Software Cost Estimation

“What do you need for the project?”

Lecture Objectives

*To describe different methods of estimation for a software project*
*To understand the factors that affects the estimation of a software development project*

What Do You Estimate?

*Time (schedule)*
*Resources*
*Cost*

Three-point Estimation Technique

*Estimate 3 values For each function or count for each information domain value*
**Optimistic value** ($s_{opt}$)
**Most likely value** ($s_m$)
**Pessimistic value** ($s_{pess}$)
*Compute the expected value ($EV$) for the estimation variable (size), S is:*

$$EV = \frac{s_{opt} + 4s_m + s_{pess}}{6}$$

Time Estimates Example

<table>
<thead>
<tr>
<th>Activity</th>
<th>Immediate Predecessor</th>
<th>Optimistic</th>
<th>Most Likely</th>
<th>Pessimistic</th>
<th>Duration (days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>6</td>
<td>7</td>
<td>14</td>
<td>8</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>8</td>
<td>10</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>C</td>
<td>A</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>B,C</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>F</td>
<td>B,C</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>G</td>
<td>D,E</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>H</td>
<td>F</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Project Schedule

Calculation of the project duration and critical path uses the following times:

*Earliest Start Time (EST)*

- Earliest possible time an activity can begin without interfering with the completion of preceding activity

*Latest Finish Time (LFT)*

- Latest time an activity can be completed without delaying the end of the project
Calculating Earliest Start Times

- Start with the first node, which has time 0
- If only one arrow leads into a node, EST at that node is EST for previous node + time estimate

![Diagram](https://example.com/diagram1.png)

Calculating Earliest Start Times (Continued)

- If more than one arrow leads to a node, the EST of the node is the largest time value of the separate paths

![Diagram](https://example.com/diagram2.png)

Project Duration

- When all the ESTs have been calculated, the EST at the last node is the project duration
- The project duration is also the LFT of the last node
- The calculation is then 'reversed' to previous nodes for the LFT values

![Diagram](https://example.com/diagram3.png)

Calculating Latest Finish Times

- Start with the last node, which has LFT = project duration
- If only one arrow originates from a node, LFT at that node is LFT for next node - time estimate

![Diagram](https://example.com/diagram4.png)

Calculating Latest Finish Times (Continued)

- If more than one arrow originates at a node, the LST of the node is the smallest time value of the separate paths

![Diagram](https://example.com/diagram5.png)

Activity Times

- Available time = LFT - EST
  - Example, Available time for X = 12 - 5 = 7
- Total float = Available time - time estimate
  - Example, Float for X = 7 - 4 = 3
- This means that activity X has a 'slack' of 3
  - can start late, within 3 days/weeks
### Critical Path

- All activities with 0 float time form the critical path of the project.
- Any delay on these activities will directly affect the project duration.

### Estimation Options

- **Expert Judgment**
  - Estimation done by panel of experts
- **Bottom-Up Approach**
  - Project separated into components
  - Estimate components, then combined
- **Algorithmic Models**
  - Use of software metrics, formulas
  - Historical models

### Critical Path Example

![Critical Path Diagram]

### Expert Judgment

- One or more experts are consulted to provide estimates, given information on the software project.
- Inherently top-down approach
- Common approach is to have a panel of experts, who will agree on estimates by consensus.
- May be affected by group dynamics.
- Interesting variation: Delphi technique.

### Software Cost Factors

- Programmer Ability
- Product Complexity
- Product Size
- Available Time
- Required Level of Reliability
- Level of Technology

### Delphi Technique

- Developed by Rand Corporation
- Coordinator provides estimator information
- Estimator provides estimation individually, without discussion with each other
- Coordinator summarises estimations and other responses, distributes for another round of estimation.
- Estimation repeated as much as required.
Bottom-Up Approach

- Product or requirements broken down into smaller components
- Estimates done for components, then combined for overall estimate
- Applies to Work Breakdown Structure, or other similar methods of decomposing the project

Constructive Cost Model (COCOMO)

- Introduced by Barry Boehm
- Widely used for effort and cost estimation
- 3 models:
  - Basic COCOMO
  - Intermediate COCOMO
  - Advanced COCOMO
- Select a model for estimation, identify ‘mode’ and estimate kLOC, and effort (and cost) is calculated from the model

Bottom-Up Approach (Continued)

- Easier to estimate, more accurate and detailed estimate can be done
- However, the product may be more than the total of the components
- Additional cost may be required to consolidate the components

Project Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Size</th>
<th>Innovation</th>
<th>Constraints</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic mode</td>
<td>Small</td>
<td>Little</td>
<td>Not tight</td>
<td>Good</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>Large</td>
<td>Greater</td>
<td>Tight</td>
<td>Mixed</td>
</tr>
<tr>
<td>Embedded mode</td>
<td>Medium</td>
<td>Medium</td>
<td>Very tight</td>
<td>-</td>
</tr>
</tbody>
</table>

Algorithmic Models

- Costs are analyzed using mathematical formulae linking costs with metrics
- Common methods use kLOC
- Example: 3-point estimation
- Detailed studies of software project provides empirical estimation models
- Example: COCOMO

Basic COCOMO

- Computes software development effort (and cost) as function of program size expressed in estimated lines of code
- Model:

<table>
<thead>
<tr>
<th>Category</th>
<th>a₀</th>
<th>b</th>
<th>c₀</th>
<th>d₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>2.4</td>
<td>1.05</td>
<td>2.5</td>
<td>0.38</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>3.0</td>
<td>1.12</td>
<td>2.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.20</td>
<td>2.5</td>
<td>0.32</td>
</tr>
</tbody>
</table>
Basic COCOMO Equations

\[ E = a_k \cdot \text{kLOC}^{b_k} \]
\[ D = c_k \cdot E^{d_k} \]

where
\( E \) is effort in person-months
\( D \) is development time in months
\( \text{kLOC} \) is estimated number of lines of code

Intermediate COCOMO

The equation computes software development effort as a function of program size and a set of “cost drivers” that include subjective assessments of product, hardware, personnel, and project attributes.

Give rating to 15 attributes, from “very low” to “extra high”, find effort multiplier (from table) and product of all effort multipliers gives an effort adjustment factor (EAF).

Cost Driver Attributes (Continued)

- Personnel attributes
  - Analyst capability, Programmer capability
  - Applications experience
  - Virtual machine experience
  - Programming language experience

- Project attributes
  - Use of modern programming practices
  - Use of software tools
  - Required development schedule

Intermediate COCOMO Equation

\[ E = a_k \cdot \text{kLOC}^{b_k} \times \text{EAF} \]

where
\( E \) is effort in person-months,
\( \text{kLOC} \) is estimated number of lines of code

Cost Driver Attributes

- Product attributes
  - Required reliability
  - Database size
  - Product complexity

- Computer attributes
  - Execution time constraint
  - Main storage constraint
  - Virtual machine volatility
  - Computer turnaround time

Advanced COCOMO

- Incorporates all characteristics of intermediate COCOMO with an assessment of the cost driver’s impact on each step of software engineering process.
Estimation Issues

- Historical Data
- Accuracy
- Estimation Technique
- Automation
- Improving the Estimate

References

- "Software Engineering” by Ian Sommerville, Addison-Wesley, 2001