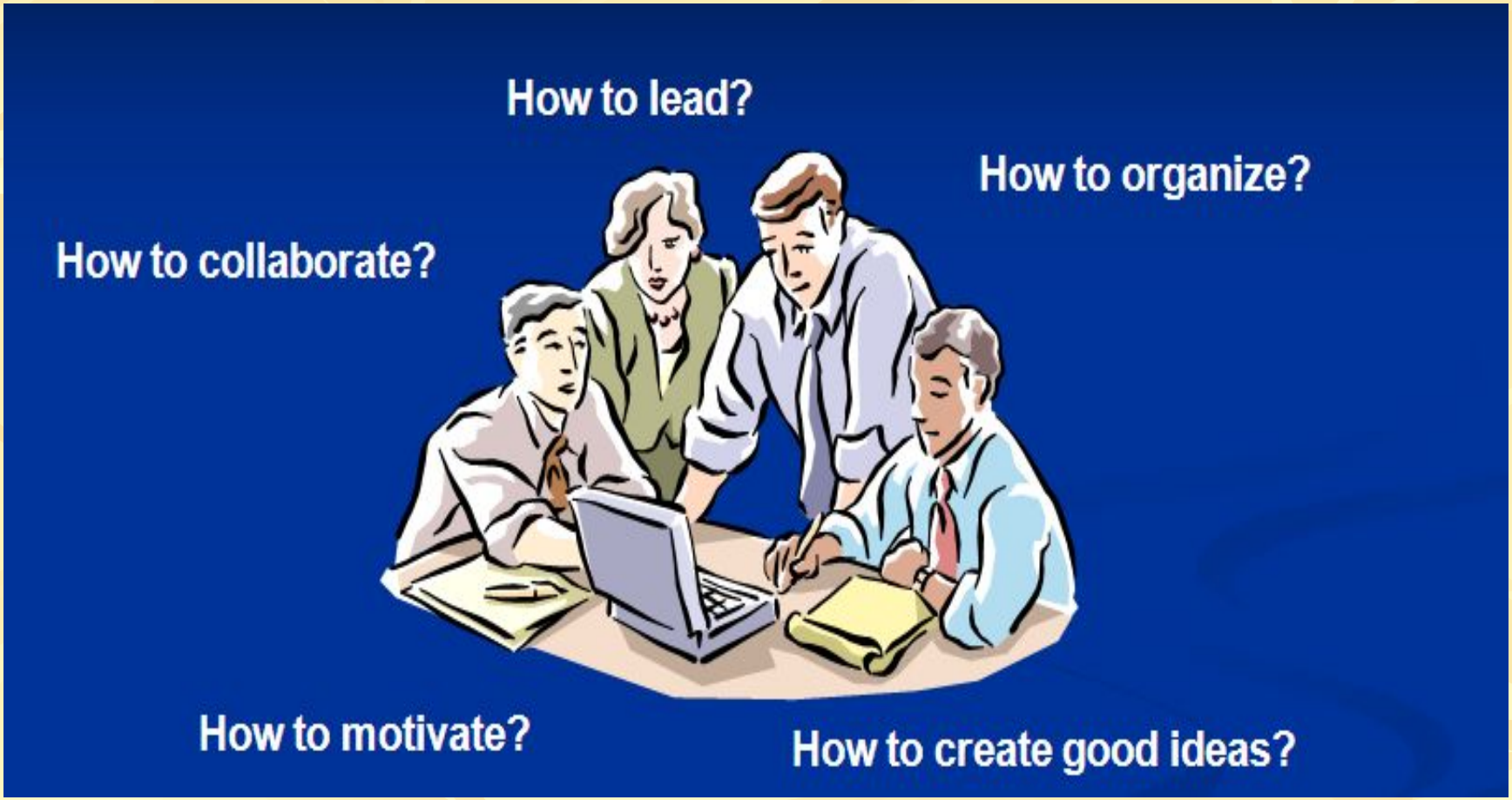


Team Structures…

- **Mixed Team Organization**



Software Teams



Team Leader

- **The MOI Model**
  - **Motivation.** The ability to encourage (by "push or pull") technical people to produce to their best ability.
  - **Organization.** The ability to mold existing processes (or invent new ones) that will enable the initial concept to be translated into a final product.
  - **Ideas or Innovation.** The ability to encourage people to create and feel creative even when they must work within bounds established for a particular software product or application.

Software Cost Estimation

- Three main approaches to estimation:
  - -Empirical
  - -Heuristic
  - -Analytical

Software Cost Estimation…

- **Empirical techniques:**

  an educated guess based on past experience.

- **Heuristic techniques:**

  assume that the characteristics to be estimated can be expressed in terms of some mathematical expression.

- **Analytical techniques:**

  derive the required results starting from certain simple assumptions.

Metrics for size estimation of a project

- The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.
- Generally three methods are used to estimate the **size** of a project

-Lines Of Code

-Function Point

-Feature Point

Software Size Metrics

- **LOC (Lines of Code):**

  -Simplest and most widely used metric.

  -Comments and blank lines should not be counted.

Disadvantages of Using LOC

- Size can vary with coding style.
- Focuses on coding activity alone.
- Correlates poorly with quality and efficiency of code.
- Penalizes higher level programming languages, code reuse, etc.

Disadvantages of Using LOC…

- if (x>y) printf(“KIIT”);
  else printf (“KISS”);
  LOC=2
- (x>y)?printf (“KIIT”) :printf (“KISS”);
  LOC=1

Disadvantages of Using LOC…

- Measures lexical/textual complexity only.
- does not address the issues of structural or logical complexity.
- Difficult to estimate LOC from problem description.
- So not useful for Project Planning.

Function Point Analysis

- **Function Point Analysis (FPA)** is an ISO recognized method to measure the functional size of an information system. The functional size reflects the amount of functionality that is relevant to and recognized by the user in the business. It is independent of the technology used to implement the system.

Function Point Analysis

- The unit of measurement is "function points". So, FPA expresses the functional size of an information system in a number of function points (for example: the size of a system is 314 fp).

Function Point Metric

- Overcomes some of the shortcomings of the LOC metric
- For FP we have to calculate the

  UFP(**U**n **a**djusted **F**unction **P**oint)

  DI(**D**egree of **I**nfluence)

  TCF (**T**echnical **C**omplexity **F**actor)
- **UFP=4*inputs + 5 *Outputs + 4*inquiries + 10 *files + 10 *interfaces**

Function Point Metric…

- **Input**:

  A set of related inputs is counted as one input.

- **Output:**

  A set of related outputs is counted as one output.

- **Inquiries:**

  Each user query type is counted.

Function Point Metric…

- **Files**:

  Files are logically related data and thus can be data structures or physical files.

- **Interface:**

  Data transfer to other systems(GUI).

Function Point Metric…

- **DI** depends on **14 factors** like access time , response time, quality, reliability and high transfer rate etc.
- The value DI is from 0 to 6 where 0 represents not at all required and 6 represents strongly required.
- After the calculation of DI we go for the **TCF(Technical Complexity Factor)** calculation.

Function Point Metric…

- **TCF=0.65+(0.01*DI)**
- DI value ranges from 0 to 84
- TCF value vary from 0.65 to 1.49
- After the completion of UFP and TCF we go for calculation of FP
  - **FP=UFP*TCF**

Feature Point Metric

- In function point metric the size of a function is considered to be independent of its complexity. Also it assume all function have **equal algorithmic complexity** and the effort required to design and develop any two functionalities of the system is same.

Feature Point Metric…

- The feature point method says that the **function which is more complex will have larger code as compared to a less complex function**.
- considers an extra parameter (Algorithm Complexity)

Feature Point Metric…

Both the function point and feature point method estimate the size from requirement specification, hence they are programming language independent as well as coding style independent but the LOC method depend upon the coding style as well as programming language.

Empirical Size Estimation Techniques

Expert Judgment:

An expert makes an **educated guess** of the problem size after analyzing the problem thoroughly.

-Suffers from human error.

Delphi Estimation:

overcomes some of the problems of expert judgment.

Expert judgment

- Experts divide a software product into component units:
- e.g. GUI, database module, data communication module, billing module, etc.

-Add up the **guesses** for each of the components

Delphi Estimation

Team of Experts and a **coordinator**.

-Experts carry out estimation independently:

- **mention the unusual characteristics of the product behind their estimation.**

- coordinator notes down any extraordinary characteristics then
- circulates among experts
- Experts **re-estimate.**
- Experts never meet each other to discuss their viewpoints.

Heuristic Estimation Techniques

- **Single Variable Model:**
- Parameter to be Estimated=C1(Estimated Characteristic)d1
- **Multivariable Model:**
- Assumes that the parameter to be estimated depends on more than one characteristic.
- **Parameter to be Estimated=C1(Estimated Characteristic)d1+ C2(Estimated Characteristic)d2+…**
- Usually more accurate than single variable models.

COCOMO Model

- COCOMO (**CO**nstructive **CO**st **MO**del) proposed by Boehm.
- Divides software product developments into 3 categories:

  -Organic

  -Semidetached

  -Embedded

COCOMO Product Classes

- Roughly correspond to:

  -application, utility and system programs respectively.

  -Data processing and scientific programs are considered to be **application programs**.

  -Compilers, linkers, editors, etc., are **utility programs.**

  -Operating systems and real-time system programs, etc. are **system programs**.

Elaboration of Product Classes

- **Organic:**

  -Relatively small groups working to develop well-understood applications.

- **Semidetached:**

  -Project team consists of a mixture of experienced and inexperienced staff.

- **Embedded:**

  -The software is strongly coupled to complex hardware, or real-time systems.

COCOMO Model…

- For each of the three product categories:
  From size estimation (in KLOC), Boehm provides equations to predict:
  -project duration in months
  -effort in programmer-months/person-month
- Boehm obtained these equations:
  examined historical data collected from a large number of actual projects.

COCOMO Model…

- Software cost estimation is done through three stages:

  -Basic COCOMO,

  -Intermediate COCOMO,

  -Complete COCOMO.

Basic COCOMO Model…

- Gives only an approximate estimation:

Effort = a1 (KLOC)a2 PM

Tdev = b1 (Effort)b2 months

Basic COCOMO…

- **KLOC** is the estimated Kilo Lines of source Code,
- a1,a2,b1,b2 are constants for different categories of software products,
- **Tdev** is the estimated time to develop the software in months,
- Effort estimation is obtained in terms of **Person Months** (PMs)

Estimation of Effort

Organic :

Effort = 2.4 (KLOC)1.05 PM

Semi-detached:

Effort = 3.0(KLOC)1.12 PM

Embedded:

Effort = 3.6 (KLOC)1.20PM

Estimation of Development Time

Organic:

Tdev = 2.5 (Effort)0.38 Months

Semi-detached:

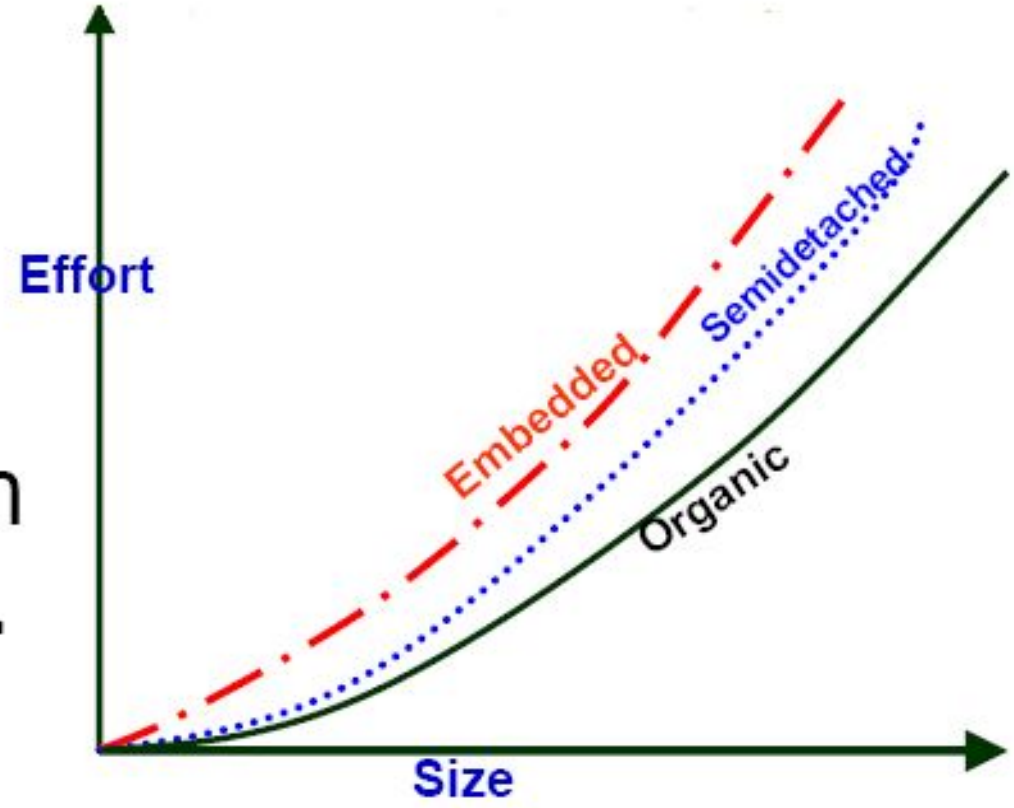
Tdev = 2.5 (Effort)0.35 Months

Embedded:

Tdev = 2.5 (Effort)0.32 Months

Basic COCOMO…

Effort is somewhat super-linear in problem size.



Basic COCOMO…

Development time

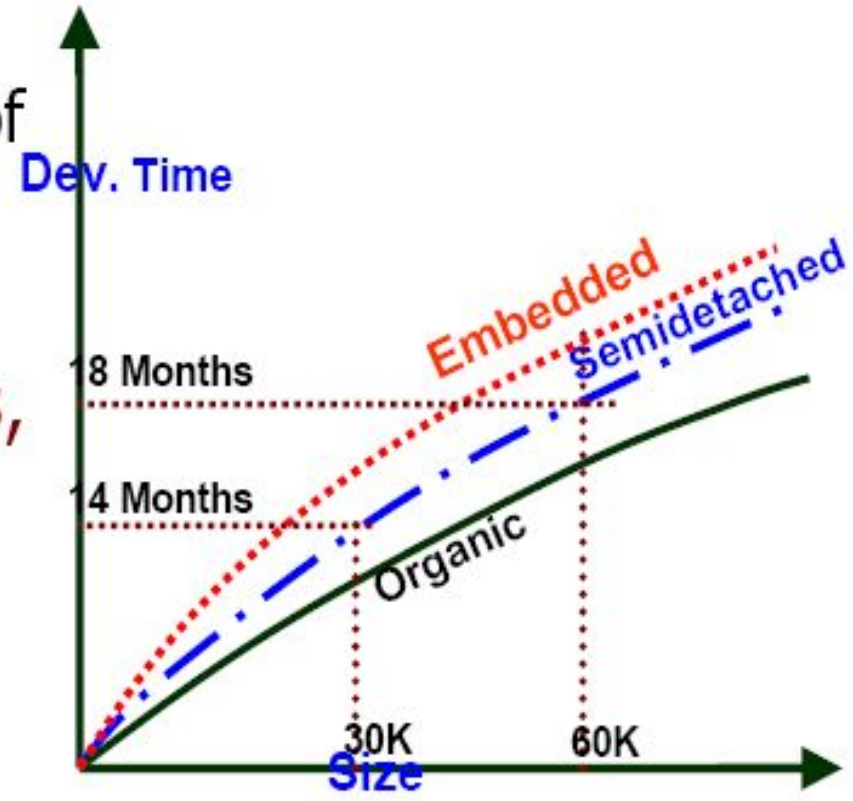
sublinear function of product size.

When product size increases two times,

development time does not double.

Time taken:

almost same for all the three product categories.



Basic COCOMO…

- Development time does not increase linearly with product size:

  -For larger products more **parallel activities** can be identified:

  can be carried out simultaneously by a number of engineers.

Basic COCOMO…

- Development time is roughly the same for all the three categories of products:

  For example, a 60 KLOC program can be developed in approximately 18months regardless of whether it is of organic, semidetached, or embedded type.

- There is more scope for parallel activities for system and application programs, than utility programs.

Basic COCOMO…

Example:

- The size of an organic software product has been estimated to be 32,000 lines of source code.

  Effort = 2.4*(32)1.05 = 91 PM

  Nominal development time = 2.5*(91)0.38 = 14 months

Intermediate COCOMO

- Basic COCOMO model assumes effort and development time depend on product size alone.

-However, several parameters affect effort and development time:

-Reliability requirements

-Availability of CASE tools and modern facilities to the developers

-Size of data to be handled

Intermediate COCOMO…

- For accurate estimation,

  -the effect of all relevant parameters must be considered:

  -Intermediate COCOMO model recognizes this fact:

  refines the initial estimate obtained by the basic COCOMO by using a set of **15 cost drivers (multipliers).**

Intermediate COCOMO…

- Rate different parameters on a scale of **one to three:**

  Depending on these ratings,

  -multiply cost driver values with the estimate obtained using the basic COCOMO.

Intermediate COCOMO…

- Cost driver classes:

  -**Product**: Inherent complexity of the product, reliability requirements of the product, etc.

  -**Computer**: Execution time, storage requirements, etc.

  -**Personnel**: Experience of personnel, etc.

  -**Development Environment**: Sophistication of the tools used for software development

Shortcoming of basic and intermediate COCOMO Models

- Both models:

-consider a software product as a **single homogeneous entity:**

-However, most large systems are made up of several **smaller sub-systems**.

-Some sub-systems may be considered as organic type, some may be considered embedded, etc.

-for some the reliability requirements may be high and so on.

Complete COCOMO

- Cost of each sub-system is estimated separately.
- Costs of the sub-systems are added to obtain total cost.
- Reduces the margin of error in the final estimate