Chapter 23:
Software Cost Estimation

Instructor:
Dr. Ghazy Assassa
Software cost estimation

- Predicting the resources required for a software development process
Objectives

- To introduce the fundamentals of software costing and pricing

- To describe
  - LOC model
  - COCOMO ‘Constructive Cost Model’
    - Object-point model
    - Function points model
Why Cost Estimation?

- Cost estimation is needed early for s/w pricing
- S/W price = cost + profit
Fundamental estimation questions

- **Effort**
  - How much effort is required to complete an activity?
  - Units: man-day (person-day), man-week, man-month, ...

- **Duration**
  - How much calendar time is needed to complete an activity?
  - Units: hour, day, week, month, year, ...

- **Cost of an activity**
  - What is the total cost of an activity?

- Project estimation and scheduling are interleaved management activities
Software Cost Components

1. Effort costs (dominant factor in most projects)
   - salaries
   - Social and insurance & benefits

2. Tools costs: Hardware and software for development
   - Depreciation on relatively small # of years

3. Travel and Training costs (for particular client)

4. Overheads(OH): Costs must take overheads into account
   - costs of building, air-conditioning, heating, lighting
   - costs of networking and communications (tel, fax, )
   - costs of shared facilities (e.g library, staff restaurant, etc.)
   - depreciation costs of assets
Estimate Uncertainty

- Higher uncertainty
- Lower uncertainty

Cost estimate vs. Development phases:
- Feasibility
- Requirements
- Design
- Code
- Delivery
**Lines Of Code (LOC)**

- Program length (LOC) can be used to predict program characteristics e.g. person-month effort and ease of maintenance
The COCOMO Cost model

Constructive Cost Model

- An empirical model based on project experience
- COCOMO'81 is derived from the analysis of 63 software projects in 1981.
- Well-documented, ‘independent’ model which is not tied to a specific software vendor
- COCOMO II (2000) takes into account different approaches to software development, reuse, etc.
COCOMO II

- COCOMO II is a 3-level model that allows increasingly detailed estimates to be prepared as development progresses
  - Early prototyping level
    - Estimates based on object-points and a simple formula is used for effort estimation
  - Early design level
    - Estimates based on function-points that are then translated to LOC
    - Includes 7 cost drivers
  - Post-architecture level
    - Estimates based on lines of source code or function point
    - Includes 17 cost drivers
COCOMO II Early prototyping level

Object-Points

- Suitable for projects built using modern GUI-builder tools
  - Based on Object-Points
- Supports prototyping projects and projects where there is extensive reuse
- Based on standard estimates of developer productivity in object points/month
- Takes CASE tool use into account
- Formula is
  \[ PM = \frac{\text{NOP} \times (1 - \frac{\%\text{reuse}}{100})}{\text{PROD}} \]
  - PM is the effort in person-months, NOP is the number of object points and PROD is the productivity
Object-Points (for 4GLs)

- Object-points are an alternative function-related measure to function points when 4GLs or similar languages are used for development.
- Object-points are NOT the same as object classes.
- The number of object-points in a program is considered as a weighted estimate of 3 elements:
  - The number of separate screens that are displayed
  - The number of reports that are produced by the system
  - The number of 3GL modules that must be developed to supplement the 4GL code

C:\Software Eng\Cocomo\Software Measurement Page, COCOMO II, object points.htm
## Object-Points – Weighting

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Simple</th>
<th>Medium</th>
<th>Difficult</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Report</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Each 3GL module</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>
Object-Points: Complexity Levels

- **srvr**: number of server data tables used with screen/report
- **clnt**: number of client data tables used with screen/report

<table>
<thead>
<tr>
<th>Number of Views contained</th>
<th># and source of data tables</th>
<th>For Screens</th>
<th>For Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total &lt; 4 (&lt; 2 srvr &lt; 3 clnt)</td>
<td>Total &lt; 8 (2/3 srvr 3-5 clnt)</td>
<td>Total 8+ (&gt; 3 srvr &gt; 5 clnt)</td>
<td>Number of Sections contained</td>
</tr>
<tr>
<td>&lt; 3</td>
<td>simple</td>
<td>simple</td>
<td>medium</td>
</tr>
<tr>
<td>3 - 7</td>
<td>simple</td>
<td>medium</td>
<td>difficult</td>
</tr>
<tr>
<td>&gt; 8</td>
<td>medium</td>
<td>difficult</td>
<td>difficult</td>
</tr>
</tbody>
</table>
Object-Point Estimation

- **Object-points are easy to estimate**
  - simply concerned with screens, reports and 3GL modules

- **At an early point in the development process:**
  - Object-points can be early estimated
  - It is very difficult to estimate the number of lines of code in a system
Productivity Estimates

- LOC productivity
  - Real-time embedded systems: 40-160 LOC/P-month
  - Systems programs: 150-400 LOC/P-month
  - Commercial applications: 200-800 LOC/P-month

- Object-points productivity: PROD
  - measured **4 - 50** object points/person-month
  - depends on tool support and developer capability

<table>
<thead>
<tr>
<th>Developer's experience and Capability / ICASE maturity and capability</th>
<th>Very low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very high</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROD: Productivity Object-point per person-month</td>
<td>4</td>
<td>7</td>
<td>13</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>
Object Point Effort Estimation

- Effort in p-m = NOP / PROD
  - NOP = number of OP of the system

- Example:
  - An application contains 840 Object-points (NOP=840) & Productivity is very high (= 50 object points/person-month)
  - Then, Effort = 840/50 = (16.8) = 17 p-m
Adjustment for % of Reuse

- % reuse: the % of screens, reports, & 3GL modules reused from previous applications, pro-rated by degree of reuse

\[
\text{Adjusted NOP} = \text{NOP} \times (1 - \frac{\% \text{ reuse}}{100})
\]

**Adjusted NOP: New NOP**

**Example:**

- An application contains 840 OP, of which 20% can be supplied by existing components.

\[
\text{Adjusted NOP} = 840 \times (1 - \frac{20}{100}) = 672 \text{ OP} \quad \text{“New OP”}
\]

\[
\text{Adjusted effort} = \frac{672}{50} = (13.4) = 14 \text{ p-m}
\]
Object-Point Estimation Procedure

1. Assess **object-counts** in the system: number of screens, reports, & 3GL.
2. Assess complexity level for each object (use table): simple, medium and difficult.
3. Calculate “NOP” the **object-point count** of the system: add all weights for all object instances.
4. Estimate the % of reuse and compute the adjusted NOP “New Object Points “ to be developed.
5. Determine the productivity rate PROD (use metrics table).
6. Compute the adjusted effort PM = adjusted NOP / PROD.
Object-Point Estimation Example

Assessment of a software system shows that:

● The system includes
  - 6 screens: 2 simple + 3 medium + 1 difficult
  - 3 reports: 2 medium + 1 difficult
  - 2 3GL components

● 30% of the objects could be supplied from previously developed components

● Productivity is high

Compute the estimated effort PM ‘Person-months’ needed to develop the system.
OP Estimation Example: Solution

- Object counts:
  - 2 simple screens $\times 1 = 2$
  - 3 medium screens $\times 2 = 6$
  - 1 difficult screen $\times 3 = 3$
  - 2 medium reports $\times 5 = 10$
  - 1 difficult report $\times 8 = 8$
  - 2 3GL components $\times 10 = 20$
  - NOP $\underline{49}$
**OP Estimation Example: Solution**

- **Adjusted NOP** ‘New NOP’ = NOP * (1 - % reuse / 100)
  
  \[ = 49 \times (1 - \frac{30}{100}) \]
  
  \[ = (34.3) \]
  
  \[ = 35 \]

- For high productivity (metric table): PROD = 25 OP/P-M

- Estimated effort Person-Month = **Adjusted NOP** / PROD

  \[ = \frac{35}{25} \]
  
  \[ = 1.4 \text{ P-M} \]
Function Points: FP
Function Points (5 characteristics)

- Based on a combination of program 5 characteristics
- The number of:
  - External (user) inputs: input transactions that update internal files
  - External (user) outputs: reports, error messages
  - User interactions: inquiries
  - Logical internal files used by the system:
    Example a purchase order logical file composed of 2 physical files/tables Purchase_Order and Purchase_Order_Item
  - External interfaces: files shared with other systems
Function Points (FP)

- A weight is associated with each of the above 5 characteristics.
- Weight range:
  - from 3 for simple feature
  - to 15 for complex feature

- The function point count is computed by multiplying each raw count by the weight and summing all values.
## Function Points - Calculation

<table>
<thead>
<tr>
<th>measurement parameter</th>
<th>count</th>
<th>weighting factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of user inputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of user outputs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of user inquiries</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of files</td>
<td></td>
<td></td>
</tr>
<tr>
<td>number of ext. interfaces</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>simple</th>
<th>avg.</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

\[
\text{count-total} = \text{sum of all values}
\]

\[
\text{complexity multiplier} = \frac{\text{count-total}}{\text{number of files}}
\]

\[
\text{function points} = \text{complexity multiplier} \times X
\]
14 factors: Each factor is rated on a scale of:

**Zero**: not important or not applicable

**Five**: absolutely essential

1. Backup and recovery
2. Data communication
3. Distributed processing functions
4. Is performance critical?
5. Existing operating environment
6. On-line data entry
7. Input transaction built over multiple screens
Adjusted Function Points Count

Complexity: 14 Factors Fi (cont.)

8. Master files updated on-line
9. Complexity of inputs, outputs, files, inquiries
10. Complexity of processing
11. Code design for re-use
12. Are conversion/installation included in design?
13. Multiple installations
14. Application designed to facilitate change by the user
Adjusted Function Points Count
Complexity: 14 Factors Fi (cont.)

\[
\text{AFPC} = \text{UFPC} \times \left( 0.65 + 0.01 \times \sum_{i=1}^{14} F_i \right)
\]

AFPC: Adjusted function point count
UFPC: Unadjusted function point count

\[0 \leq F_i \leq 5\]
Function points and LOC

- FPs can be used to estimate LOC depending on the average number of LOC per FP for a given language.
  - LOC = AVC * number of function points
  - AVC is a language-dependent factor varying from approximately 300 for assemble language to 12-40 for a 4GL
## Relation Between FP & LOC

<table>
<thead>
<tr>
<th>Programming Language</th>
<th>LOC/FP (average)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly language</td>
<td>320</td>
</tr>
<tr>
<td>C</td>
<td>128</td>
</tr>
<tr>
<td>COBOL</td>
<td>106</td>
</tr>
<tr>
<td>FORTRAN</td>
<td>106</td>
</tr>
<tr>
<td>Pascal</td>
<td>90</td>
</tr>
<tr>
<td>C++</td>
<td>64</td>
</tr>
<tr>
<td>Ada</td>
<td>53</td>
</tr>
<tr>
<td>Visual Basic</td>
<td>32</td>
</tr>
<tr>
<td>Smalltalk</td>
<td>22</td>
</tr>
<tr>
<td>Power Builder (code generator)</td>
<td>16</td>
</tr>
<tr>
<td>SQL</td>
<td>12</td>
</tr>
</tbody>
</table>
Expected Software Size

- Based on three-point
- Compute Expected Software Size ($S$) as weighted average of:
  - Optimistic estimate: $S_{\text{opt}}$
  - Most likely estimate: $S_{\text{ml}}$
  - Pessimistic estimate: $S_{\text{pess}}$

\[
S = \left\{ \frac{S_{\text{opt}} + 4 S_{\text{ml}} + S_{\text{pess}}}{6} \right\}
\]

- Beta probability distribution
Example 1: LOC Approach

- A system is composed of 7 subsystems as below

- Given for each subsystem:
  - size in LOC
  - productivity metric: LOC/pm (pm: person month),
  - Cost metric: $/LOC

- Calculate the system total cost in $ and effort in months
## Productivity metric and Cost metric

<table>
<thead>
<tr>
<th>Functions</th>
<th>estimated LOC</th>
<th>LOC/pm</th>
<th>$/LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>UICF</td>
<td>2340</td>
<td>315</td>
<td>14</td>
</tr>
<tr>
<td>2DGA</td>
<td>5380</td>
<td>220</td>
<td>20</td>
</tr>
<tr>
<td>3DGA</td>
<td>6800</td>
<td>220</td>
<td>20</td>
</tr>
<tr>
<td>DSM</td>
<td>3350</td>
<td>240</td>
<td>18</td>
</tr>
<tr>
<td>CGDF</td>
<td>4950</td>
<td>200</td>
<td>22</td>
</tr>
<tr>
<td>PCF</td>
<td>2140</td>
<td>140</td>
<td>28</td>
</tr>
<tr>
<td>DAM</td>
<td>8400</td>
<td>300</td>
<td>18</td>
</tr>
</tbody>
</table>
**Example 1: LOC Approach**

<table>
<thead>
<tr>
<th>Functions</th>
<th>estimated LOC</th>
<th>LOC/pm</th>
<th>$/LOC</th>
<th>Cost</th>
<th>Effort (months)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UICF</td>
<td>2340</td>
<td>315</td>
<td>14</td>
<td>32,000</td>
<td>7.4</td>
</tr>
<tr>
<td>2DGA</td>
<td>5380</td>
<td>220</td>
<td>20</td>
<td>107,000</td>
<td>24.4</td>
</tr>
<tr>
<td>3DGA</td>
<td>6800</td>
<td>220</td>
<td>20</td>
<td>136,000</td>
<td>30.9</td>
</tr>
<tr>
<td>DSM</td>
<td>3350</td>
<td>240</td>
<td>18</td>
<td>60,000</td>
<td>13.9</td>
</tr>
<tr>
<td>CGDF</td>
<td>4950</td>
<td>200</td>
<td>22</td>
<td>109,000</td>
<td>24.7</td>
</tr>
<tr>
<td>PCF</td>
<td>2140</td>
<td>140</td>
<td>28</td>
<td>60,000</td>
<td>15.2</td>
</tr>
<tr>
<td>DAM</td>
<td>8400</td>
<td>300</td>
<td>18</td>
<td>151,000</td>
<td>28.0</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>33,360</strong></td>
<td></td>
<td></td>
<td><strong>655,000</strong></td>
<td></td>
</tr>
</tbody>
</table>
Example 2: LOC Approach

Assuming

- Estimated project LOC = 33200
- Organisational productivity (similar project type) = 620 LOC/p-m
- Burdened labour rate = 8000 $/p-m

Then

- Effort = 33200/620 = (53.6) = 54 p-m
- Cost per LOC = 8000/620 = (12.9) = 13 $/LOC
- Project total Cost = 8000 * 54 = 432000 $
### Example 3: FP Approach

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Info Domain</td>
<td>Optimistic</td>
<td>Likely</td>
<td>Pessim.</td>
<td>Est Count</td>
<td>Weight</td>
<td>FP count</td>
</tr>
<tr>
<td>2</td>
<td># of inputs</td>
<td></td>
<td>22</td>
<td>26</td>
<td>30</td>
<td>26</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td># of outputs</td>
<td></td>
<td>16</td>
<td>18</td>
<td>20</td>
<td>18</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td># of inquiries</td>
<td></td>
<td>16</td>
<td>21</td>
<td>26</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td># of files</td>
<td></td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td># of external int</td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>UFC: Unadjusted Function Count</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Complexity adjustment factor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>FP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Example 3: FP Approach (cont.)

**Complexity Factor**

<table>
<thead>
<tr>
<th>Complexity factor: Fi</th>
<th>value=0</th>
<th>value=1</th>
<th>value=2</th>
<th>value=3</th>
<th>value=4</th>
<th>value=5</th>
<th>Fi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup and recovery</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Data communication</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Distributed processing functions</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Is performance critical?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Existing operating environment</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>On-line data entry</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Input transaction built over multiple screens</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Master files updated on-line</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Complexity of inputs, outputs, files, inquiries</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Complexity of processing</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Code design for re-use</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>Are conversion/installation included in design?</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Multiple installations</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Application designed to facilitate change by the user</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

**Sigma (F)**: 52

**Complexity adjustment factor**: $0.65 + 0.01 \times \Sigma (F) = 1.17$
Example 3: FP Approach (cont.)

Assuming $\sum_i F_i = 52$

$$FP = UFC \times \left[ 0.65 + 0.01 \times \sum_i F_i \right]$$

$$FP = 342 \times 1.17 = 400$$

Complexity adjustment factor = 1.17
Example 4: FP Approach (cont.)

Assuming

- Estimated FP = 400
- Organisation average productivity (similar project type) = 6.5 FP/p-m (person-month)
- Burdened labour rate = 8000  $/p-m

Then

- Estimated effort = 400/6.5 = (61.53) = 62 p-m
- Cost per FP = 8000/6.5 = 1231  $/FP
- Project cost = 8000 * 62 = 496000  $
Estimate Uncertainty: again

![Graph showing cost estimate over project phases]

- **Feasibility**
- **Requirements**
- **Design**
- **Code**
- **Delivery**

**Cost estimate**

- Higher uncertainty
- Lower uncertainty

*x* = Feasibility, 2x = Requirements, 4x = Design, 0.5x = Code, 0.25x = Delivery
Estimation techniques

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

Historical Data

Analysis of historical data

METRICS

Project 1

Project 2

Project n
New Project estimation using available Metrics
Empirical Estimation Models - Algorithmic Cost Modelling

\[ \text{effort} = \text{tuning coefficient} \times \text{size} \]

- Exponent: usually derived as person-months of effort required or usually LOC but may also be function point, empirically derived.
- Either an organisation-dependent constant or a number derived based on complexity of project.
Algorithmic Cost Modelling

\[ \text{Effort} = A \times \text{Size}^B \times M \]

- A is an organisation-dependent constant
- B reflects the nonlinearity (disproportionate) effort for large projects
- M is a multiplier reflecting product, process and people attributes

- Most commonly used product attribute for cost estimation is code size (LOC)
- Most models are basically similar but with different values for A, B and M
COCOMO II Early Design Level: 7 cost drivers

- Estimates can be made after the requirements have been agreed
- Based on standard formula for algorithmic models

\[ PM = A \times \text{Size}^B \times M \]

\[ M = \text{PERS} \times \text{RCPX} \times \text{RUSE} \times \text{PDIF} \times \text{PREX} \times \text{FCIL} \times \text{SCED} \]

- \( A = 2.5 \) in initial calibration,
- Size: manually developed code in KLOC (e.g., C++ console application)
- Exponent \( B \)
  - varies from 1.1 to 1.24 depending on novelty of the project, development flexibility, risk management approaches and the process maturity.
  - \( B \) is calculated using a Scale Factor based on 5 exponent drivers