CSE- 321
Software Engineering

Software Cost Estimation

Software Cost Estimation

Software Cost Estimation

Estimation is the process of **finding an estimate, or approximation**, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable.

Estimation determines **how much money, effort, resources, and time** it will take to build a specific system or product.

Estimation is based on –

1. Past Data/Past Experience
2. Available Documents/Knowledge
3. Assumptions
4. Identified Risks

Fundamental estimation questions

1. How much effort is required to complete an activity?
2. How much calendar time is needed to complete an activity?
3. What is the total cost of an activity?
4. Project estimation and scheduling and interleaved management activities

Software Cost Estimation

Software Cost Estimation

For any new software project, it is necessary to know how much it will cost to develop and how much development time will it take. These estimates are needed before development is initiated, but how is this done?

Several estimation procedures have been developed and are having the **following attributes in common.**

- Project scope must be established in **advanced**.
- Software metrics are used as a **support from** which evaluation is made.
- The project is **broken into small PCs** which are estimated individually. To achieve true cost & schedule estimate, several option arise.
- **Delay estimation**
- Used symbol **decomposition techniques** to generate project cost and schedule estimates.
- Acquire one or more **automated estimation tools**.

Software cost components

Software cost components

- Hardware and software costs
- Travel and training costs
- Effort costs (the dominant factor in most projects)
  - salaries of engineers involved in the project
  - Social and insurance costs

This following costs are all part of the total effort cost:

- Costs of providing, heating and lighting office space
- Costs of support staff such as accountants, administrators, system managers, cleaners and technicians
- Costs of networking and communications
- Costs of central facilities such as a library or recreational facilities
- Costs of Social Security and employee benefits such as pensions and health insurance.

Factors that affecting software pricing

Factor	Description
Market opportunity	A development organisation may quote a low price because it wishes to move into a new segment of the software market. Accepting a low profit on one project may give the opportunity of more profit later. The experience gained may allow new products to be developed.
Cost estimate uncertainty	If an organisation is unsure of its cost estimate, it may increase its price by some contingency over and above its normal profit.
Contractual terms	A customer may be willing to allow the developer to retain ownership of the source code and reuse it in other projects. The price charged may then be less than if the software source code is handed over to the customer.
Requirements volatility	If the requirements are likely to change, an organisation may lower its price to win a contract. After the contract is awarded, high prices may be charged for changes to the requirements.
Financial health	Developers in financial difficulty may lower their price to gain a contract. It is better to make a small profit or break even than to go out of business.

Estimation techniques

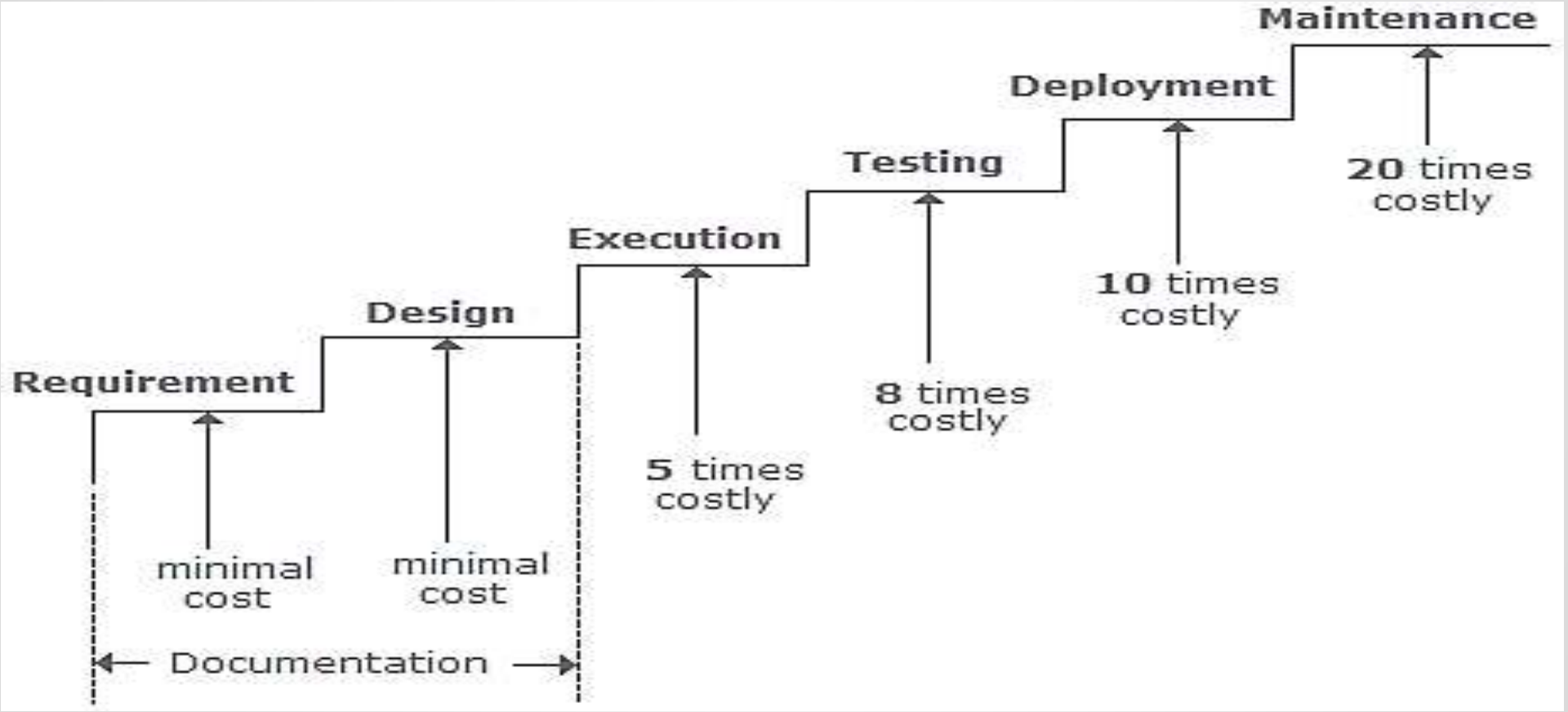
- There is no simple way to make an accurate estimate of the effort required to develop a software system
  - Initial estimates are based on inadequate information in a user requirements definition
  - The software may run on unfamiliar computers or use new technology
  - The people in the project may be unknown
- Project cost estimates may be self-fulfilling
  - The estimate defines the budget and the product is adjusted to meet the budget

Estimation techniques

- Algorithmic cost modelling
- Expert judgement
- Estimation by analogy
- Parkinson's Law
- Pricing to win

Estimation techniques error effects

A defect, which could have been removed during the initial stage, is removed in a later stage. How does this affect the cost?



COCOMO model

- LOC: Lines of Code
- Function Point
- Cost Estimation
- COCOMO I
  - Basic COCOMO
  - Intermediate COCOMO
  - Detailed COCOMO
- COCOMO II

LINES OF CODE (LOC)

- **Lines of code(LOC)** or **Source lines of code (SLOC)**is a software metric used to measure the **size of a software program** by counting the number of lines in the text of the program's source code.

- LOC is typically used **to predict the amount of effort that will be required to develop a program**, as well as to estimate programming productivity or effort once the software is produced.

FUNCTION POINT

A **Function Point (FP)** is a unit of measurement **to express the amount of business functionality**, an information system (as a product) provides to a user. **FPs measure software size**. They are widely accepted as an industry standard for **functional sizing**.

Constructive Cost Model(COCOMO)

- The COCOMO model is an algorithmic software cost estimation model.

- It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

- The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics.

- 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

Constructive Cost Model(COCOMO)

COCOMO model are categorized into three types:

1. Basic COCOMO
2. Intermediate COCOMO
3. Detailed COCOMO

According to COCOMO, there are three modes of software development projects that depend on complexity.

Such as:

1. Organic Project

2. Semidetached Project

3. Embedded Project

.

Development Mode of S/W Projects

Mode	Project Size	Nature of Project	Innovation	Deadline
Organic	Typically 2-50 KLOC	Small size project, Experienced developers.	Little	Not Tight
Semi Detached	Typically 50-300KLOC	Medium size project and team.	Medium	Medium
Embedded	Typically over 300KLOC	Large project, Real-time systems	Significant	Tight

KLOC is a measure of the size of a computer program. The size is determined by measuring the number of lines of source code a program has.

Constructive Cost Model(COCOMO)

1. 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.

2. 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.

3. 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.

Basic COCOMO Model

The Basic COCOMO

It is the one type of static model to estimates software development effort quickly and roughly. It mainly deals with the number of lines of code and 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) = $a*(KLOC)^b$ MM

Scheduled Time (D) = $c*(E)^d$ Months(M)

Basic COCOMO Model

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

Basic cocomo: Example-1

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.

Ans: Given estimated size of project is: 300 KLOC

For Organic

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

Basic cocomo: Example-1

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.

Ans: Given estimated size of project is: 300 KLOC

For Semidetached

Effort (E) = $a*(KLOC)^b = 3.0*(300)^{1.12} = 1784.42$ MM

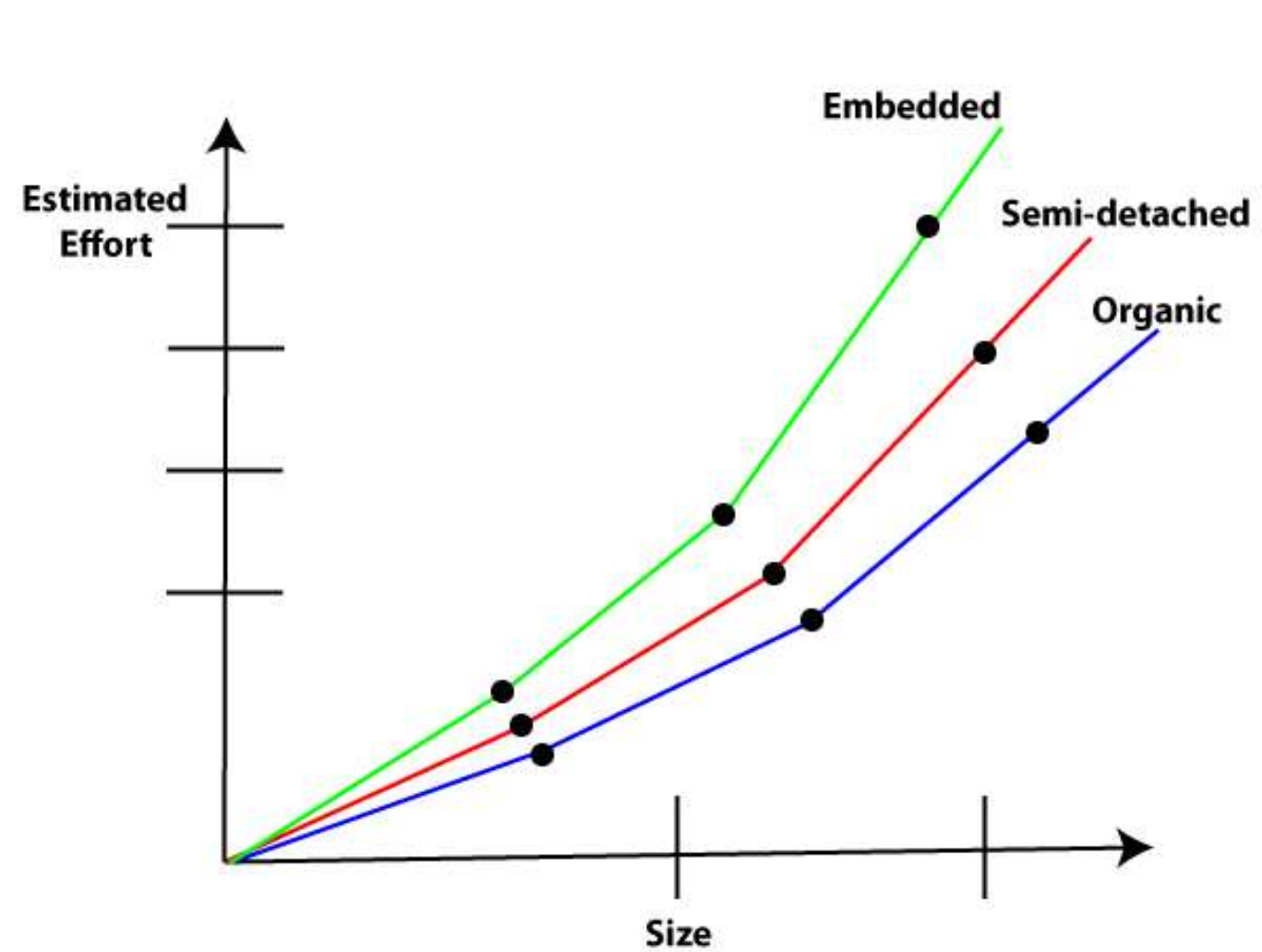
Scheduled Time (D) = $c*(E)^d = 2.5*(1784.42)^{0.35} = 34.35$ Months(M)

For Embedded

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)

development time versus the product size



Effort versus product size

https://www.javatpoint.com/cocomo-model

Basic cocomo: Example-2

Example: A project size of 200KLOC is to be developed. S/W development team has average experience on similar type of projects. The project schedule is not very tight (that means medium). Calculate the effort and development time of the project.

Ans: 200 KLOC implies semi-detached mode.

Hence,

$E = 3.0 * (200)^{1.12} = 1133.12$ MM

$D = 2.5 * (1133.12)^{0.35} = 29.3$ M

Avg. staff size(SS) = E/D = 1133.12/29.3=38.67 Persons.

Productivity (P) = KLOC/E = 200/1133.12=0.1765 KLOC/MM.

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.

Classification of Cost Drivers :

(i) Product attributes
(ii) Computer attributes
(iii) Personnel attributes
(iv) Project attributes

Intermediate COCOMO

- Extension of Basic COCOMO
- **Why Use ?**

  Basic model lacks accuracy
- Computes software development effort as a function of program size and set of 15 Cost Drivers
- **Cost Driver:** A multiplicative factor that determines the effort required to complete the software project.
- **Why Cost Drivers?**

  Adjust the nominal cost of a project to the actual project Environment.
- For each Characteristics, Estimator decides the scale factor

Cost Drivers

Product Attributes

- Required Software Reliability (RELY)
- Database Size (DATA)
- Product Complexity (CPLX)

Computer Attributes

- Execution Time Constraint (TIME)
- Main Storage constraint (STOR)
- Virtual Machine volatility (VIRT)
- Computer turnaround time (TURN)

Personnel Attributes

- Analyst Capability (ACAP)
- Application Experience (AEXP)
- Programmer Capability (PCAP)
- Virtual Machine Experience (VEXP)
- Programming language Experience (LEXP)

Project Attributes

- Modern programming practices (MODP)
- Use of Software tools (TOOL)
- Required development schedule (SCED)

The Calculation

- Multiply all 15 Cost Drivers to get **Effort Adjustment Factor(EAF)**

- **E(Effort) = a(KLOC)^{b} * EAF**(in Person-Month)
- **D(Development Time) = c(E)^{d}** (in month)
- **SS (Avg Staff Size) = E/D** (in persons)
- **P (Productivity) = KLOC/E** (in KLOC/Person-month)

Project	a	b	c	d
Organic	3.2	1.05	2.5	0.38
Semidetached	3.0	1.12	2.5	0.35
Embedded	2.8	1.20	2.5	0.32

Intermediate COCOMO : Example

A new project with estimated **400 KLOC embedded system** has to be developed. Project manager hires developers of very **low quality** but a **lot of experience** in programming language. Calculate the Effort, Development time, Staff size & Productivity.

Cost Drivers	Very Low	Low	Nominal	High	Very High	Extra High
AEXP	**1.29**	1.13	1.00	0.91	0.82	--
LEXP	1.14	1.07	1.00	**0.95**	--	--

Project	a	b	c	d
Embedded	2.8	1.20	2.5	0.32

EAF = 1.29 * 0.95 = 1.225

400 LOC implies Embedded System

Effort = $2.8*(400)^{1.20}$ * 1.225 = 3712 * 1.22 = 4528 person-months

Development Time = $2.5 * (4528)^{0.32}$ = 2.5 * 14.78 = 36.9 months

Avg. Staff Size = E/D = 4528/36.9 = 122 persons

Productivity = KLOC/Effort = 400/4528 = 0.0884 KLOC/person-month

Detailed COCOMO

Detailed COCOMO =

Intermediate COCOMO + assessment of Cost Drivers impact on each phase.

- Phases
  1) Plans and requirements
  2) System Design
  3) Detailed Design
  4) Module code and test
  5) Integrate and test

  Cost of each subsystem is estimated separately. This reduces the margin of error.

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

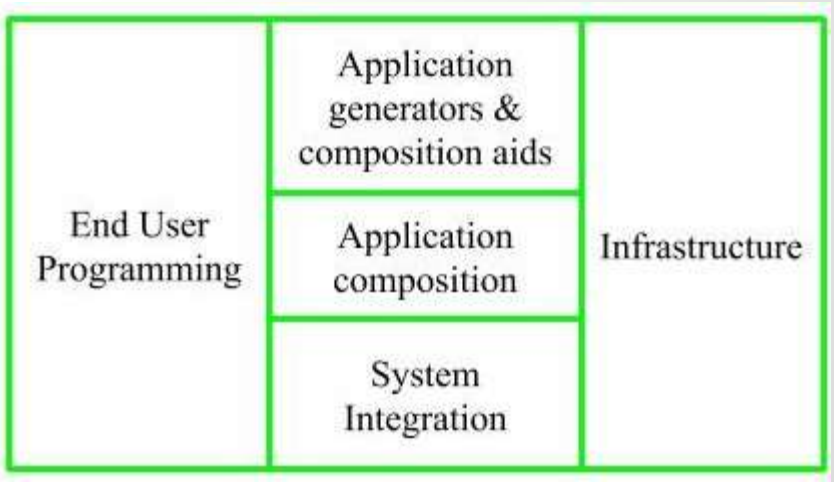
- It ignores requirements, customer skills, and hardware issues.
- It limits the accuracy of the software costs.
- It mostly depends on time factors.

WHY COCOMO-II ?

- The changes in s/w development techniques included a move away from mainframe overnight batch processing to desktop-based real-time turnaround.
- These changes and others began to make applying the original COCOMO model problematic.
- The model is tuned to the life cycle practices of the 21st century.

COCOMO-II is the revised version of the original COCOMO (Constructive Cost Model) and is developed at **University of Southern California**. It is the model that allows one to estimate the cost, effort and schedule when planning a new software development activity.

It consists of three sub-models:



Stages Of Cocomo-II

1. Application Composition
2. Earlier Design
3. Post Architecture

Final Word

"The models are just there to help, not to make the management decisions for you."

--Barry Boehm