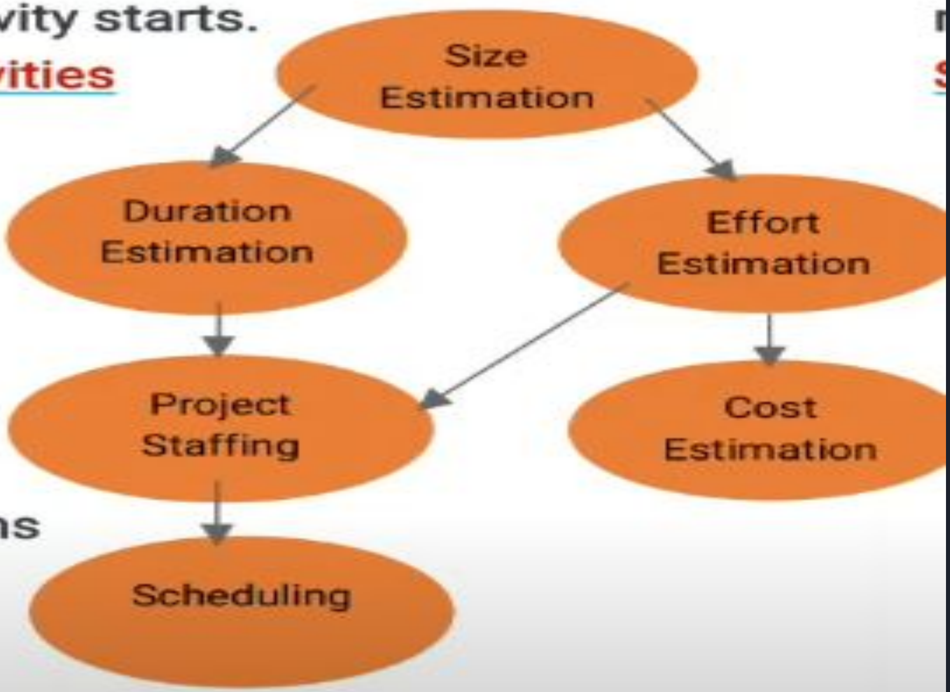
Project Planning

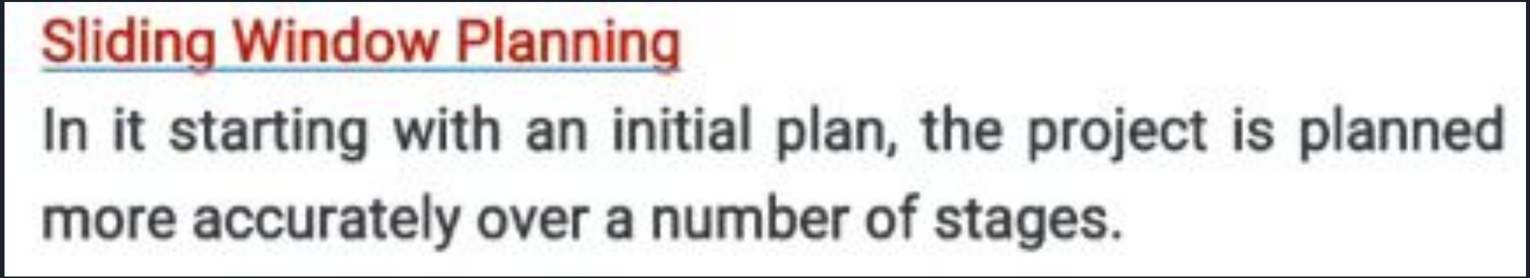
PROJECT PLANNING

Project planning is undertaken and completed before any development activity starts.

Project Planning Activities

- Estimation
  - ✓ Cost
  - ✓ Duration
  - ✓ Effort
- Scheduling
- Staffing
- Risk management
- Miscellaneous plans




Sliding Window Planning

In it starting with an initial plan, the project is planned more accurately over a number of stages.

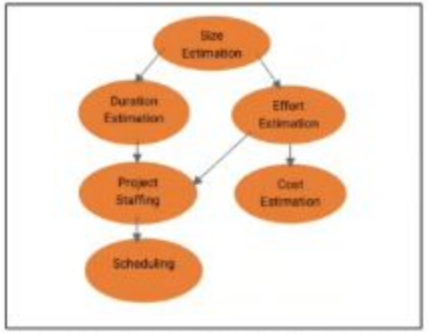
SPMP Document of Project Planning

1. Introduction
2. Project estimates
3. Schedule
4. Project resources
5. Staff organisation
6. Risk management plan
7. Project tracking and control
8. Miscellaneous plan

Project size Estimation

PROJECT SIZE ESTIMATION

The project size is a measure of the problem complexity in terms of the effort and time required to develop the product.



Two metrics are used to measure size

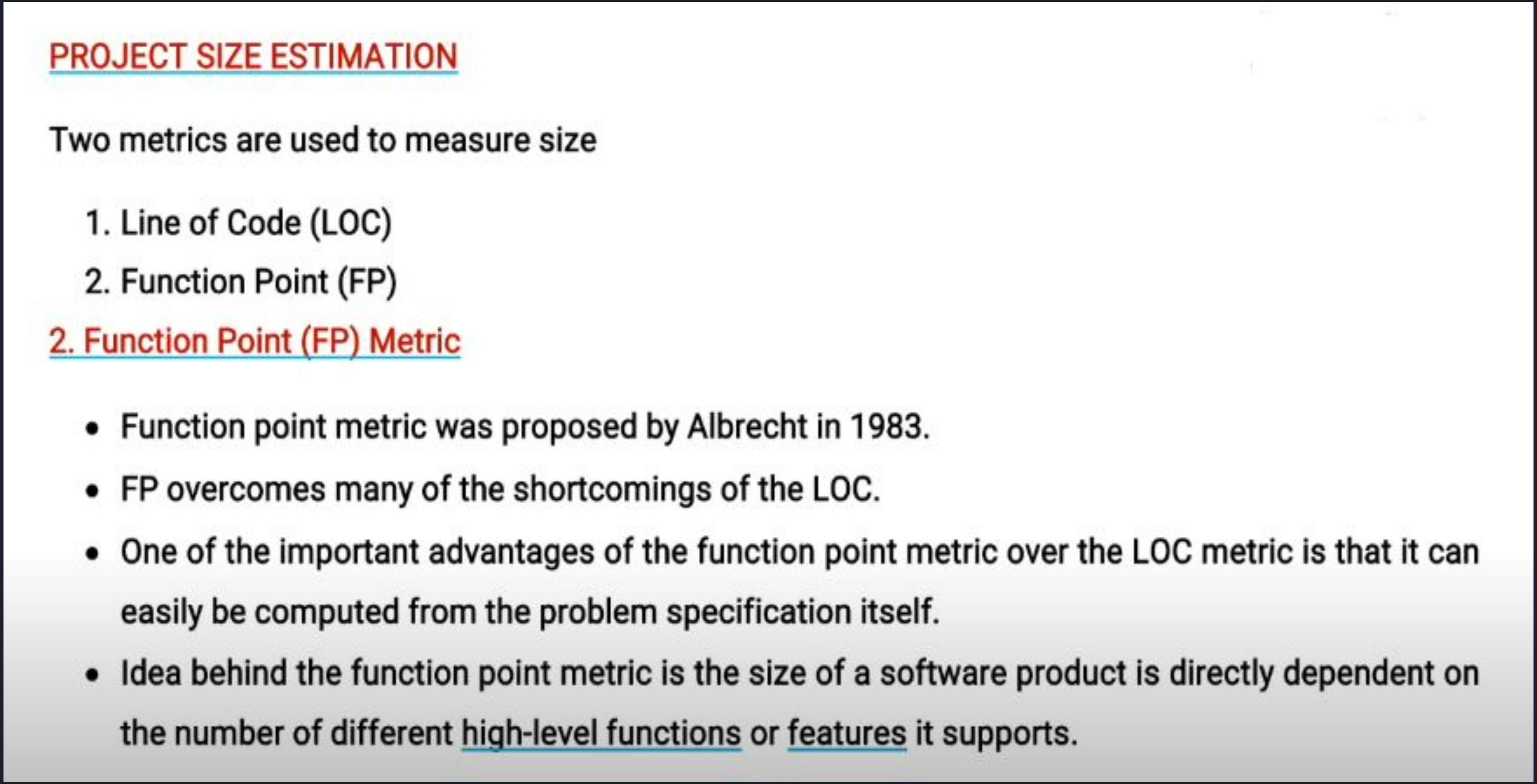
- Lines of code (LOC).
- Function point (FP).

Lines of Code (LOC)

LOC is the simplest among all metrics available to measure project size. This metric measures the size of a project by counting the number of source instructions in the developed program.

Shortcomings of LOC

- LOC is a measure of coding activity alone.
- LOC count depends on the choice of specific instructions.
- LOC measure correlates poorly with the quality and efficiency of the code.
- LOC metric penalises use of higher-level programming languages and code reuse.
- LOC metric measures the lexical complexity of a program and does not address the more important issues of logical and structural complexities



PROJECT SIZE ESTIMATION

Two metrics are used to measure size

1. Line of Code (LOC)
2. Function Point (FP)

2. Function Point (FP) Metric

- Function point metric was proposed by Albrecht in 1983.
- FP overcomes many of the shortcomings of the LOC.
- One of the important advantages of the function point metric over the LOC metric is that it can easily be computed from the problem specification itself.
- Idea behind the function point metric is the size of a software product is directly dependent on the number of different high-level functions or features it supports.

Function Point (FP) Metric Computation

In FP The size of a software product is computed using different characteristics of the product identified in its requirements specification.

It is computed using the following three steps

Step 1: Compute the Unadjusted Function Point (UFP).
Step 2: Refine UFP parameters.
Step 3: Compute FP by further refine UFP based on complexity of the overall project.

Step 1: Compute the Unadjusted Function Point (UFP)

UFP is computed as the weighted sum of 5 characteristics of a product as

UFP = (Number of inputs)*4 + (Number of outputs)*5 + (Number of inquiries)*4 + (Number of files)*10 + (Number of interfaces)*10

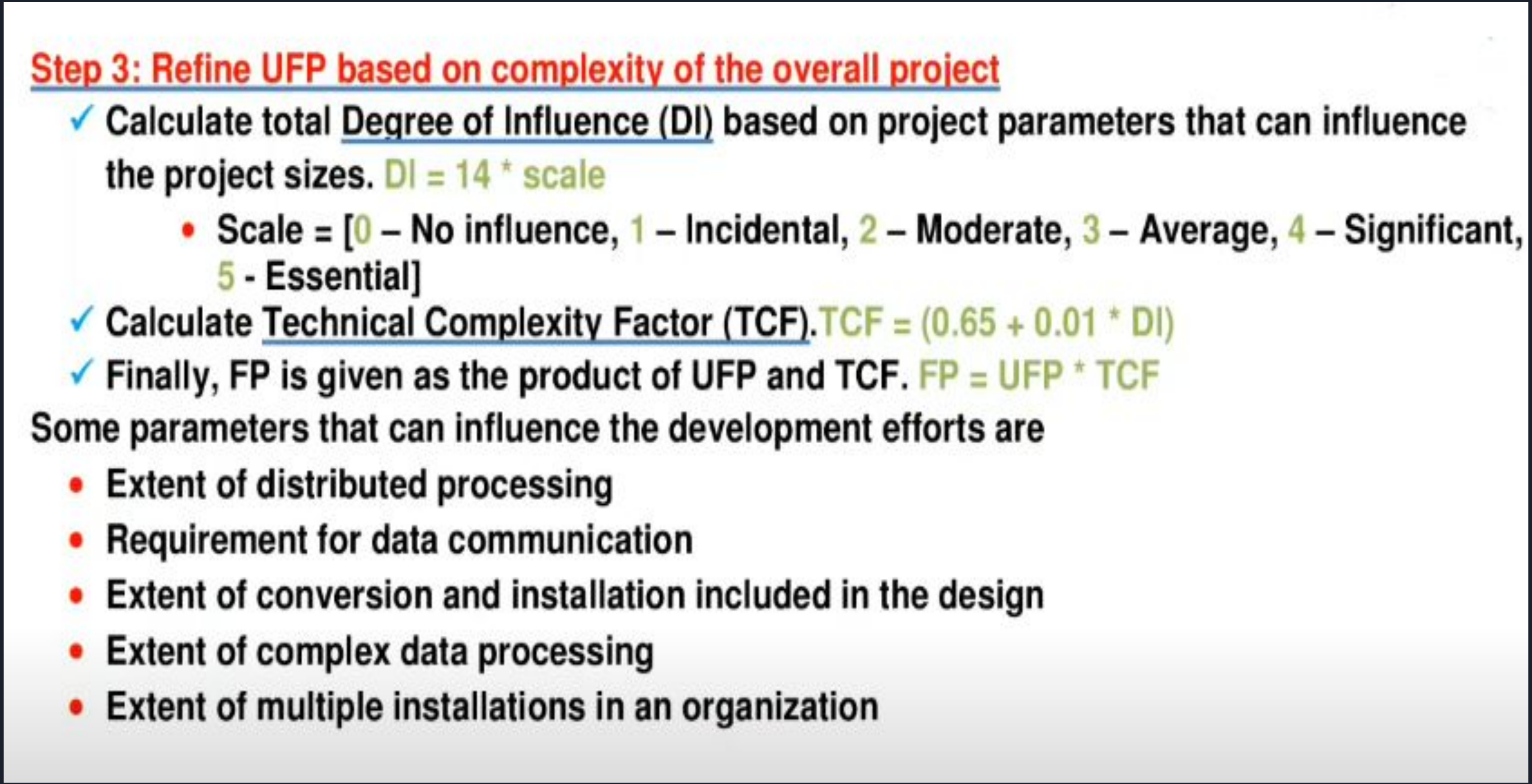
Five different parameters are

1. Number of inputs.
2. Number of outputs.
3. Number of inquiries
4. Number of files.
5. Number of interfaces.

Step 2 Refine parameters

Complexity of each parameter is graded into three broad categories simple, average, or complex. The weights for the different parameters are determined based on the numerical values as

Type	Simple	Average	Complex
Input(I)	3	4	6
Output (O)	4	5	7
Inquiry (E)	3	4	6
Number of files (F)	7	10	15
Number of interfaces	5	7	10



Step 3: Refine UFP based on complexity of the overall project

- ✓ Calculate total Degree of Influence (DI) based on project parameters that can influence the project sizes. DI = 14 * scale
  - Scale = [0 – No influence, 1 – Incidental, 2 – Moderate, 3 – Average, 4 – Significant, 5 - Essential]
- ✓ Calculate Technical Complexity Factor (TCF). TCF = (0.65 + 0.01 * DI)
- ✓ Finally, FP is given as the product of UFP and TCF. FP = UFP * TCF

Some parameters that can influence the development efforts are

- Extent of distributed processing
- Requirement for data communication
- Extent of conversion and installation included in the design
- Extent of complex data processing
- Extent of multiple installations in an organization

Example

Given the following values, compute function point when all degree of influence (scale) and weighting factors are average.

User Input = 50
User Output = 40
User Inquiries = 35
User Files = 6
External Interface = 4

Solution

Calculate UFP (Weighting factors are average)

UFP = (50 * 4) + (40 * 5) + (35 * 4) + (6 * 10) + (4 * 7) = 628

Calculate Degree of Influence (DI) (Scale is average)

DI = 14 * 3 = 42

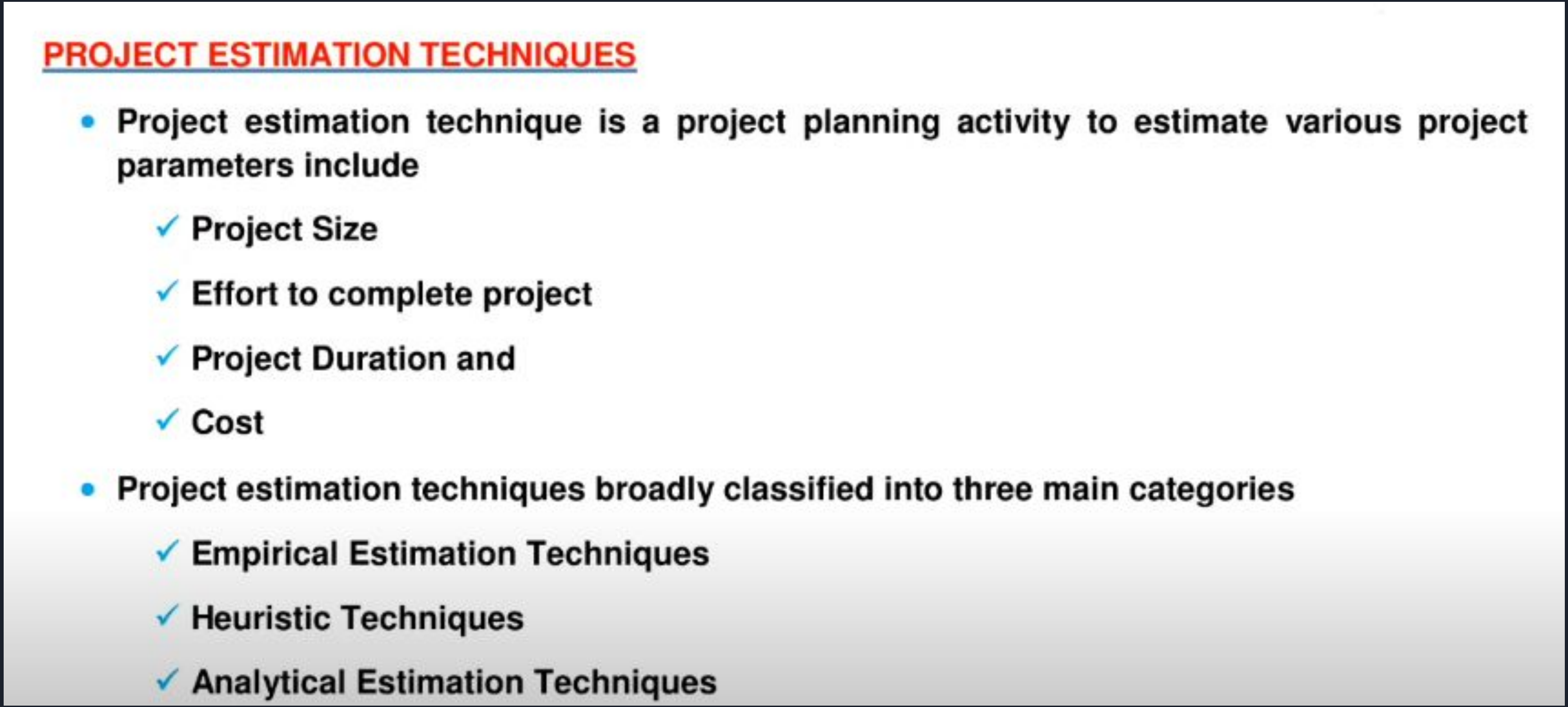
Calculate Technical Complexity Factor (TCF)

TCF = (0.65 + 0.01 * DI) = (0.65 + 0.01 * 42) = 1.07

Calculate FP

FP = UFP * TCF = 628 * 1.07 = 671. 96

PROJECT ESTIMATION TECHNIQUES



PROJECT ESTIMATION TECHNIQUES

- Project estimation technique is a project planning activity to estimate various project parameters include
  - ✓ Project Size
  - ✓ Effort to complete project
  - ✓ Project Duration and
  - ✓ Cost
- Project estimation techniques broadly classified into three main categories
  - ✓ Empirical Estimation Techniques
  - ✓ Heuristic Techniques
  - ✓ Analytical Estimation Techniques

EMPIRICAL ESTIMATION TECHNIQUES

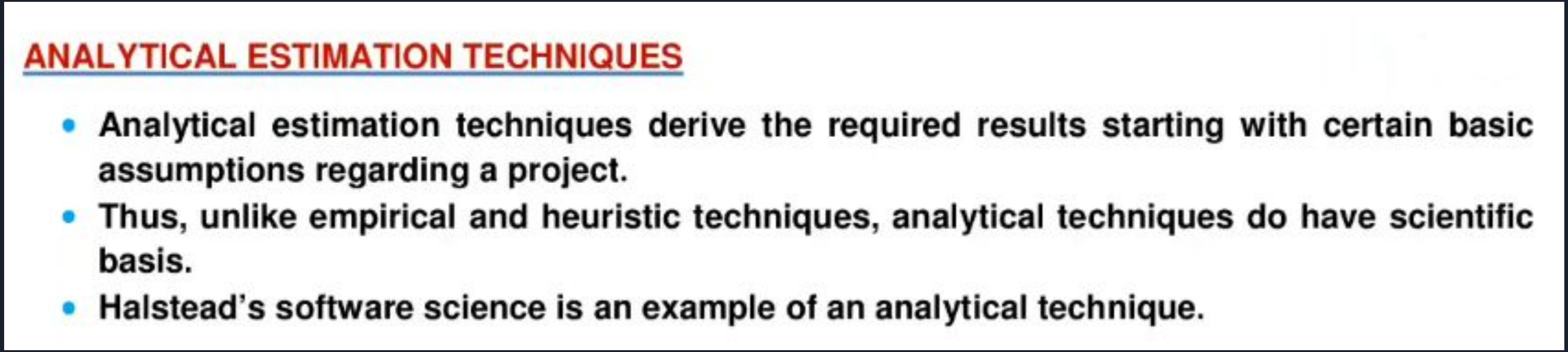
- Based on Experience not Principal.
- Based on making an educated guess of the project parameters.
- Based on common sense.
- Experience with development of similar products is helpful for this technique.
- Two popular empirical estimation techniques are

  1. Expert judgement Technique and
  2. Delphi Cost Estimation.

HEURISTIC TECHNIQUES

- Based on research and experience.
- Project parameters modelled using suitable mathematical expressions.
- Once the basic (independent) parameters are known, the other (dependent) parameters can be easily determined by substituting value of basic parameters in the mathematical expression.
- Different heuristic estimation models can be divided into two categories

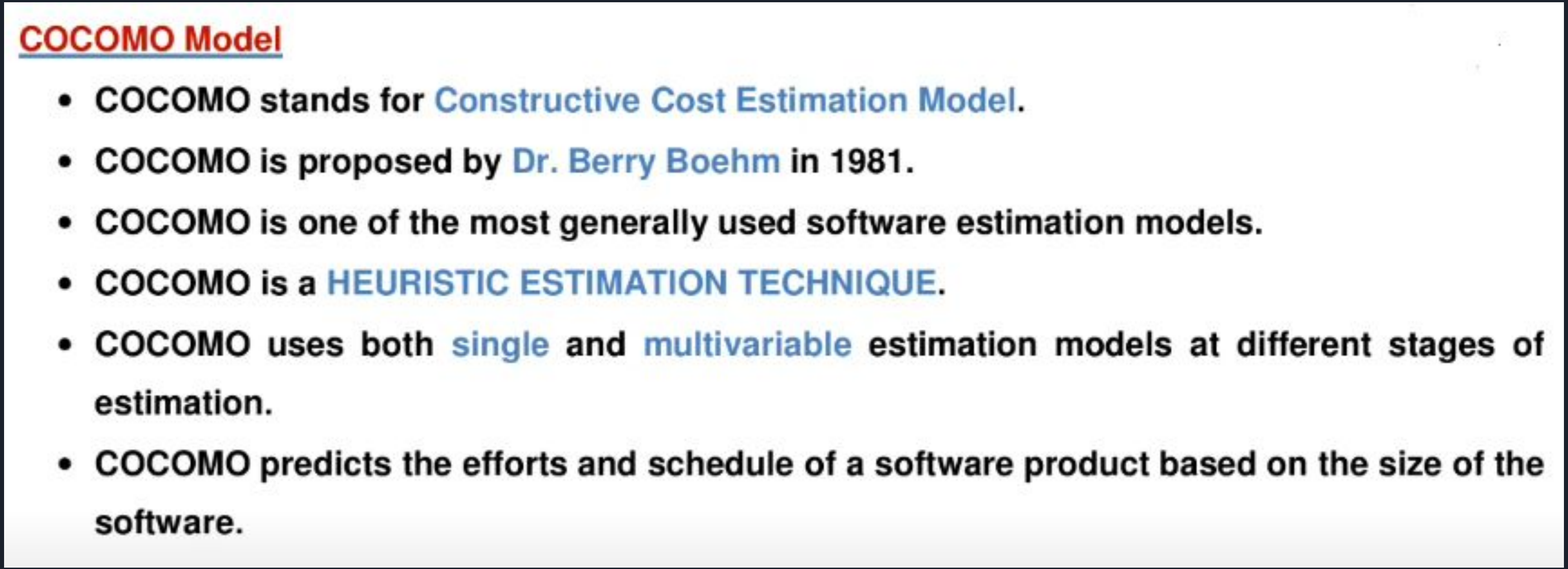
1. Single variable model:– Estimated Parameter = $c1 * e^{d1}$
   - ➢ e is independent variable
   - ➢ Estimated Parameter is dependent variable it could be effort, project duration, staff size etc
   - ➢ c1, d1 are constants (c1, d1 determined from past project)
   - ➢ Basic COCOMO model is an example.
2. Multivariable model:- Estimated Resource = $c1 * e^{d1} + c2 * e^{d2} + \ldots$
   - ➢ Multivariable estimation models are expected to give more accurate estimates compared to the single variable models
   - ➢ Intermediate COCOMO model is an example.



ANALYTICAL ESTIMATION TECHNIQUES

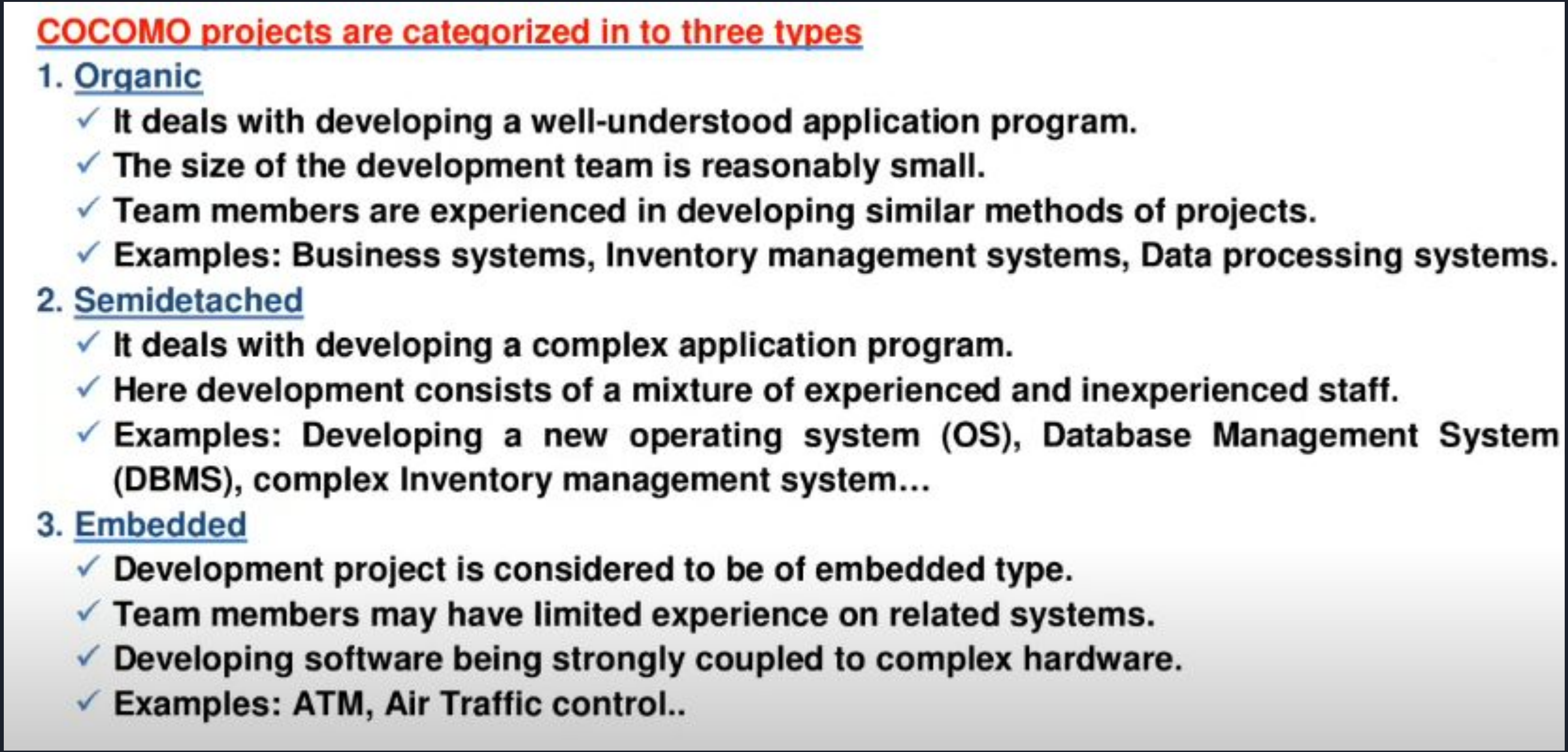
- Analytical estimation techniques derive the required results starting with certain basic assumptions regarding a project.
- Thus, unlike empirical and heuristic techniques, analytical techniques do have scientific basis.
- Halstead's software science is an example of an analytical technique.

COCOMO MODEL



COCOMO Model

- COCOMO stands for Constructive Cost Estimation Model.
- COCOMO is proposed by Dr. Berry Boehm in 1981.
- COCOMO is one of the most generally used software estimation models.
- COCOMO is a HEURISTIC ESTIMATION TECHNIQUE.
- COCOMO uses both single and multivariable estimation models at different stages of estimation.
- COCOMO predicts the efforts and schedule of a software product based on the size of the software.



COCOMO projects are categorized in to three types

1. Organic
   - ✓ It deals with developing a well-understood application program.
   - ✓ The size of the development team is reasonably small.
   - ✓ Team members are experienced in developing similar methods of projects.
   - ✓ Examples: Business systems, Inventory management systems, Data processing systems.
2. Semidetached
   - ✓ It deals with developing a complex application program.
   - ✓ Here development consists of a mixture of experienced and inexperienced staff.
   - ✓ Examples: Developing a new operating system (OS), Database Management System (DBMS), complex Inventory management system…
3. Embedded
   - ✓ Development project is considered to be of embedded type.
   - ✓ Team members may have limited experience on related systems.
   - ✓ Developing software being strongly coupled to complex hardware.
   - ✓ Examples: ATM, Air Traffic control..

Three stages of COCOMO estimation technique are

- ✓ Basic COCOMO
- ✓ Intermediate COCOMO and
- ✓ Complete COCOMO

1. Basic COCOMO Model: The basic COCOMO model provides an accurate size of the project parameters. The following expressions give the basic COCOMO estimation model.

$$\text{Effort} = a1 * (\text{KLOC})^{a2}\ \text{PM}$$
$$\text{Tdev} = b1 * (\text{efforts})^{b2}\ \text{Months}$$

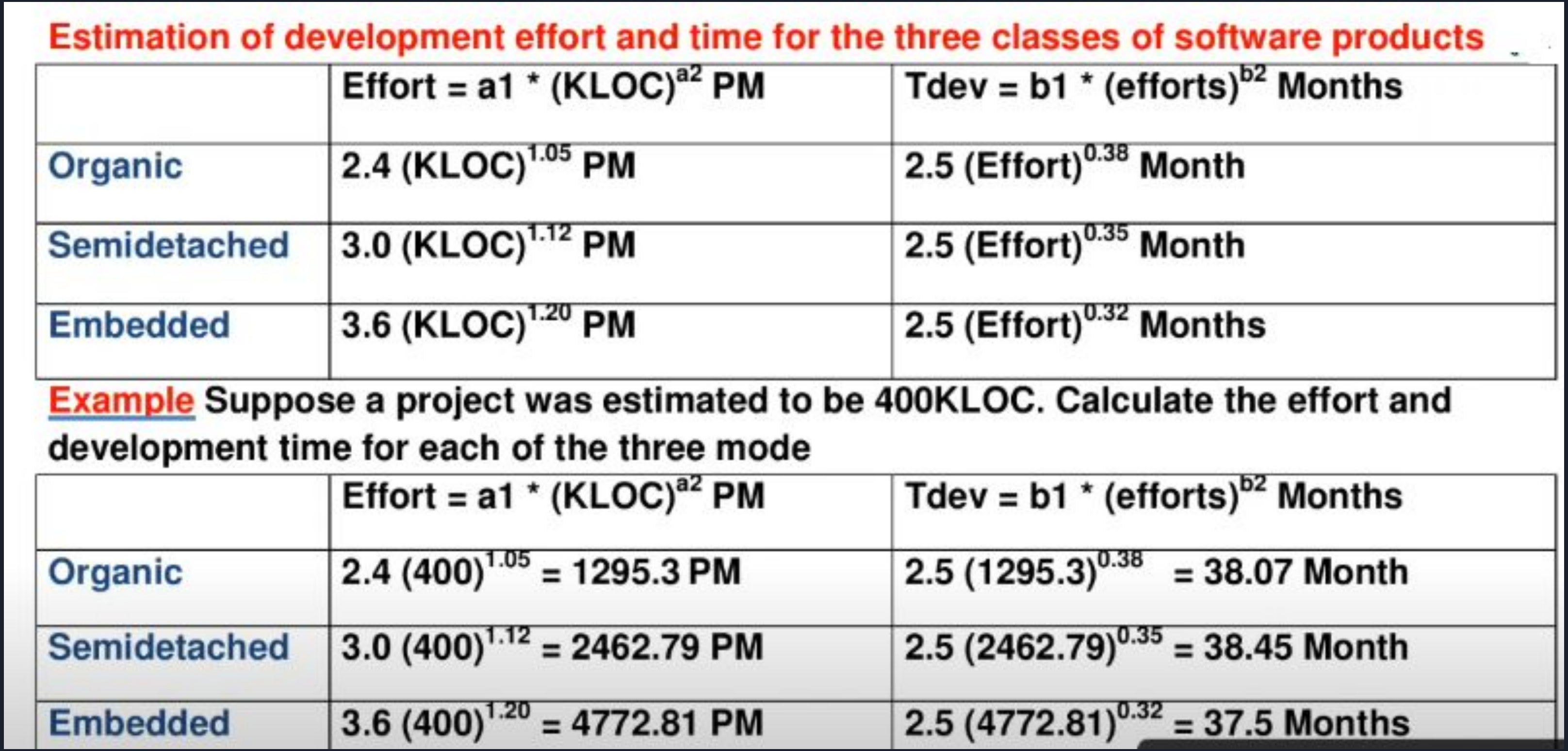
KLOC - size of the software product expressed in Kilo Lines Of Code.

a1, a2, b1, b2 are constants for each category of software product.

PM - Person-month is a popular unit for effort measurement.

Effort - is the total effort required to develop the software product, expressed in person-months (PMs).

Tdev - is the estimated time to develop the software, expressed in months.



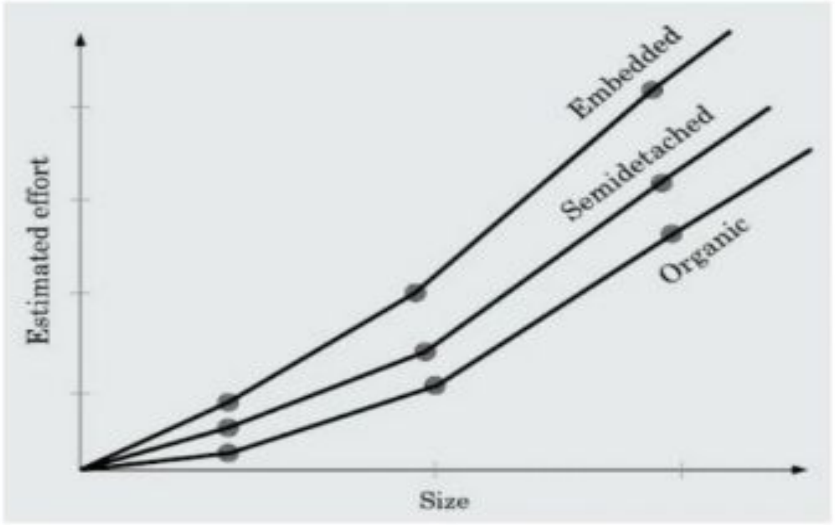
Estimation of development effort and time for the three classes of software products

	Effort = a1 * $(KLOC)^{a2}$ PM	Tdev = b1 * $(efforts)^{b2}$ Months
Organic	2.4 $(KLOC)^{1.05}$ PM	2.5 $(Effort)^{0.38}$ Month
Semidetached	3.0 $(KLOC)^{1.12}$ PM	2.5 $(Effort)^{0.35}$ Month
Embedded	3.6 $(KLOC)^{1.20}$ PM	2.5 $(Effort)^{0.32}$ Months

Example Suppose a project was estimated to be 400KLOC. Calculate the effort and development time for each of the three mode

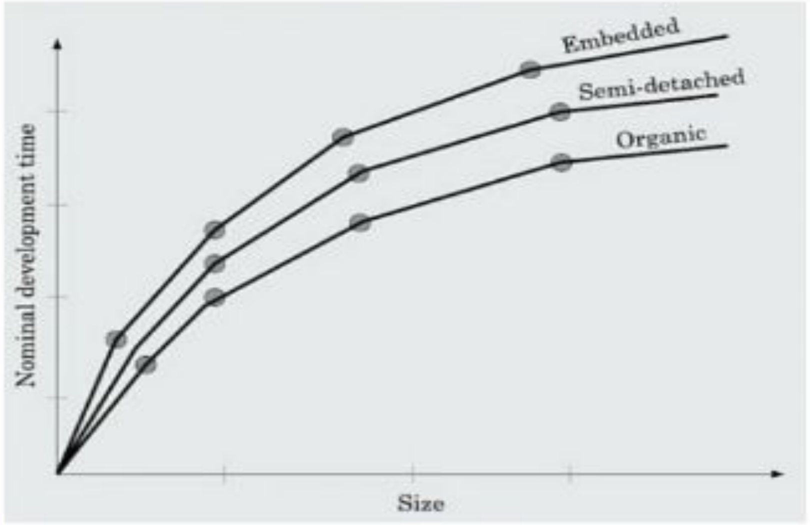
	Effort = a1 * $(KLOC)^{a2}$ PM	Tdev = b1 * $(efforts)^{b2}$ Months
Organic	2.4 $(400)^{1.05}$ = 1295.3 PM	2.5 $(1295.3)^{0.38}$ = 38.07 Month
Semidetached	3.0 $(400)^{1.12}$ = 2462.79 PM	2.5 $(2462.79)^{0.35}$ = 38.45 Month
Embedded	3.6 $(400)^{1.20}$ = 4772.81 PM	2.5 $(4772.81)^{0.32}$ = 37.5 Months

Estimated Effort Vs Product size



Effort is superliner in the size of the software product. Thus, the effort required to develop a product increases very rapidly with project size.

Estimated Time Vs Product size



Development time is a sub linear function of the size of the product. When the size of the product increases by two times, the time to develop the product does not double but rises moderately.