COCOMO Model

Lecture – 10
Outline of the talk

- COCOMO Model
- Types of COCOMO Model
- COCOMO – II
The Constructive Cost Model (COCOMO)

Constructive Cost model
(COCOMO)

Basic  Intermediate  Detailed

Model proposed by
B. W. Boehm’s
through his book
Software Engineering Economics in 1981
Software Project Planning

COCOMO applied to

- Organic mode
- Semidetached mode
- Embedded mode
## Software Project Planning

<table>
<thead>
<tr>
<th>Mode</th>
<th>Project size</th>
<th>Nature of Project</th>
<th>Innovation</th>
<th>Deadline of the project</th>
<th>Development Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>Typically 2-50 KLOC</td>
<td>Small size project, experienced developers in the familiar environment. For example, pay roll, inventory projects etc.</td>
<td>Little</td>
<td>Not tight</td>
<td>Familiar &amp; In house</td>
</tr>
<tr>
<td>Semi detached</td>
<td>Typically 50-300 KLOC</td>
<td>Medium size project, Medium size team, Average previous experience on similar project. For example: Utility systems like compilers, database systems, editors etc.</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>Embedded</td>
<td>Typically over 300 KLOC</td>
<td>Large project, Real time systems, Complex interfaces, Very little previous experience. For example: ATMs, Air Traffic Control etc.</td>
<td>Significant</td>
<td>Tight</td>
<td>Complex Hardware/ customer Interfaces required</td>
</tr>
</tbody>
</table>
Basic COCOMO model takes the form

\[ E = a_b (KLOC)^{b_b} \]

\[ D = c_b (E)^{d_b} \]

where \( E \) is effort applied in Person-Months, and \( D \) is the development time in months. The coefficients \( a_b, b_b, c_b \) and \( d_b \) are given in table 4 (a).
## Software Project Planning

<table>
<thead>
<tr>
<th>Software Project</th>
<th>( a_b )</th>
<th>( b_b )</th>
<th>( c_b )</th>
<th>( d_b )</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>Semidetached</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>

*Table 4(a): Basic COCOMO coefficients*
When effort and development time are known, the average staff size to complete the project may be calculated as:

Average staff size

\[
(SS) = \frac{E}{D} \text{ Persons}
\]

When project size is known, the productivity level may be calculated as:

Productivity

\[
(P) = \frac{KLOC}{E} \text{ KLOC / PM}
\]
Example: 4.5

Suppose that a project was estimated to be 400 KLOC. Calculate the effort and development time for each of the three modes i.e., organic, semidetached and embedded.
Solution

The basic COCOMO equation take the form:

\[ E = a_b (KLOC)^{b_b} \]
\[ D = c_b (KLOC)^{d_b} \]

Estimated size of the project = 400 KLOC

(i) Organic mode

\[ E = 2.4(400)^{1.05} = 1295.31 \text{ PM} \]
\[ D = 2.5(1295.31)^{0.38} = 38.07 \text{ PM} \]
(ii) Semidetached mode

\[ E = 3.0(400)^{1.12} = 2462.79 \text{ PM} \]
\[ D = 2.5(2462.79)^{0.35} = 38.45 \text{ PM} \]

(iii) Embedded mode

\[ E = 3.6(400)^{1.20} = 4772.81 \text{ PM} \]
\[ D = 2.5(4772.8)^{0.32} = 38 \text{ PM} \]
A project size of 200 KLOC is to be developed. Software development team has average experience on similar type of projects. The project schedule is not very tight. Calculate the effort, development time, average staff size and productivity of the project.
The semi-detached mode is the most appropriate mode; keeping in view the size, schedule and experience of the development team.

Hence

\[ E = 3.0(200)^{1.12} = 1133.12 \text{ PM} \]
\[ D = 2.5(1133.12)^{0.35} = 29.3 \text{ PM} \]

Average staff size

\[ (SS) = \frac{E}{D} \text{ Persons} \]

\[ = \frac{1133.12}{29.3} = 38.67 \text{ Persons} \]
Software Project Planning

Productivity \[ P = \frac{KLOC}{E} = \frac{200}{1133.12} = 0.1765 \text{ KLOC/PM} \]

\[ P = 176 \text{ LOC/PM} \]
Intermediate Model

Cost drivers

(i) Product Attributes
- Required s/w reliability
- Size of application database
- Complexity of the product

(ii) Hardware Attributes
- Run time performance constraints
- Memory constraints
- Virtual machine volatility
- Turnaround time
(iii) Personal Attributes

- Analyst capability
- Programmer capability
- Application experience
- Virtual m/c experience
- Programming language experience

(iv) Project Attributes

- Modern programming practices
- Use of software tools
- Required development Schedule
# Software Project Planning

Multipliers of different cost drivers

<table>
<thead>
<tr>
<th>Cost Drivers</th>
<th>RATINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very low</td>
</tr>
<tr>
<td><strong>Product Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>RELY</td>
<td>0.75</td>
</tr>
<tr>
<td>DATA</td>
<td>--</td>
</tr>
<tr>
<td>CPLX</td>
<td>0.70</td>
</tr>
<tr>
<td><strong>Computer Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>--</td>
</tr>
<tr>
<td>STOR</td>
<td>--</td>
</tr>
<tr>
<td>VIRT</td>
<td>--</td>
</tr>
<tr>
<td>TURN</td>
<td>--</td>
</tr>
</tbody>
</table>
# Software Project Planning

<table>
<thead>
<tr>
<th>Cost Drivers</th>
<th>RATINGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Very low</td>
</tr>
<tr>
<td><strong>Personnel Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>ACAP</td>
<td>1.46</td>
</tr>
<tr>
<td>AEXP</td>
<td>1.29</td>
</tr>
<tr>
<td>PCAP</td>
<td>1.42</td>
</tr>
<tr>
<td>VEXP</td>
<td>1.21</td>
</tr>
<tr>
<td>LEXP</td>
<td>1.14</td>
</tr>
<tr>
<td><strong>Project Attributes</strong></td>
<td></td>
</tr>
<tr>
<td>MODP</td>
<td>1.24</td>
</tr>
<tr>
<td>TOOL</td>
<td>1.24</td>
</tr>
<tr>
<td>SCED</td>
<td>1.23</td>
</tr>
</tbody>
</table>

*Table 5: Multiplier values for effort calculations*
Intermediate COCOMO equations

\[ E = a_i (KLOC)^{b_i} \times EAF \]
\[ D = c_i (E)^{d_i} \]

<table>
<thead>
<tr>
<th>Project</th>
<th>(a_i)</th>
<th>(b_i)</th>
<th>(c_i)</th>
<th>(d_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>3.2</td>
<td>1.05</td>
<td>2.5</td>
<td>0.38</td>
</tr>
<tr>
<td>Semidetached</td>
<td>3.0</td>
<td>1.12</td>
<td>2.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>2.8</td>
<td>1.20</td>
<td>2.5</td>
<td>0.32</td>
</tr>
</tbody>
</table>

**Table 6:** Coefficients for intermediate COCOMO
Software Project Planning

Detailed COCOMO Model

- Phase-Sensitive effort multipliers
- Three level product hierarchy
  - Modules
  - Subsystem
  - System level

Cost drivers
- design
- & test
- & test

Manpower allocation for each phase
Development Phase

Plan / Requirements

EFFORT : 6% to 8%

DEVELOPMENT TIME : 10% to 40%

% depend on mode & size
Software Project Planning

Design
Effort : 16% to 18%
Time : 19% to 38%

Programming
Effort : 48% to 68%
Time : 24% to 64%

Integration & Test
Effort : 16% to 34%
Time : 18% to 34%
Software Project Planning

Principle of the effort estimate

Size equivalent

As the software might be partly developed from software already existing (that is, re-usable code), a full development is not always required. In such cases, the parts of design document (DD%), code (C%) and integration (I%) to be modified are estimated. Then, an adjustment factor, A, is calculated by means of the following equation.

\[ A = 0.4 \text{DD} + 0.3 \text{C} + 0.3 \text{I} \]

The size equivalent is obtained by

\[ S \text{(equivalent)} = \frac{S \times A}{100} \]

\[ E_p = \mu_p E \]

\[ D_p = \tau_p D \]
## Software Project Planning

### Lifecycle Phase Values of $\mu_p$

<table>
<thead>
<tr>
<th>Mode &amp; Code Size</th>
<th>Plan &amp; Requirements</th>
<th>System Design</th>
<th>Detailed Design</th>
<th>Module Code &amp; Test</th>
<th>Integration &amp; Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic Small S≈2</td>
<td>0.06</td>
<td>0.16</td>
<td>0.26</td>
<td>0.42</td>
<td>0.16</td>
</tr>
<tr>
<td>Organic medium S≈32</td>
<td>0.06</td>
<td>0.16</td>
<td>0.24</td>
<td>0.38</td>
<td>0.22</td>
</tr>
<tr>
<td>Semidetached medium S≈32</td>
<td>0.07</td>
<td>0.17</td>
<td>0.25</td>
<td>0.33</td>
<td>0.25</td>
</tr>
<tr>
<td>Semidetached large S≈128</td>
<td>0.07</td>
<td>0.17</td>
<td>0.24</td>
<td>0.31</td>
<td>0.28</td>
</tr>
<tr>
<td>Embedded large S≈128</td>
<td>0.08</td>
<td>0.18</td>
<td>0.25</td>
<td>0.26</td>
<td>0.31</td>
</tr>
<tr>
<td>Embedded extra large S≈320</td>
<td>0.08</td>
<td>0.18</td>
<td>0.24</td>
<td>0.24</td>
<td>0.34</td>
</tr>
</tbody>
</table>

**Table 7**: Effort and schedule fractions occurring in each phase of the lifecycle
Table 7: Effort and schedule fractions occurring in each phase of the lifecycle

<table>
<thead>
<tr>
<th>Mode &amp; Code Size</th>
<th>Plan &amp; Requirements</th>
<th>System Design</th>
<th>Detailed Design</th>
<th>Module Code &amp; Test</th>
<th>Integration &amp; Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic Small S≈2</td>
<td>0.10</td>
<td>0.19</td>
<td>0.24</td>
<td>0.39</td>
<td>0.18</td>
</tr>
<tr>
<td>Organic medium S≈32</td>
<td>0.12</td>
<td>0.19</td>
<td>0.21</td>
<td>0.34</td>
<td>0.26</td>
</tr>
<tr>
<td>Semidetached medium S≈32</td>
<td>0.20</td>
<td>0.26</td>
<td>0.21</td>
<td>0.27</td>
<td>0.26</td>
</tr>
<tr>
<td>Semidetached large S≈128</td>
<td>0.22</td>
<td>0.27</td>
<td>0.19</td>
<td>0.25</td>
<td>0.29</td>
</tr>
<tr>
<td>Embedded large S≈128</td>
<td>0.36</td>
<td>0.36</td>
<td>0.18</td>
<td>0.18</td>
<td>0.28</td>
</tr>
<tr>
<td>Embedded extra large S≈320</td>
<td>0.40</td>
<td>0.38</td>
<td>0.16</td>
<td>0.16</td>
<td>0.30</td>
</tr>
</tbody>
</table>
Software Project Planning

Distribution of software life cycle:

1. Requirement and product design
   (a) Plans and requirements
   (b) System design

2. Detailed Design
   (a) Detailed design

3. Code & Unit test
   (a) Module code & test

4. Integrate and Test
   (a) Integrate & Test
Example: 4.7

A new project with estimated 400 KLOC embedded system has to be developed. Project manager has a choice of hiring from two pools of developers: Very highly capable with very little experience in the programming language being used

Or

Developers of low quality but a lot of experience with the programming language. What is the impact of hiring all developers from one or the other pool?
Software Project Planning

Solution

This is the case of embedded mode and model is intermediate COCOMO.

Hence

\[ E = a_i (KLOC)^{d_i} \]

\[ = 2.8 \times (400)^{1.20} = 3712 \text{ PM} \]

Case I: Developers are very highly capable with very little experience in the programming being used.

\[ \text{EAF} = 0.82 \times 1.14 = 0.9348 \]

\[ E = 3712 \times 0.9348 = 3470 \text{ PM} \]

\[ D = 2.5 \times (3470)^{0.32} = 33.9 \text{ M} \]
**Software Project Planning**

**Case II:** Developers are of low quality but lot of experience with the programming language being used.

\[
\begin{align*}
\text{EAF} & = 1.29 \times 0.95 = 1.22 \\
E & = 3712 \times 1.22 = 4528 \text{ PM} \\
D & = 2.5 \times (4528)^{0.32} = 36.9 \text{ M}
\end{align*}
\]

Case II requires more effort and time. Hence, low quality developers with lot of programming language experience could not match with the performance of very highly capable developers with very little experience.
Consider a project to develop a full screen editor. The major components identified are:

I. Screen edit
II. Command Language Interpreter
III. File Input & Output
IV. Cursor Movement
V. Screen Movement

The size of these are estimated to be 4k, 2k, 1k, 2k and 3k delivered source code lines. Use COCOMO to determine

1. Overall cost and schedule estimates (assume values for different cost drivers, with at least three of them being different from 1.0)

2. Cost & Schedule estimates for different phases.
Solution

Size of five modules are:

Screen edit = 4 KLOC
Command language interpreter = 2 KLOC
File input and output = 1 KLOC
Cursor movement = 2 KLOC
Screen movement = 3 KLOC
Total = 12 KLOC
Let us assume that significant cost drivers are

i. Required software reliability is high, i.e., 1.15

ii. Product complexity is high, i.e., 1.15

iii. Analyst capability is high, i.e., 0.86

iv. Programming language experience is low, i.e., 1.07

v. All other drivers are nominal

\[ EAF = 1.15 \times 1.15 \times 0.86 \times 1.07 = 1.2169 \]
(a) The initial effort estimate for the project is obtained from the following equation

\[ E = a_i \times (KLOC)^{b_i} \times EAF \]

\[ = 3.2 \times (12)^{1.05} \times 1.2169 = 52.91 \text{ PM} \]

Development time

\[ D = C_i(E)^{d_i} \]

\[ = 2.5 \times (52.91)^{0.38} = 11.29 \text{ M} \]

(b) Using the following equations and referring Table 7, phase wise cost and schedule estimates can be calculated.

\[ E_p = \mu_p E \]

\[ D_p = \tau_p D \]
Since size is only 12 KLOC, it is an organic small model. Phase wise effort distribution is given below:

- System Design = 0.16 x 52.91 = 8.465 PM
- Detailed Design = 0.26 x 52.91 = 13.756 PM
- Module Code & Test = 0.42 x 52.91 = 22.222 PM
- Integration & Test = 0.16 x 52.91 = 8.465 Pm

Now Phase wise development time duration is

- System Design = 0.19 x 11.29 = 2.145 M
- Detailed Design = 0.24 x 11.29 = 2.709 M
- Module Code & Test = 0.39 x 11.29 = 4.403 M
- Integration & Test = 0.18 x 11.29 = 2.032 M
The following categories of applications / projects are identified by COCOMO-II and are shown in fig. 4 shown below:

Fig. 4 : Categories of applications / projects
# Table 8: Stages of COCOMO-II

<table>
<thead>
<tr>
<th>Stage No</th>
<th>Model Name</th>
<th>Application for the types of projects</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stage I</td>
<td>Application composition estimation model</td>
<td>Application composition</td>
<td>In addition to application composition type of projects, this model is also used for prototyping (if any) stage of application generators, infrastructure &amp; system integration.</td>
</tr>
<tr>
<td>Stage II</td>
<td>Early design estimation model</td>
<td>Application generators, infrastructure &amp; system integration</td>
<td>Used in early design stage of a project, when less is known about the project.</td>
</tr>
<tr>
<td>Stage III</td>
<td>Post architecture estimation model</td>
<td>Application generators, infrastructure &amp; system integration</td>
<td>Used after the completion of the detailed architecture of the project.</td>
</tr>
</tbody>
</table>
Application Composition Estimation Model

Assess object counts

Classify complexity levels of each object

Assign complexity weights to each object

Determine object points

Compute new object points

Calculate productivity rate

Compute the estimated effort in person months

Fig.5: Steps for the estimation of effort in person months
Software Project Planning

i. Assess object counts: Estimate the number of screens, reports and 3 GL components that will comprise this application.

ii. Classification of complexity levels: We have to classify each object instance into simple, medium and difficult complexity levels depending on values of its characteristics.

<table>
<thead>
<tr>
<th>Number of views contained</th>
<th># and sources of data tables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total &lt; 4</td>
</tr>
<tr>
<td></td>
<td>(Total &lt; 4 (Total &lt; 4)</td>
</tr>
<tr>
<td></td>
<td>(Total &lt; 4 (Total &lt; 4)</td>
</tr>
<tr>
<td></td>
<td>(&lt; 2 server)</td>
</tr>
<tr>
<td></td>
<td>(&lt; 2 server)</td>
</tr>
<tr>
<td></td>
<td>(&lt; 3 client))</td>
</tr>
<tr>
<td>&lt; 3</td>
<td>Simple</td>
</tr>
<tr>
<td>3 - 7</td>
<td>Simple</td>
</tr>
<tr>
<td>&gt; 8</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 9 (a): For screens
### Software Project Planning

**Table 9 (b):** For reports

<table>
<thead>
<tr>
<th>Number of sections contained</th>
<th># and sources of data tables</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total &lt; 4</td>
</tr>
<tr>
<td></td>
<td>(&lt; 2 server</td>
</tr>
<tr>
<td></td>
<td>&lt; 3 client)</td>
</tr>
<tr>
<td>0 or 1</td>
<td>Simple</td>
</tr>
<tr>
<td>2 or 3</td>
<td>Simple</td>
</tr>
<tr>
<td>4 +</td>
<td>Medium</td>
</tr>
</tbody>
</table>
iii. Assign complexity weight to each object: The weights are used for three object types i.e., screen, report and 3GL components using the Table 10.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Complexity Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Simple</td>
</tr>
<tr>
<td>Screen</td>
<td>1</td>
</tr>
<tr>
<td>Report</td>
<td>2</td>
</tr>
<tr>
<td>3GL Component</td>
<td>—</td>
</tr>
</tbody>
</table>

**Table 10:** Complexity weights for each level
iv. **Determine object points:** Add all the weighted object instances to get one number and this known as object-point count.

v. **Compute new object points:** We have to estimate the percentage of reuse to be achieved in a project. Depending on the percentage reuse, the new object points (NOP) are computed.

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

NOP are the object points that will need to be developed and differ from the object point count because there may be reuse.
vi. Calculation of productivity rate: The productivity rate can be calculated as:

Productivity rate (PROD) = NOP/Person month

<table>
<thead>
<tr>
<th>Developer’s experience &amp; capability; ICASE maturity &amp; capability</th>
<th>PROD (NOP/PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very low</td>
<td>4</td>
</tr>
<tr>
<td>Low</td>
<td>7</td>
</tr>
<tr>
<td>Nominal</td>
<td>13</td>
</tr>
<tr>
<td>High</td>
<td>25</td>
</tr>
<tr>
<td>Very high</td>
<td>50</td>
</tr>
</tbody>
</table>

Table 11: Productivity values
vii. Compute the effort in Persons-Months: When PROD is known, we may estimate effort in Person-Months as:

\[
\text{Effort in PM} = \frac{\text{NOP}}{\text{PROD}}
\]
Consider a database application project with the following characteristics:

I. The application has 4 screens with 4 views each and 7 data tables for 3 servers and 4 clients.

II. The application may generate two report of 6 sections each from 07 data tables for two server and 3 clients. There is 10% reuse of object points. The developer’s experience and capability in the similar environment is low. The maturity of organization in terms of capability is also low. Calculate the object point count, New object points and effort to develop such a project.
Solution

This project comes under the category of application composition estimation model.

Number of screens = 4 with 4 views each
Number of reports = 2 with 6 sections each

From Table 9 we know that each screen will be of medium complexity and each report will be difficult complexity.

Using Table 10 of complexity weights, we may calculate object point count.

\[
= 4 \times 2 + 2 \times 8 = 24
\]

\[
24 \times (100 - 10)
\]

\[
\text{NOP} = \frac{24}{100} = 21.6
\]
Table 11 gives the low value of productivity (PROD) i.e. 7.

\[
\text{NOP} = \frac{\text{Efforts in PM}}{\text{PROD}}
\]

\[
\begin{align*}
\text{Efforts} &= \frac{21.6}{7} = 3.086 \text{ PM}
\end{align*}
\]
Assignment

- Explain COCOMO Model in detail
Research Work

- **Cost Models for Future Software Life Cycle Processes: COCOMO 2**