Software Project Management

Lecture 19

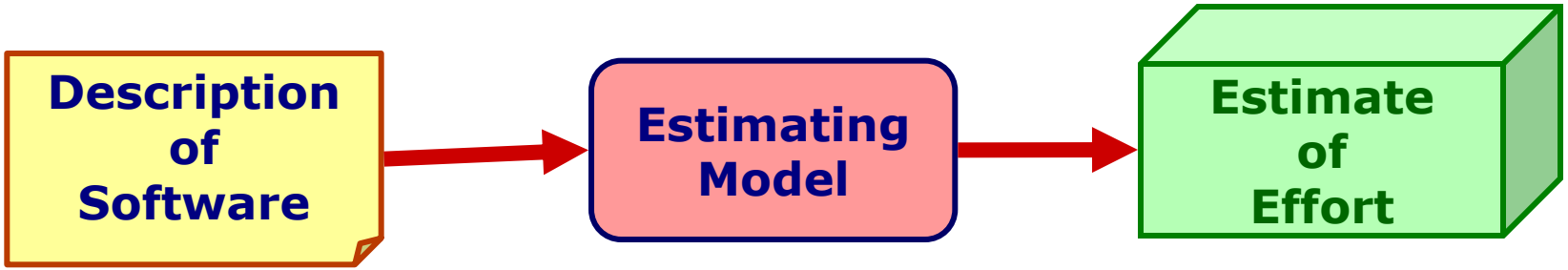
Words of Wisdom

- Finish today's work today. Learn to things quicker and better, every time.

- *Never save anything for the tomorrow. (Tirmidhiy).*

Effort Estimating Models

- Certain models can be used to predict effort, given many facts about the software
  - The principal facts are the estimated size and complexity of the software
- An Estimating Model Is ...
  - An equation or set of equations that produces an estimate of the effort, given inputs that describe the software to be written



A Very Simple Model

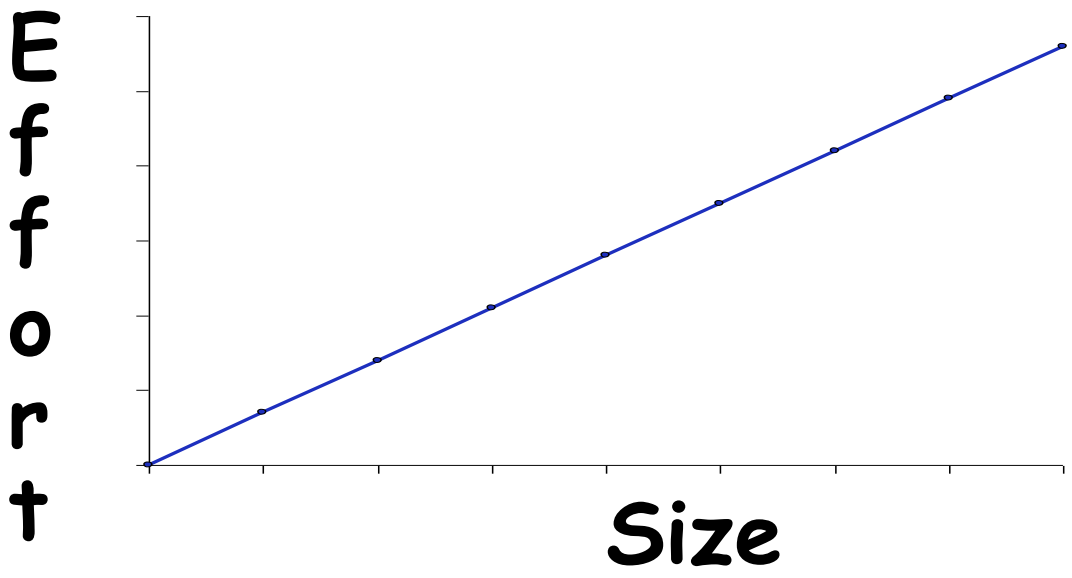
Effort = Size * *Productivity*

Examples:

- Staff Days = Lines of Code**Staff Days/LOC*
- Staff Days = Modules**Staff Days/Module*
- Staff Days = Function Points**Staff Days/FP*

A Graph of the Model

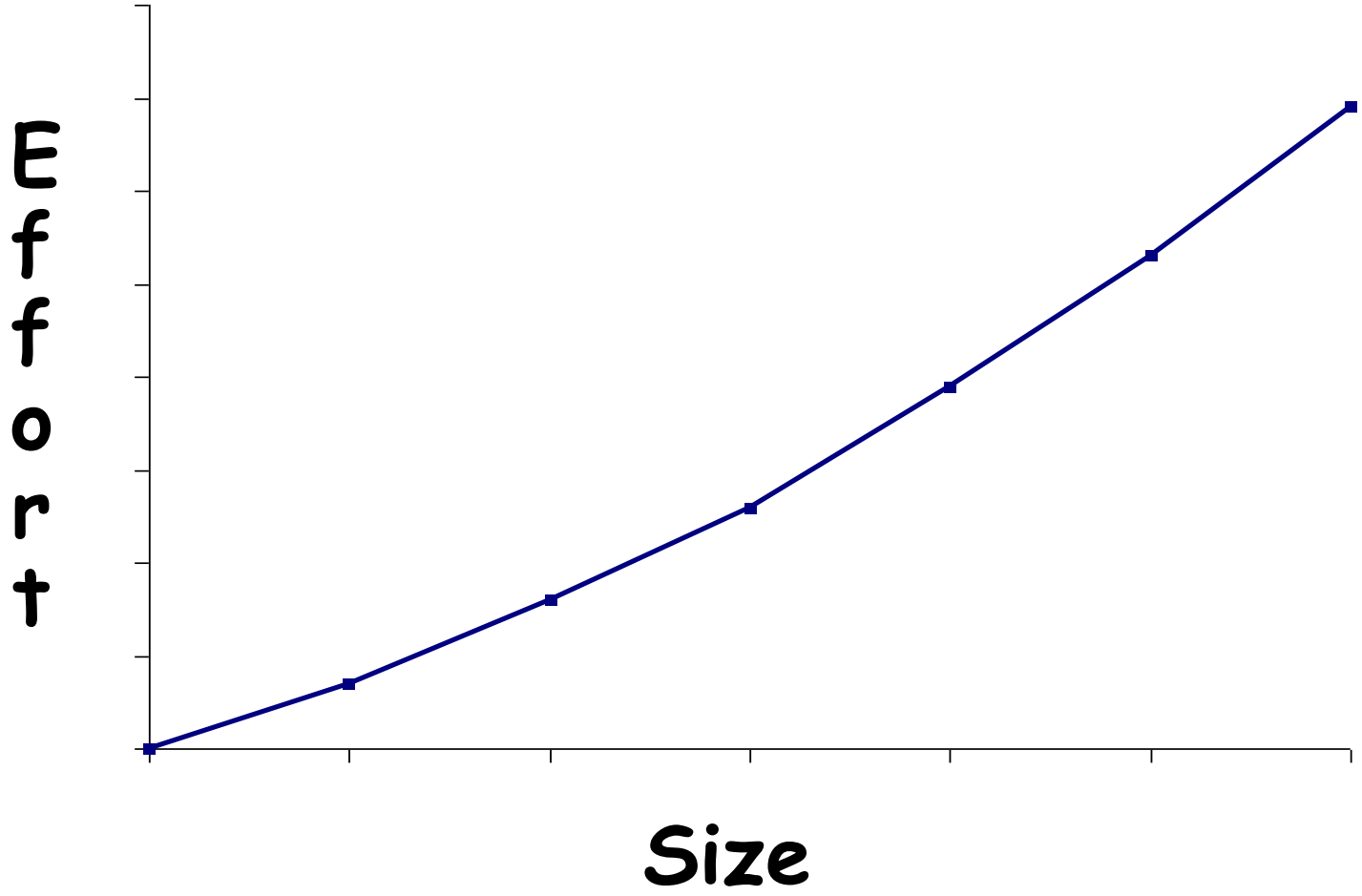
Effort = Size * *Constant*



But We Know That ...

- Effort grows faster as size increases, due to management overhead and other such factors
- So the graph ought to curve up rather than being a straight line

Perhaps Something Like This

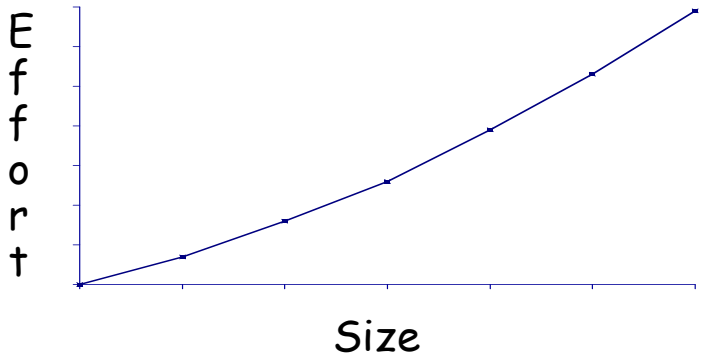


One Model That Produces Such a Curve

$$Effort = a * Size^{b}$$

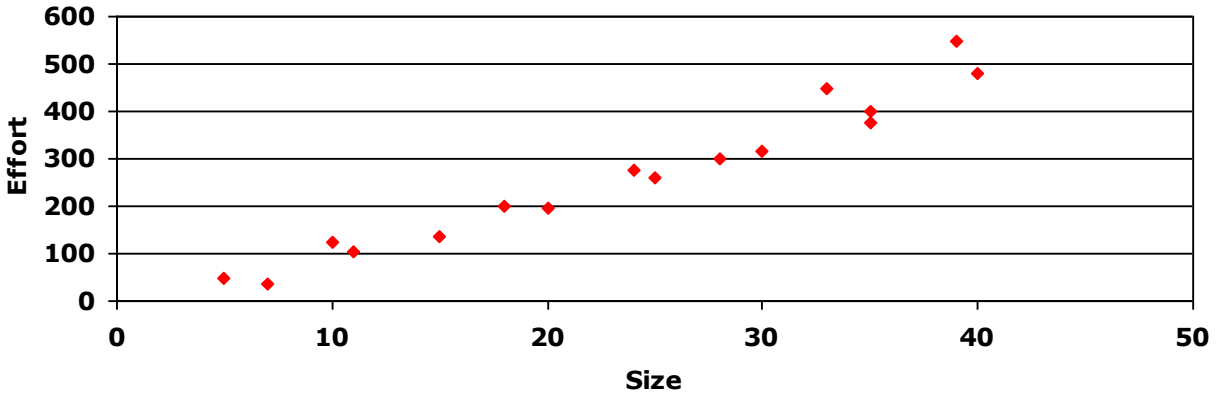
- a and b are constants that depend on the company and the type of software

Many organizations find that their software effort fits this model for effort as a function of software size



How to Determine the Curve

- By observing company experience over a period of time, you can collect enough data to determine whether this curve fits your data
- And calculate typical values of a and b
- While such models are more commonly used for large projects, there are many principles applicable to small projects as well



COCOMO

- Constructive Cost Model, developed by Barry Boehm in 1981
- Most widely used and accepted of the effort estimation models
- Predicts the effort and duration of a project based on inputs relating to the size and a number of "cost drives" that affect productivity

Original COCOMO

- Original COCOMO is a collection of three models
  - Basic
    - Applied early in the project
  - Intermediate
    - After requirements are specified
  - Advanced/Detailed
    - After a design is complete
- All three models take the form

$$Effort = a * Size^{b}$$

- Size is in *Thousands* of Lines of Code
- Effort is in Staff Months, assuming 19 staff days per staff month

Three Levels of Difficulty

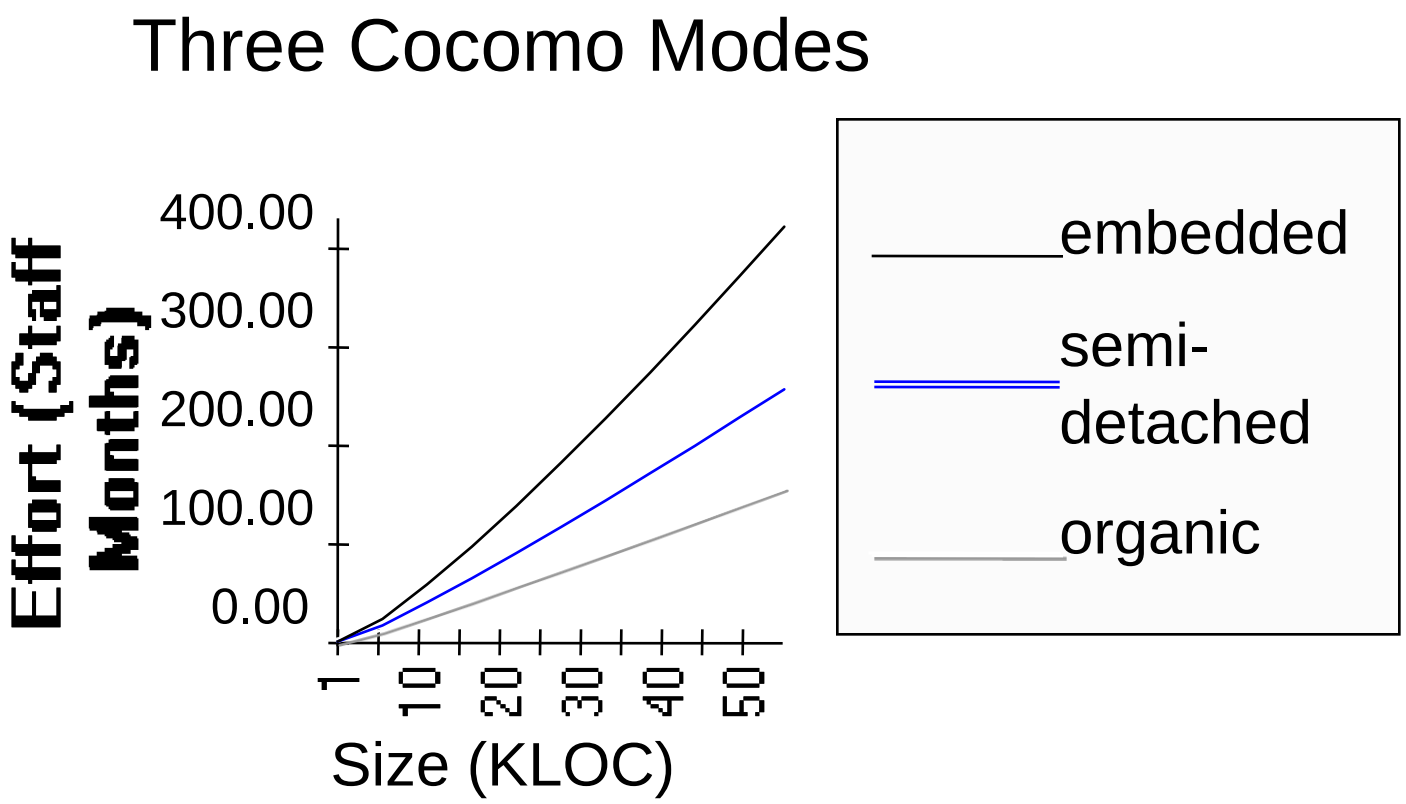
- Organic -- Fairly easy software, software development team is familiar with application, language, and tools.
  - Typically from 2-50 KLOC
- Semi-Detached -- Average software, average team, rigid requirements
  - Typically 50-300 KLOC
- Embedded -- Severe constraints, real time, etc.
  - No stated size range, but model is known to fail under 10KLOC

COCOMO Modes

- Boehm found that the calibration to COCOMO was similar within each level of difficulty.
- Thus there are three COCOMO “modes” or “standard values of a and b”

MODE	a	b
Organic	2.4	1.05
Semi-Detached	3.0	1.12
Embedded	3.6	1.20

Basic COCOMO Equation in Graph Form



Intermediate COCOMO

- Variances among programs within a given mode (level of difficulty) tend to be driven by specific **cost drivers**.
- Boehm originally identified fifteen such cost drivers.
- Others have since identified additional cost drivers.
- And specific projects often find unique situations that drive cost up or down.

Cost Drivers - I
The Nature of the Job to Be Done

- Required Reliability
  - Applies mainly to real time applications
- Data Base Size
  - Applies mainly to data processing applications
- Product Complexity
- Execution Time Constraints
  - Applies when processor speed is barely sufficient
- Main Storage Constraints
  - Applies when memory size is barely sufficient
- Virtual Machine Volatility
  - Includes hardware and operating system of target machine
- Required Reuse

Cost Drivers - II
The Practices and Tools

- Turnaround Time for Development
  - Usually not much of a problem these days
- Modern Programming Practices
  - Structured Analysis or OO
- Modern Programming Tools
  - E.g., CASE, good debuggers, test generation tools
- Schedule Compression (or Expansion)
  - Note -- deviation from ideal can never help, but shorter is worse than longer

Cost Drivers - III
The People Who Will Do It

- Analyst Capability
- Application Experience
- Programmer Capability
- Virtual Machine Experience
  - Includes operating system and hardware
- Programming Language Experience
  - Includes tools and practices

People-related drivers tend to have high potential for cost increase or decrease

Cost Drivers - IV
Additional Items Commonly Added

- Requirements Volatility
  - Some is expected -- but too much can be a big problem
- Development Machine Volatility
  - Unstable OS, compilers, CASE tools, etc.
- Security Requirements
  - for classified programs
- Access to Data
  - Sometimes very difficult
- Impact of Standards and Imposed Methods
- Impact of Physical Surroundings

Intermediate COCOMO Adjustment Factors

- Concept:
- 1) Define a set of *cost adjustment factors* [Cis] corresponding to the cost drivers. The nominal value of each Ci = 1.
- Ci = 1 implies the cost driver does not apply
- Ci > 1 implies increased cost due to this factor
- Ci < 1 implies decreased cost due to this factor

- 2) Multiply the cost adjustment factors by the basic COCOMO equation to get net cost.

Intermediate COCOMO Equation

$$\text{Effort} = \text{EAF} * a * \text{Size}^{b}$$

where EAF = $C_1 * C_2 * \ldots * C_n$

[C_i = i th cost adjustment factor]

Intermediate COCOMO Revised Values of "a"

MODE	a	b
Organic	**3.2**	**1.05**
Semi-Detached	**3.0**	**1.12**
Embedded	**2.8**	**1.20**

The "b" values are the same. The value of "a" changes because of the "effort adjustment factor" (EAF)

Cost Adjustment Factors

Cost drivers	Very Low	Low	Normal	High	Very High	Extra High
REPLY	0.75	0.88	1.00	1.15	1.40	
DATA		0.94	1.00	1.08	1.16	
CPLX	0.70	0.85	1.00	1.15	1.30	1.65
TIME			1.00	1.11	1.30	1.66
STOR			1.00	1.06	1.21	1.56
VIRT		0.87	1.00	1.15	1.30	
TURN		0.87	1.00	1.07	1.15	
ACAP		1.19	1.00	0.86	0.71	
AEXP		1.13	1.00	0.91	0.82	
PCAP		1.17	1.00	0.86	0.70	
VEXP		1.10	1.00	0.90		
LEXP		1.07	1.00	0.95		
MODP		1.10	1.00	0.91	0.82	
TOOL		1.10	1.00	0.91	0.83	
SCED		1.08	1.00	1.04	1.10	

Detailed COCOMO

- This is the most complex version of the Cocomo model. It involves the following additional steps:
  - Decompose the program into specific products and components of products
  - Analyze cost drivers separately for each component
  - Use Specific formulas for calculating cost adjustment factors, rather than “judgment”

Schedule Estimation

- COCOMO presents an equation for estimating the development schedule (Time of Develop **TDEV**) of the project in months
  - ***TDEV= c * (SM)^{d}***

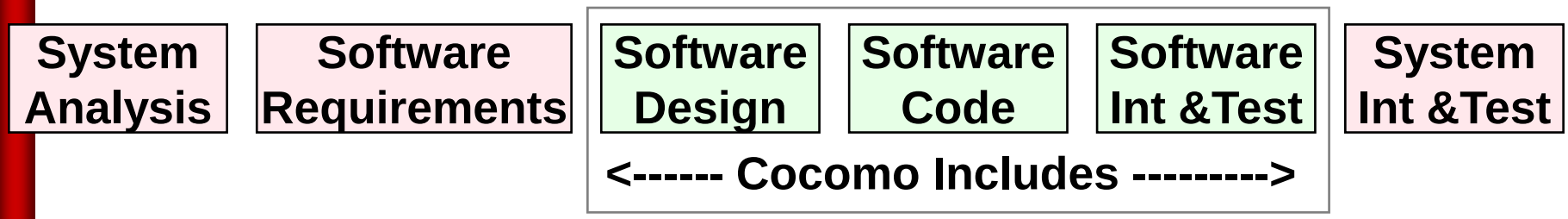
Mode	C	d
Organic	2.5	0.38
Semi-detached	2.5	0.35
Embedded	2.5	0.32

Issues With Cocomo (and Other Estimating Models)

- Amount of judgment required in determining values for cost adjustment factors and which mode applies to the software
- Your data may not match the data used to develop the model you wish to use
  - But your company may not want to collect the data needed to correlate the model
- Many models, including Cocomo, assume a basic waterfall process model
  - 30% design; 30% coding; 40% integration and test
  - Your process may not match up well

Each Model Includes Only Selected Tasks of Software Development

- For example, Cocomo does not include
  - System level analysis and design
  - Software requirements analysis or
  - Software support of system level integration and test



Q&A