ESC501 (COCOMO Model & Cost Estimation)

- PROF. POULAMI DUTTA

Intended Learning Outcomes (ILOs)

- Differentiate among organic, semidetached and embedded software projects.
- Explain basic COCOMO.
- Differentiate between basic COCOMO model and intermediate COCOMO model.
- Explain the complete COCOMO model.

COCOMO Model

- COCOMO (COnstructive COst MOdel) 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:
  - The project deals with developing a well understood application.
  - The development team is small.
  - The team members have prior experience in working with similar types of projects.

- Semidetached:
  - The team consists of some experienced as well as inexperienced staff.
  - Team members may have some experience on the type of system to be developed.

- Embedded:
  - Aims to develop a software strongly related to machine hardware or real-time systems.
  - Team size is usually large.

Comparative Study of Product Classes

Comparison of three COCOMO modes

Mode	Project Size	Nature of Project	Innovation	Deadline of the Project	Development Environment
Organic	Typically 2 – 50 KLOC	Small Size Projects, experienced developers.	Little	Not tight	Familiar And In house
Semi-Detached	Typically 50 – 300 KLOC	Medium size project, average previous experience on similar projects.	Medium	Medium	Medium
Embedded	Typically over 300 KLOC	Large projects, complex interfaces, very little previous experience.	Significant	Tight	Complex Hardware / Customer interfaces required

COCOMO Model (CONT.)

- 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-months
- Boehm obtained these equations:
  - examined historical data collected from a large number of actual projects.
  - estimation of project parameters should be done through three stages: Basic COCOMO, Intermediate COCOMO, and Complete COCOMO.

COCOMO Model (CONT.)

- Software cost estimation is done through three stages:
  - Basic COCOMO,
  - Intermediate COCOMO,
  - Complete COCOMO.

Basic COCOMO Model

- The basic COCOMO model helps to obtain a rough estimate of the project parameters.
- It estimates effort and time required for development in the following way:
  - Effort = a * $(KDSI)^b$ PM
  - T_{dev} = 2.5 * $(Effort)^c$ Months
- where,
  - KDSI is the estimated size of the software expressed in Kilo Delivered Source Instructions,
  - a, b, c are constants determined by the category of software project,
  - Effort denotes the total effort required for the software development, expressed in person months (PMs),
  - T_{dev} denotes the estimated time required to develop the software (expressed in months).

Basic COCOMO Model (CONTD)….

Software project	*a*	*b*	*c*
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

Development Effort Estimation

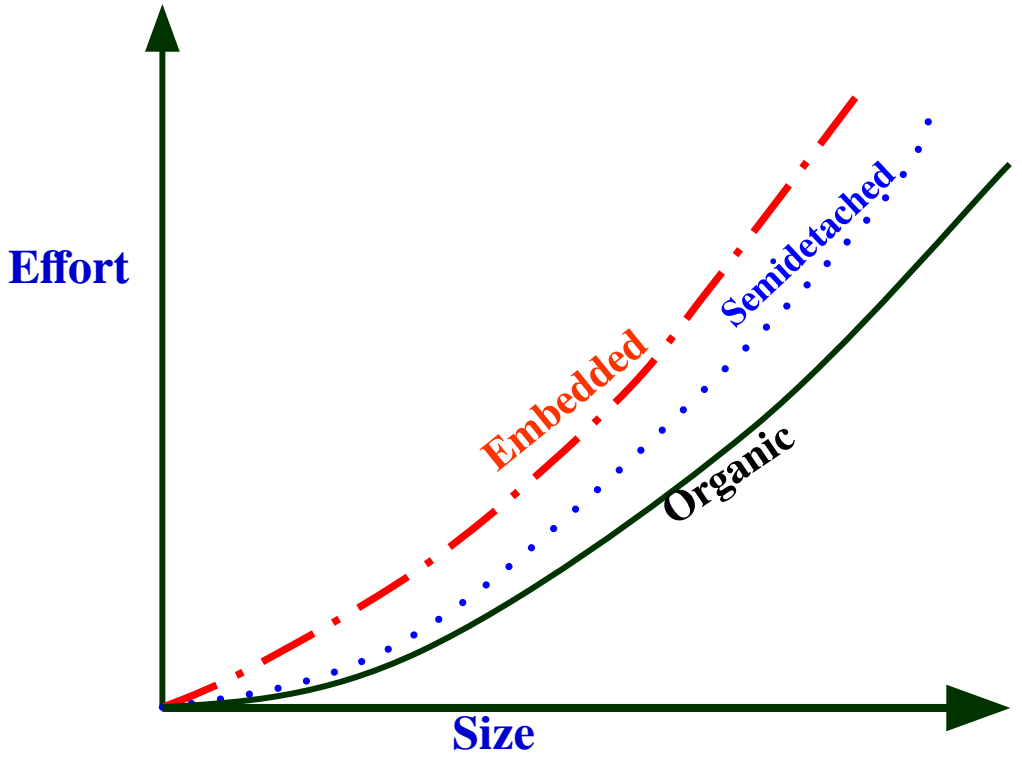
- Organic :
  - Effort = 2.4 x $(KLOC)^{1.05}$ PM.
- Semi-detached:
  - Effort = 3.0 x $(KLOC)^{1.12}$ PM.
- Embedded:
  - Effort = 3.6 x $(KLOC)^{1.20}$ PM.

Development Time Estimation

- Organic:
  - $T_{dev} = 2.5\ (Effort)^{0.38}$ Months.
- Semi-detached:
  - $T_{dev} = 2.5\ (Effort)^{0.35}$ Months.
- Embedded:
  - $T_{dev} = 2.5\ (Effort)^{0.32}$ Months.

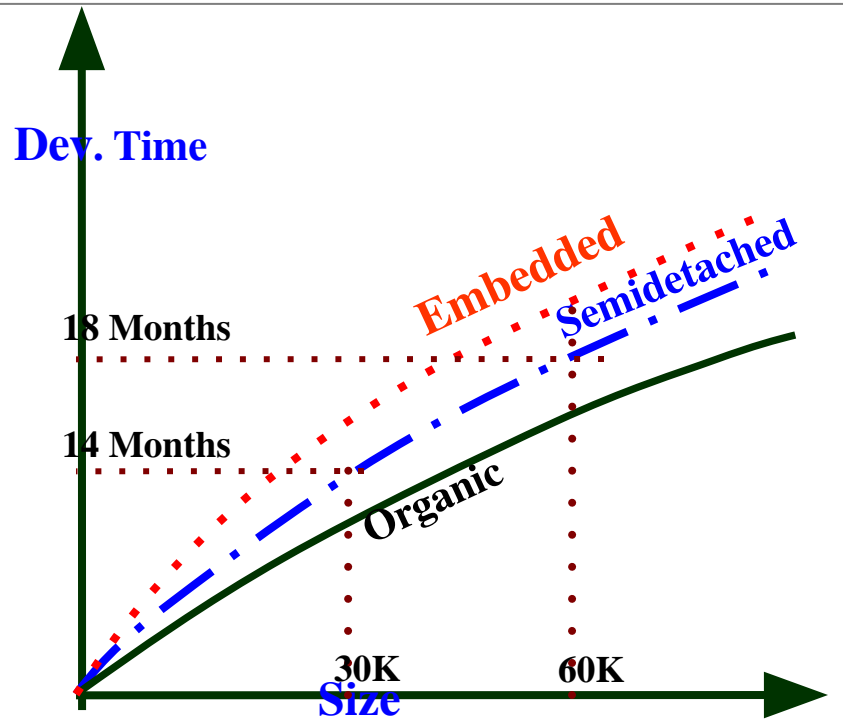
Basic COCOMO Model (CONTD)….

Effort is somewhat super-linear in problem size.



Basic COCOMO Model (CONTD)….

- 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 Model (CONTD)....

- 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 Model (CONTD)....

- Development time is roughly the same for all the three categories of products:
  - For example, a 60 KLOC program can be developed in approximately 18 months.
    - regardless of whether it is of organic, semi-detached, or embedded type.
  - There is more scope for parallel activities for system and application programs, than utility programs.
    - Effort = $2.4 \times (60)^{1.05}$ PM. = 176.713 PM.
    - Effort = $3.0 \times (60)^{1.12}$ PM. = 294.205 PM.
    - Effort = $3.6 \times (60)^{1.20}$ PM. = 489.873 PM.

    - $T_{dev} = 2.5 \times (176.713)^{0.38}$ Months. = 17.8 Months.
    - $T_{dev} = 2.5 \times (294.205)^{0.35}$ Months. = 18.27 Months.
    - $T_{dev} = 2.5 (489.873)^{0.32}$ Months. = 18.14 Months.

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 Model

- Intermediate 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 Model (CONTD)…..

- The intermediate COCOMO take those other factors into consideration by defining a set of 15 cost drivers (multipliers).
- Each of the 15 such attributes can be rated on a six-point scale ranging from "very low" to "extra high" in their relative order of importance.
- Each attribute has an effort multiplier fixed as per the rating.
- The product of effort multipliers of all the 15 attributes gives the **Effort Adjustment Factor (EAF)**.

Intermediate COCOMO Model (CONTD)……

- 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.

Intermediate COCOMO Model (CONTD)……

Cost Drivers	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
Product attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.46	1.19	1.00	0.86	0.71	
Applications experience	1.29	1.13	1.00	0.91	0.82	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Intermediate COCOMO Model (CONTD)……

- EAF is used to refine the estimates obtained by basic COCOMO as follows:
  - $\text{Effort}|_{\text{corrected}} = \text{Effort} * \text{EAF}$
  - $T_{dev}|_{\text{corrected}} = 2.5 * (\text{Effort}|_{\text{corrected}})^{c}$

Intermediate COCOMO Model (CONTD)……

Software project	a	b	c
Organic	3.2	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	2.8	1.20	0.32

Example

- The size of a software product has been estimated to be 200 KLOC. Cost drivers are as follows:
  - Low Reliability 0.88
  - High Product Complexity 1.15
  - Low Application Experience 1.13
  - High Programming Language Experience 0.95
- EAF = 0.88 * 1.15 * 1.13 * 0.95 = 1.086
- Semi-Detached:
- $\text{Effort}|_{\text{corrected}} = \text{Effort} * \text{EAF}$
  - $= 3.0 \text{ X } (200)^{1.12} \text{ X } 1.086 = 1230.56 = 1231 \text{ PM}$
- $\text{Tdev}|_{\text{corrected}} = 2.5 * (\text{Effort}|_{\text{corrected}})^{c}$
  - $= 2.5 \text{ x } (1231)^{0.35}$
  - $= 30.16 \text{ months} = 30 \text{ months}$

Shortcomings 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 Model

- 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.

Complete COCOMO Model (CONTD)

- A Management Information System (MIS) for an organization having offices at several places across the country:
  - Database part (semi-detached)
  - Graphical User Interface (GUI) part (organic)
  - Communication part (embedded)
- Costs of the components are estimated separately:
  - summed up to give the overall cost of the system.

Complete COCOMO Model (CONTD) ..

- To illustrate this, consider a very popular distributed application: the ticket booking system of the Indian Railways. There are computerized ticket counters in most of the railway stations of our country. Tickets can be booked / cancelled from any such counter. Reservations for future tickets, cancellation of reserved tickets could also be performed. On a high level, the ticket booking system has three main components:
  - Database
  - Graphical User Interface (GUI)
  - Networking facilities
- Among these, development of the GUI is considered as an organic project type; the database module could be considered as a semi-detached software. The networking module can be considered as an embedded software. To obtain a realistic cost, one should estimate the costs for each component separately, and then add it up.

Advantages & Disadvantages of COCOMO

- Easy to estimate the total cost of the project.
- Easy to implement with various factors.
- Provide ideas about historical projects.

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

COCOMO II Model

- 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.

Thank You