Software Project Management

Lecture # 6

Outline

- Remaining topics from Chapter 23
  - Empirical Estimation Models
    - COCOMO and COCOMO II
  - The Software Equation
  - Estimating Cost
  - The Make/Buy Decision
  - Outsourcing

Empirical Estimation Models

- An estimation model for software uses empirically derived formulas.
- These formulas can predict effort as a function of LOC or FP.
- Empirical data that support most estimation models are derived from limited sample of projects, that is why, no estimation model is appropriate for all classes of software and in all development environments.
- Estimation model must be calibrated to reflect local conditions.

Structure of Estimation Models

- A typical empirical model is derived using regression analysis on data collected from past projects
- Overall structure of such models takes the form

$$E = A + B \times (e_v)^C$$

  - A, B and C are empirically derived constants, E is effort in person-months and e_v is estimation variable (either LOC or FP)

Estimation Models - Examples

- Proposed LOC-oriented estimation models

$E = 5.2 \times (KLOC)^{0.91}$	Walston-Felix model
$E = 5.5 + 0.73 \times (KLOC)^{1.16}$	Bailey-Basili model
$E = 3.2 \times (KLOC)^{1.05}$	Boehm simple model
$E = 5.288 \times (KLOC)^{1.047}$	Doty model for KLOC>9

- Proposed FP-oriented estimation models

$E = -91.4 + 0.355\ FP$	Albrecht and Gaffney model
$E = -37 + 0.96\ FP$	Kemerer model
$E = -12.88 + 0.405\ FP$	Small project regression model

- Each of the above will yield a different result for the same values of LOC or FP. The implication is clear – estimation models must be calibrated for local needs.

The COCOMO Model

- In his classic book "*Software Engineering Economics*", Barry Boehm suggested the **CO**nstructive **CO**st **MO**del (COCOMO).
- COCOMO is a software cost estimation method for estimating effort, cost, and schedule for software projects.
- It is based on a set of empirically derived equations.
- These equations incorporate variables considered to be the major cost drivers of software development and maintenance.

The COCOMO Model (Contd.)

- The original COCOMO (COCOMO 81) was first published in 1981 and reflected the software development practices of the day.
- A decade and a half later, the software development techniques changed dramatically and the original COCOMO became problematic.
- It then evolved into COCOMO II to fulfill the changing needs.

The COCOMO Model (Contd.)

- There are three versions of COCOMO 81:
  - Basic
    - Used for rough, early estimates
  - Intermediate
    - The most commonly used version which includes 15 different factors to account for the influence of various project attributes such as personnel capability, hardware constraints, etc.
  - Detailed
    - This version accounts for influence of different factors on individual project phases.
    - It is not used often.

The COCOMO Model (Contd.)

- COCOMO applies to three classes of software projects:
  - **Organic projects** - "small" teams with "good" experience working with "less than rigid" requirements
  - **Semi-detached projects** - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
  - **Embedded projects** - developed within a set of "tight" constraints (hardware, software, operational, ...), may require new technology, unfamiliar algorithms, or new problem solving method

COCOMO II Model (Contd.)

- COCOMO II is actually a hierarchy of estimation models that address the following areas:
  - **Application composition model**
    - Used during the early stages of s/w engg. when prototyping of UI, s/w and system interaction, performance assessment and tech. evaluation are paramount.
  - **Early design stage model**
    - Used once requirements have been stabilized and basic architecture has been established
  - **Post-architecture stage model**
    - Used during the construction of software

COCOMO II Model (Contd.)

- Like all estimation models for software, the COCOMO II requires sizing information.
- Three different sizing options available as part of model hierarchy are:
  - object points,
  - function points and
  - lines of source code.

COCOMO II Model (Contd.)

- COCOMO II Application composition model uses object points
  - Object points is an indirect software measure computed using counts of number of
    - Screens (at UI)
    - Reports
    - Components likely to be required to build the application
  - Each object instance is classified into one of these complexity levels (simple, medium, difficult) on criteria suggested by Boehm.

COCOMO II Model (Contd.)

- Complexity is a function of number of client and server tables required to generate a screen or report and number of sections or views within a screen or report
- After determining complexity, no. of screens, reports and components are weighted as in figure (23.6).
- The object point count is then determined by multiplying the original no. of object instances by weighting factor and summing to obtain a total object point count.

Figures 23.6 & 23.7

Object Type	Complexity Weight		
	Simple	Medium	Difficult
Screen	1	2	3
Report	2	5	8
3GL component			10

Developer's experience/capability	Very Low	Low	Nominal	High	Very High
Environment maturity/capability	Very Low	Low	Nominal	High	Very high
PROD	4	7	13	25	50

COCOMO II Model (Contd.)

- For component-based development or when software reuse is applied, the %reuse is estimated and object point count is adjusted:
  - NOP = (object points) x [(100 - %reuse)/100]
    - NOP is new object points
- To derive estimate of effort based on computed NOP value, a productivity rate must be derived
  - PROD = NOP / person-month
- Estimate of project effort can be derived as:
  - Estimated effort = NOP/PROD

The Software Equation

- Suggested by Putnam & Myers
- It is a multivariable model
- It assumes a specific distribution of effort over life of s/w project
- It has been derived from productivity data collected for over 4000 modern-day s/w projects
- **$E = [LOC \times B^{0.333} / P]^3 \times (1/t^4)$**
  - E = effort in person-months or person-years
  - B = special skills factor
  - P = productivity factor
  - t = project duration (months or years)

The Software Equation (Contd.)

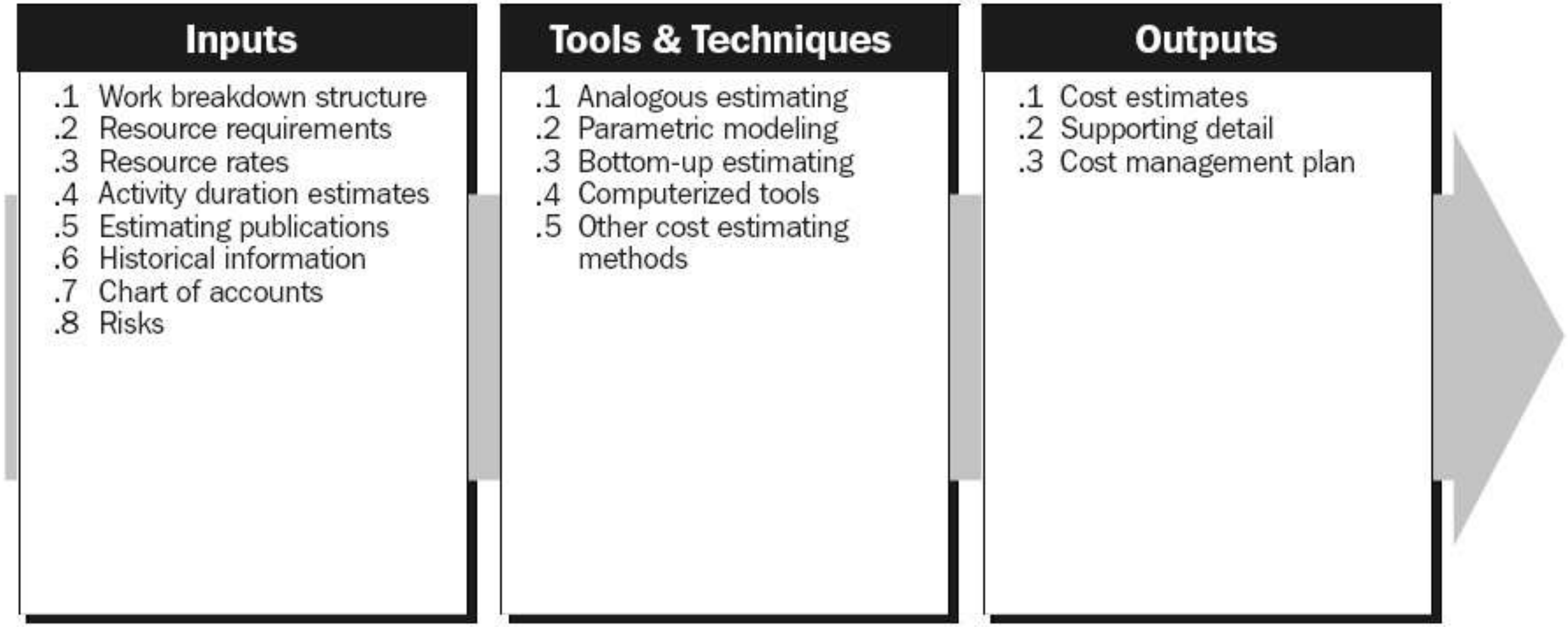
- P reflects
  - Overall process maturity
  - Management practices
  - Extent to which good s/w engg practices are used
  - Level of prog. Languages used
  - State of s/w environment
  - Skills & experience of team
  - Application complexity
- Typical values of P
  - P= 2000 - for a real-time embedded s/w
  - P= 10,000 - for telecomm. & systems s/w
  - P= 28,000 for business applications
- Value of B
  - increases slowly as “the need for integration, testing, quality assurance, documentation and management skills grows”.
  - For small programs (KLOC=5 to 15), B= 0.16, for larger programs (KLOC=more than 70), B=0.39

The Software Equation (Contd.)

- Software equation has two independent parameters
  - LOC
  - t
- Minimum dev. Time equations derived from software equation
  - $t_{min} = 8.14 \, (LOC/P)^{0.43}$
    - in months for $t_{min} > 6$ months
  - $E = 180 \, Bt^3$
    - In person-months for $E >= 20$ person-months

Estimating Cost

- It involves developing an approximation of the costs of resources needed to complete project activities.



Inputs	Tools & Techniques	Outputs
.1 Work breakdown structure	.1 Analogous estimating	.1 Cost estimates
.2 Resource requirements	.2 Parametric modeling	.2 Supporting detail
.3 Resource rates	.3 Bottom-up estimating	.3 Cost management plan
.4 Activity duration estimates	.4 Computerized tools	
.5 Estimating publications	.5 Other cost estimating methods	
.6 Historical information		
.7 Chart of accounts		
.8 Risks		

Inputs to Cost Estimating

- Work Breakdown Structure
  - It organizes cost estimates and ensures that all identified work has been estimated.
- Resource Requirements
  - Involves description of what type of resources are required and in what quantities for each element of WBS.
- Resource Rates
  - Unit rates (e.g., staff cost per hour, bulk material cost per cubic yard) for each resource must be known to calculate project cost.

Inputs to Cost Estimating (Contd.)

- Activity duration estimates
  - These affect cost estimates on any project where project budget includes an allowance for cost of financing (i.e., interest charges)
- Estimating publications
  - Commercially available data on cost estimating.
- Historical Information
  - Historical data can be obtained from one or more of the following:
    - Previous Project files,
    - Commercial cost estimating databases
    - Project team knowledge

Inputs to Cost Estimating (Contd.)

- Chart of accounts
  - It describes coding structure used by performing organization to report financial information in its general ledger.
- Risks
  - The project team considers the extent to which the effect of risk is included in the cost estimates for each activity.

Tools and Techniques for Cost Estimating

- Analogous estimating
  - It is also called top-down estimating, as actual cost of similar previous project is used as basis for current estimating cost of current project.
  - It is a less costly than other techniques but generally less accurate.
  - It is most reliable when
    - previous projects are similar in fact not just in appearance and
    - people making the estimates have the required expertise.

Tools and Techniques for Cost Estimating (Contd.)

- Parametric modeling
  - It involves using project parameters in a mathematical model to predict project costs.
  - Cost and accuracy of this technique varies.
  - It is reliable when
    - the historical info used to develop model was accurate,
    - the parameters used are quantifiable and
    - the model is scalable (works for both very small and very large projects).

Tools and Techniques for Cost Estimating (Contd.)

- Bottom-up estimating
  - It involves cost estimation of individual activities or work packages, then summarizing and rolling up the individual estimates to get a project total.
  - The cost and accuracy of this technique is driven by size and complexity of individual activity or work package.
    - Smaller activities increase the cost and accuracy of the estimating process.
- Computerized tools
  - These include project management software spreadsheets and simulation/statistical tools.
- Other cost estimating methods
  - For example, vendor bid analysis.

Outputs from Cost Estimating

- Cost estimates
  - These are quantitative assessments of likely costs of resources (e.g., labor, material, supplies, etc.) required to complete project activities.
  - They are expressed in units of currency.
- Supporting detail
  - Description of scope of work estimated (by reference to WBS)
  - Documentation about how estimate was developed.
  - Documentation of any assumptions made.
  - An indication of range of possible results, e.g., $10,000 $\pm$ $1,000 to indicate cost between $9,000 and $11,000

Outputs from Cost Estimating (Contd.)

- Cost Management Plan
  - It describes how cost variances will be managed.
  - It may be formal or informal, highly detailed or broadly framed, based on the needs of the project stakeholders.

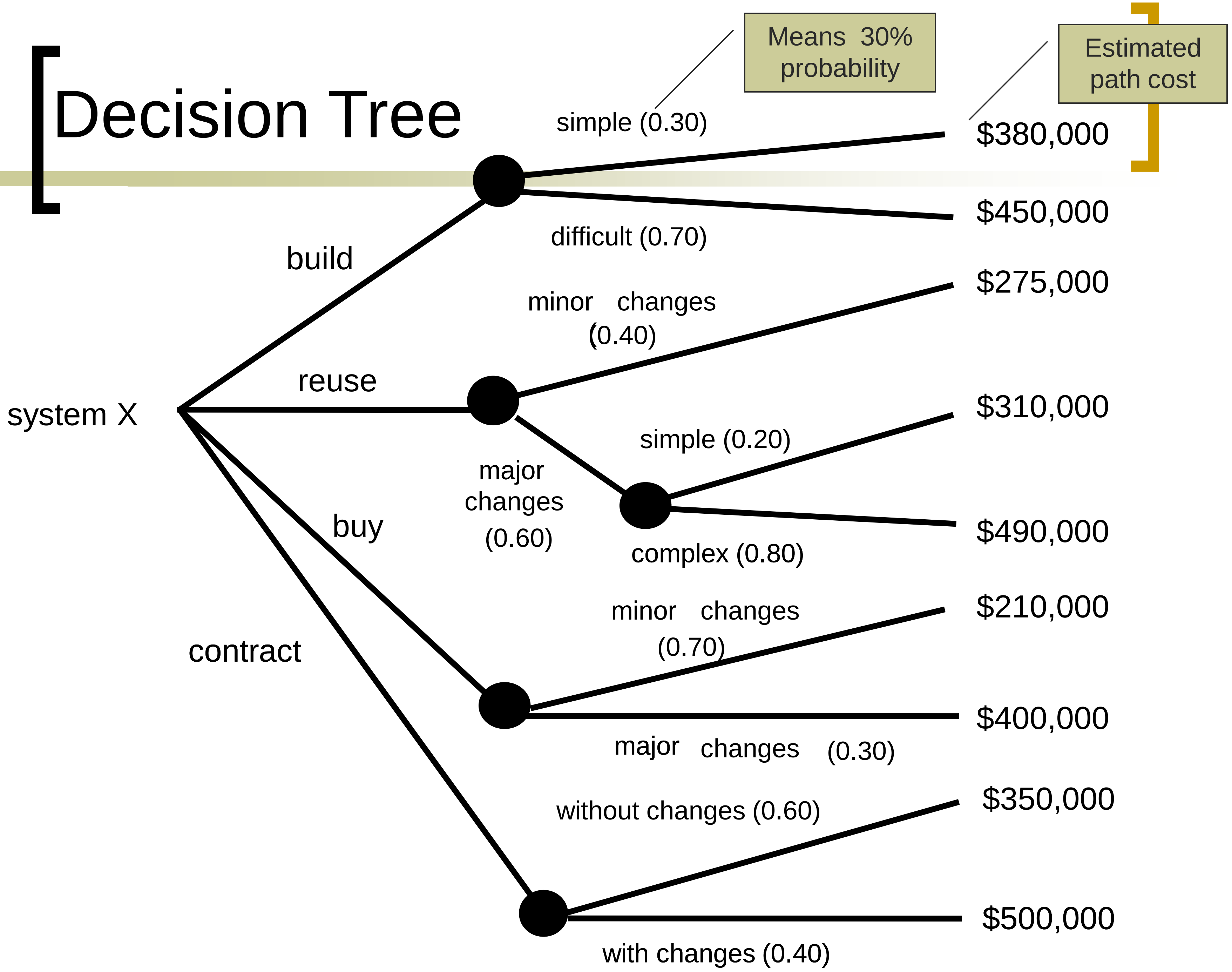
The Make/Buy decision

- Often it is more cost effective to acquire rather than develop a software
- Software managers have following options while making make/buy decisions
  - Software may be purchased (or licensed) off the shelf
  - “Full experience” or “partial experience” software components may be acquired and then modified as needed
  - Software may be custom-built by an outside contractor to meet specifications
- Software criticality to be purchased and the end cost also affect acquisition process

The Make/Buy decision (Contd.)

- For each of the discussed acquisition options, the Make/Buy decision is made based on following conditions
  - Will the software product be available sooner than internally developed software?
  - Will the acquisition cost plus cost of customization be less than cost of developing the software internally?
  - Will the cost of outside support (e.g., maintenance contract) be less than the cost of internal support?

Decision Tree



Decision Tree

- Expected value of cost computed along each branch of the decision tree is:

expected cost =

$$\Sigma \ (\text{path probability})_i \times (\text{estimated path cost})_i$$

- where i is the decision tree path, for example,
  - For Build path
    - expected cost = 0.30($380K)+0.70($450K) = $429K
  - Similarly, for Reuse path, expected cost is $382K; for Buy path, it is $267K; for Contract path, it is $410K.
  - So the obvious choice is “to buy”

Outsourcing

- Acquisition of software (or components) from a source outside the organization
- Software engineering activities are contracted to a third party who does the work at lower cost and (hopefully) at higher quality
- Software work within the company is reduced to contract management activity
- Outsourcing is often a financial decision
- Positive side
  - Cost saving can usually be achieved by reducing own resources (people & infrastructure)
- Negative side
  - Company loses some control over the software and bears the risk of putting its fate in hands of a third party

- Excluded
  - 23.8
  - 23.9 and sub topics

References

- Today’s lecture contents were taken from
  - Chapter 23 – “Software Engineering, A Practitioner's Approach” by Roger Pressman
  - Cost Estimating from PMBOK 2000
  - http://sunset.usc.edu/csse/research/COCOMOII/cocomo_main.html