Planning a Software Project

Topics to be covered

- Introduction to Project Planning
- Software Cost Estimation
  - Cost Estimation Models
  - Software Size Metrics
  - Empirical Estimation
  - Heuristic Estimation
  - COCOMO
  - Effort estimation

Software Project

- Goal: Build a software system to meet commitments on cost, schedule, quality
- Worldwide - many projects fail
  - one-third are runaways with cost or schedule

Project Failures

- Major reasons for project runaways
  - unclear objectives
  - bad planning
  - no project management methodology
  - new technology
  - insufficient staff
- All of these relate to project management
- Effective project management is key to successfully executing a project

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,
- Project cost estimation,
- Scheduling,
- Project staffing,
- Project monitoring and control,
- Software configuration management,
- Risk management,
- Managerial report writing and presentations, etc.

Introduction

- A project manager's activities are varied.
  - can be broadly classified into:
    - project planning,
    - project monitoring and control activities.
    - closure

Project Planning

- Once a project is found to be feasible,
  - project managers undertake project planning.

Project Planning Activities

- Estimation:
  - Effort, cost, resource, and project duration
- Project scheduling
- Staff organization:
  - staffing plans
- Risk handling:
  - identification, analysis, and abatement procedures
- Miscellaneous plans:
  - quality assurance plan, configuration management plan, etc.

Sliding Window Planning

- Involves project planning over several stages:
  - protects managers from making big commitments too early.
  - More information becomes available as project progresses.
    - Facilitates accurate planning

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,Special Resources)
- Schedules (Work Breakdown Structure,Task Network, 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)

Why improve PM?

- Better predictability leading to commitments that can be met
- Lower cost through reduced rework, better resource mgmt, better planning,..
- Improved quality through proper quality planning and control
- Better control through change control, CM, monitoring etc.

The Project Mgmt Process

- Has three phases - planning, monitoring and control, and closure
- Planning is done before the much of the engineering process (life cycle, LC) and closure after the process
- Monitoring phase is in parallel with LC
- We focus on planning; monitoring covered through its planning

Project Planning

- Basic objective: To create a plan to meet the commitments of the project, I.e. create a path that, if followed, will lead to a successful project
- Planning involves defining the LC process to be followed, estimates, detailed schedule, plan for quality, etc.
- Main output - a project management plan and the project schedule

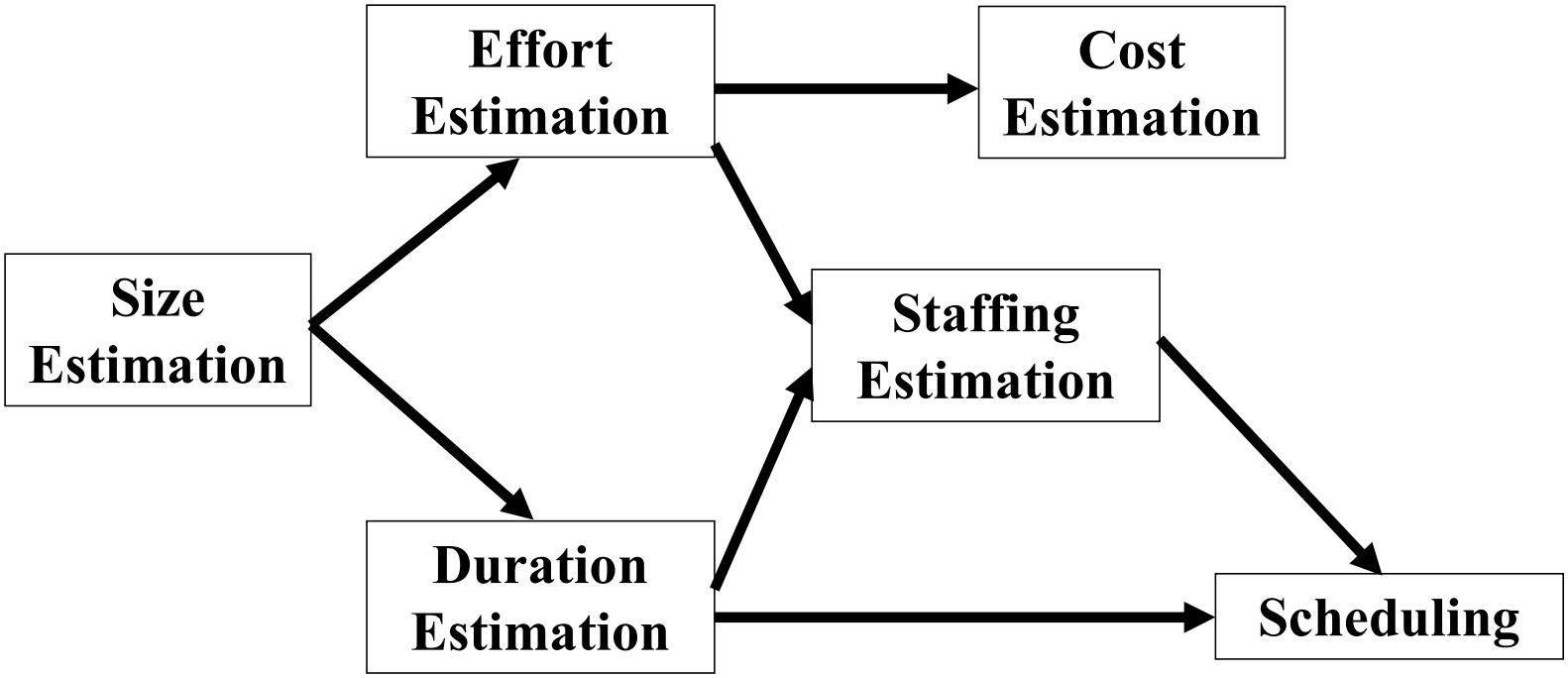
Key Planning Tasks

- Estimate effort
- Define project milestones and create a schedule
- Define quality objectives and a quality plan
- Identify risks and make plans to mitigate them
- Define measurement plan, project-tracking procedures, training plan, team organization, etc.

Software Cost Estimation

- Determine size of the product.
- From the size estimate,
  - determine the effort needed.
- From the effort estimate,
  - determine project duration, and cost.

Software Cost Estimation



Software Cost Estimation

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

Software Cost Estimation Techniques

- 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.

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.

Top down estimation

- First determines the total effort, then effort for components
- Simple approach – estimate effort from size and productivity
  - Get the estimate of the total size of the software
  - Estimate project productivity using past data and project characteristics
  - Obtain the overall effort estimate from productivity and size estimates
- Effort distribution data from similar project are used to estimate effort for different phases

Top-down Estimation

- A better method is to have effort estimate as a function of size using:

  $$\text{Effort} = a * \text{size}^{b}$$

- E is in person-months, size in KLOC
- Incorporates the observation that productivity can dip with increased size
- Constants a and b determined through regression analysis of past project data

Bottom-up Estimation

- An alternate approach to top-down
- Effort for components and phases first estimated, then the total
- Can use activity based costing - all activities enumerated and then each activity estimated separately
- Can group activities into classes - their effort estimate from past data

An Estimation Procedure

- Identify programs in the system and classify them as simple, medium, or complex (S/M/C)
- Define the average coding effort for S/M/C
- Get the total coding effort.
- Use the effort distribution in similar projects to estimate effort for other tasks and total
- Refine the estimates based on project specific factors

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.

Disadvantages of Using LOC (cont…)

- 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 (FPA)

- **Function Point Analysis (FPA) technique** quantifies the functions contained within software in terms that are meaningful to the software users. FPs consider the number of functions being developed based on the requirements specification.

Function Point Metric

- Overcomes some of the shortcomings of the LOC metric
- Developed by Allan J. Albrecht in 1979 at IBM
- further modified by the International Function Point Users Group (IFPUG).
- The functional size of the product is measured in terms of the function point

- There are two types of functions −
  - Data Functions
  - Transaction Functions
- Data Functions
  - Internal Logical Files
  - External Interface Files
- Transaction Functions
  - External Inputs
  - External Outputs
  - External Inquiries

Function Point Analysis

- 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).

Count the number of functions of each proposed type

- Various functions used in an application can be put under five types
  - #inputs, #Outputs, inquiries, #files, #interfaces
- 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.
- Files:
  - Files are logically related data and thus can be data structures or physical files.
- Interface:
  - Data transfer to other systems.

Compute the Unadjusted Function Points(UFP):

Weights of 5-FP Attributes

All these parameters are then individually assessed for complexity

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 →						

Find Total Degree of Influence

- Fourteen processing complexity factors are:
  - 4, 1, 0, 3, 3, 5, 4, 4, 3, 3, 2, 2, 4, 5.
  - $TDI=\Sigma(f_i)]$

14 Factors in Function Point Analysis Complexity

1. Will the application use data communications?
2. Are data or functions distributed?
3. Are there specific performance objectives that must be met?
4. Will the application run on a heavily used configuration requiring special design considerations?
5. Will the transaction rate of the application be high?
6. Will there be on-line data entry?
7. Will the application be designed for end-user efficiency?
8. Will there be on-line updates?
9. Is complex processing logic involved?
10. Is there an intent to provide usability for other applications?
11. How important are installation ease and conversion?
12. How important is operational ease?
13. Will the application be accessed from multiple sites?
14. Is there an intent with the design to facilitate change?

Compute Value Adjustment Factor(VAF):

- VAF = (TDI * 0.01) + 0.65

Find the Function Point Count:

- FPC = UFP * VAF

Example: Compute the function point, productivity, documentation, cost per function for the following data

1. Number of user inputs = 24
2. Number of user outputs = 46
3. Number of inquiries = 8
4. Number of files = 4
5. Number of external interfaces = 2
6. Effort = 36.9 p-m
7. Technical documents = 265 pages
8. User documents = 122 pages
9. Average monthly salary of each software developer=$209.86

- 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

- TDI=sum of all f_i (i ← 1 to 14) = 4 + 1 + 0 + 3 + 5 + 4 + 4 + 3 + 3 + 2 + 2 + 4 + 5 = 43
- FP = UFP * [0.65 + 0.01 *∑(f_i)]
  = 378 * [0.65 + 0.01 * 43]
  = 378 * [0.65 + 0.43]
  = 378 * 1.08 = 408

$$\text{Productivity} = \frac{\text{FP}}{\text{Effort}} = \frac{408}{36.9} = 11.1$$

Total pages of documentation =
technical document + user document
= 265 + 122 = 387pages

Documentation = Pages of documentation/FP
= 387/408 = 0.94

Cost = effort x average salary
= 36.9 x 209.86= $7744 per Person month

$$\text{Cost per function} = \frac{\text{cost}}{\text{productivity}} = \frac{7744}{11.1} = \$700$$

Difference between FP and LOC

FP	LOC
1. FP is specification based.	1. LOC is an analogy based.
2. FP is language independent.	2. LOC is language dependent.
3. FP is user-oriented.	3. LOC is design-oriented.
4. It is extendible to LOC.	4. It is convertible to FP

Function Point Metric (CONT.)

- It is not good for real time systems and embedded systems.
- Many cost estimation models like COCOMO uses LOC and hence FPC must be converted to LOC.
- Suffers from a major drawback:
  - the size of a function is considered to be independent of its complexity.
- Extend function point metric:
  - Feature Point metric:
  - considers an extra parameter:
    - Algorithm Complexity.

Function Point Metric (CONT.)

- Proponents claim:
  - FP is language independent.
  - Size can be easily derived from problem description
- Opponents claim:
  - it is subjective --- Different people can come up with different estimates for the same problem.

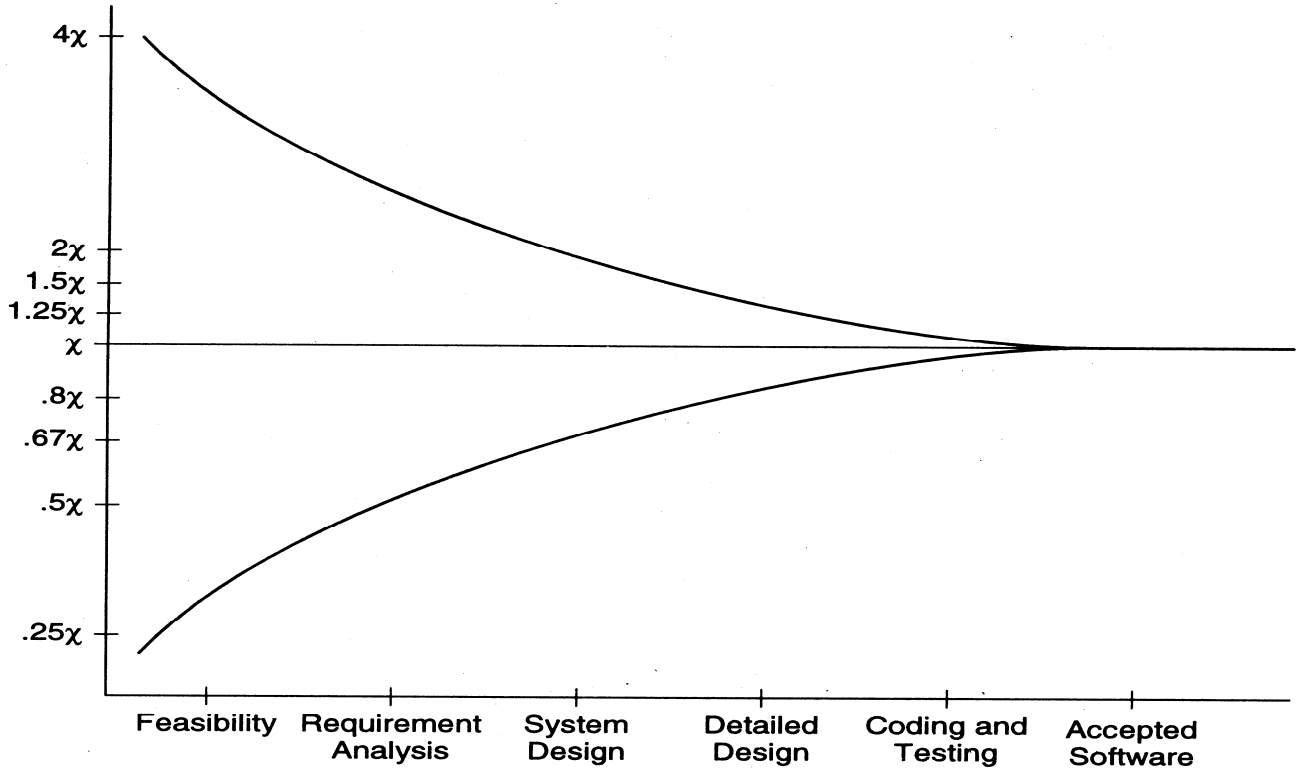
Effort Estimation

- For a project total cost and duration has to be committed in start
- Requires effort estimation, often in terms of person-months
- Effort estimate is key to planning - schedule, cost, resources depend on it
- Many problems in project execution stem from improper estimation

Estimation..

- No easy way, no silver bullet
- Estimation accuracy can improve with more information about the project
- Early estimates are more likely to be inaccurate than later
  - More uncertainties in the start
  - With more info, estimation becomes easier

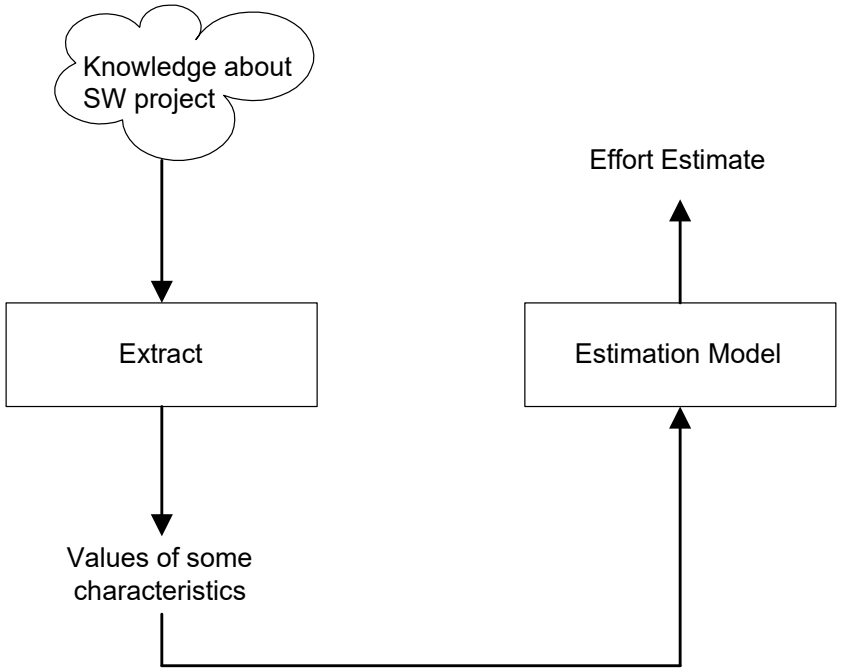
Estimation accuracy



Effort Estimation Models..

- A model tries to determine the effort estimate from some parameter values
- A model also requires input about the project, and cannot work in vacuum
- So to apply a model, we should be able to extract properties about the system
- Two types of models - top-down and bottom-up

Effort Estimation Models



COCOMO Model

- **Cocomo** (Constructive Cost **Model**) is a regression **model** based on LOC
- developed by a software engineer Barry Boehm in 1981.
- The COCOMO estimates the cost for software product development in terms of effort (resources required to complete the project work) and schedule (time required to complete the project work) based on the size of the software product.
- It estimates the required number of Man-Months (MM) for the full development of software products

COCOMO Model

- Uses size, but adjusts using some factors
- Basic procedure
  - Obtain initial estimate using size
  - Determine a set of 15 multiplying factors from different project attributes
  - Adjust the effort estimate by scaling it with the final multiplying factor

- According to COCOMO, there are three modes of software development projects that depend on complexity
  - **Organic Project**
  - **Semidetached Project**
  - **Embedded Project**

Organic Project

- It belongs to small & simple software projects which are handled by a small team with good domain knowledge and few rigid requirements.
- **Example:** Small data processing or Inventory management system.

Semidetached Project

- It is an intermediate (in terms of size and complexity) project, where the team having mixed experience (both experience & inexperience resources) to deals with rigid/nonrigid requirements.
- **Example:** Database design or OS development.

Embedded Project

- This project having a high level of complexity with a large team size by considering all sets of parameters (software, hardware and operational).
- **Example:** Banking software or Traffic light control software.

Types of COCOMO Model

- Depending upon the complexity of the project the COCOMO has three types
- **The Basic COCOMO**
- **The Intermediate COCOMO**
- **The Detailed COCOMO**

The Basic COCOMO

- Static model to estimates software development effort quickly and roughly.
- It mainly deals with the number of lines of code
- the level of estimation accuracy is less as we don't consider the all parameters belongs to the project.
- The estimated effort and scheduled time for the project are given by the relation:
  - Effort (E) = $a*(KLOC)^b$ Man-Months (MM)
    Scheduled Time (D) = $c*(E)^d$ Months(M)
  - Where,
    - **E =** Total effort required for the project in Man-Months (MM).
    - **D =** Total time required for project development in Months (M).
    - **KLOC =** the size of the code for the project in Kilo lines of code.
    - **a, b, c, d =** The constant parameters for a software project.

PROJECT TYPE	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semidetached	3	1.12	2.5	0.35
Embedded	3.6	1.2	2.5	0.32

Example

- For a given project was estimated with a size of 300 KLOC.
- Calculate the Effort, Scheduled time for development.
- Also, calculate the Average resource size and Productivity of the software for Organic project type.

For Organic: Answer

- Effort (E) = $a*(KLOC)^b = 2.4*(300)^{1.05} = 957.61$ MM
- Scheduled Time (D) = $c*(E)^d = 2.5*(957.61)^{0.38} = 33.95$ Months(M)
- Avg. Resource Size = E/D = 957.61/33.95 = 28.21 Mans
- Productivity of Software = KLOC/E = 300/957.61 = 0.3132 KLOC/MM = 313 LOC/MM

For Semidetached: Answer

- Effort (E) = $a*(KLOC)^b = 3.0*(300)^{1.12} = 1784.42$ MM
- Scheduled Time (D) = $c*(E)^d = 2.5*(1784.42)^{0.35} = 34.35$ Months(M)

For Embedded: Answer

- Effort (E) = $a*(KLOC)^b = 3.6*(300)^{1.2} = 3379.46$ MM
- Scheduled Time (D) = $c*(E)^d = 2.5*(3379.46)^{0.32} = 33.66$ Months(M)

The Intermediate COCOMO

- The intermediate model estimates software development effort in terms of
  - size of the program and
  - other related cost drivers parameters (product parameter, hardware parameter, resource parameter, and project parameter) of the project.

The Intermediate COCOMO

- The estimated effort and scheduled time are given by the relationship:
- Effort (E) = $a*(KLOC)^b*EAF$ MM
  Scheduled Time (D) = $c*(E)^d$ Months(M)
- Where,
- **E =** Total effort required for the project in Man-Months (MM).
- **D =** Total time required for project development in Months (M).
- **KLOC =** The size of the code for the project in Kilo lines of code.
- **a, b, c, d =** The constant parameters for the software project.
- **EAF =** It is an Effort Adjustment Factor, which is calculated by multiplying the parameter values of different cost driver parameters. For ideal, the value is 1.

Product Parameter

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Required Software	0.75	0.88		1.15	1.4
Size of Project Database	NA	0.94	1	1.08	1.16
Complexity of The Project	0.7	0.85		1.15	1.3

Hardware Parameter

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Performance Restriction	NA	NA	1	1.11	1.3
Memory Restriction	NA	NA		1.06	1.21
virtual Machine Environment	NA	0.87		1.15	1.3
Required Turnabout Time	NA	0.94		1.07	1.15

Personnel Parameter

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Analysis Capability	1.46	1.19		0.86	0.71
Application Experience	1.29	1.13		0.91	0.82
Software Engineer Capability	1.42	1.17	1	0.86	0.7
Virtual Machine Experience	1.21	1.1		0.9	NA
Programming Experience	1.14	1.07		0.95	NA

Project Parameter

COST DRIVERS PARAMETERS	VERY LOW	LOW	NOMINAL	HIGH	VERY HIGH
Software Engineering Methods	1.24	1.1	1	0.91	0.82
Use of Software Tools	1.24	1.1		0.91	0.83
Development Time	1.23	1.08		1.04	1.1

Example

- For a given project was estimated with a size of 300 KLOC.
- Calculate the Effort, Scheduled time for development by considering developer having high application experience and very low experience in programming.

Ans:

- Given the estimated size of the project is: 300 KLOC
- Developer having highly application experience: 0.82 (as per above table)
- Developer having very low experience in programming: 1.14(as per above table)
- EAF = 0.82*1.14 = 0.9348
  Effort (E) = $a*(KLOC)^{b}$ *EAF = $3.0*(300)^{1.12}$ *0.9348 = 1668.07 MM
  Scheduled Time (D) = $c*(E)^{d}$ = $2.5*(1668.07)^{0.35}$ = 33.55 Months(M)

The Detailed COCOMO

- It is the advanced model that estimates the software development effort like Intermediate COCOMO in each stage of the software development life cycle process.

Advantages and Disadvantages of COCOMO Model

- **Advantages**
  - Easy to estimate the total cost of the project.
  - Easy to implement with various factors.
  - Provide ideas about historical projects.
- **Disadvantages**
  - It ignores requirements, customer skills, and hardware issues.
  - It limits the accuracy of the software costs.
  - It mostly depends on time factors.

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.