Software Cost Estimation and COCOMO II

Sunita Devnani-Chulani
USC-CSE

Wescon ‘97
November 4
Presentation Outline:

→ Steps in Software Estimation

• COCOMO II
  – Overview and Objectives
  – Post-Architecture Model
  – Reuse Model
  – Other Ongoing Research and Plans
• Information Sources
Steps in Software Estimation

1. Establish Objectives
   - Rough Sizing
   - Make-or-Buy
   - Detailed Planning

2. Allocate enough time, dollars, talent

3. Pin down software requirements
   - Document definitions, assumptions

4. Work out as much detail as objectives permit

5. Use several independent techniques + sources
   - Top-Down vs. Bottom-Up
   - Algorithm vs. Expert Judgement

6. Compare and iterate estimates
   - Pin down and resolve inconsistencies
   - Be conservative

7. Follow up
SW Costing and Sizing
Accuracy vs. Phase

Relative Size Range

Completed Programs
USAF/ESD Proposals

0.25x
0.5x
x
1.25x
1.5x
2x
4x

Feasibility
Plans and Rqts.
Product Design
Detail Design
Accepted Software

Product Design Spec.
Detail Design Spec.

Concept of Operation
Rqts. Spec.

Devel. and Test
Phases and Milestones

Size (DSI) + Cost ($)
# Software Cost Estimation Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithmic Model</td>
<td>• Objective, repeatable, analyzable formula &lt;br&gt; • Efficient, good for sensitivity analysis &lt;br&gt; • Objectively calibrated to experience</td>
<td>• Subjective inputs &lt;br&gt; • Assessment of exceptional circumstances &lt;br&gt; • Calibrated to past, not future</td>
</tr>
<tr>
<td>Expert Judgement</td>
<td>• Assessment of representativeness, interactions, exceptional circumstances</td>
<td>• No better than participants &lt;br&gt; • Biases, incomplete recall</td>
</tr>
<tr>
<td>Analogy</td>
<td>• Based on representative experience</td>
<td>• Representativeness of experience</td>
</tr>
<tr>
<td>Parkinson</td>
<td>• Correlates with some experience</td>
<td>• Reinforces poor practice</td>
</tr>
<tr>
<td>Price-To-Win</td>
<td>• Often wins</td>
<td>• Generally produces large overruns</td>
</tr>
<tr>
<td>Top-Down</td>
<td>• System level focus Efficient</td>
<td>• Less detailed basis &lt;br&gt; • Less stable</td>
</tr>
<tr>
<td>Bottom-Up</td>
<td>• More detailed basis &lt;br&gt; • More stable &lt;br&gt; • Fosters individual commitment</td>
<td>• May overlook system level costs &lt;br&gt; • Requires more effort</td>
</tr>
</tbody>
</table>
Presentation Outline:

• Steps in Software Estimation

  ➞ COCOMO II

    ➞ Overview and Objectives
      – Post-Architecture Model
      – Reuse Model
      – Other Ongoing Research and Plans

• Information Sources
Partial List of COCOMO Packages - STSC, Mar 1993

- CB COCOMO
- COCOMO1D
- COCOMO1
- CoCoPro
- COSTAR
- COSTMODL
- GECOMO Plus
- GECOMO Plus
- GHL COCOMO
- REVIC
- SECOMO
- SWAN
COCOMO Black Box Model

- Software product size estimate
- Software product, process, computer, and personnel attributes
- Software reuse, maintenance, and increment parameters
- Software organization’s project data

COCOMO II

- Software development, maintenance cost and schedule estimates
- Cost, schedule distribution by phase, activity, increment
- COCOMO recalibrated to organization’s data
Future Software Marketplace
Sector

User programming
(55M performers in US in year 2005)

Application generators
(0.6M)

Application composition
(0.7M)

System integration
(0.7M)

Infrastructure
(0.75M)
COCOMO II Family of Models

• User Programming: No need for cost model
• Applications Composition: Use object counts or object points
  - Count (weight) screens, reports, 3GL routines
• System Integration; development of applications generators and infrastructure software
  - Prototyping: Applications composition model
  - Early design: Function Points and/or Source Statements and 7 cost drivers
  - Post-architecture: Source Statements and/or Function Points and 17 cost drivers
  - Stronger reuse/reengineering model
Presentation Outline:

- Steps in Software Estimation
  - COCOMO II
    - Overview and Objectives
  - Post-Architecture Model
    - Reuse Model
    - Other Ongoing Research and Plans
- Information Sources
Post-Architecture Model Formulation

Effort (person-months) = A*(Size)^B* \( \prod_{i=1}^{17} EM_i \)

- where A is a constant derived from calibration
- B = 1.01 + 0.01* \( \sum_{i=1}^{5} SF_i \), where SF_i is a weighting factor for 5th scale driver
- and EM_i is the effort multiplier for the 5th cost driver.
Scale Factors

\[ B = 1.01 + 0.01 \times \sum SF_i \]

<table>
<thead>
<tr>
<th>Scale Factors (( W ))</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>thoroughly unprecedented</td>
<td>largely unprecedented</td>
<td>somewhat unprecedented</td>
<td>generally familiar</td>
<td>largely familiar</td>
<td>thoroughly familiar</td>
</tr>
<tr>
<td>FLEX</td>
<td>rigorous</td>
<td>occasional relaxation</td>
<td>some relaxation</td>
<td>general conformity</td>
<td>some conformity</td>
<td>general goals</td>
</tr>
<tr>
<td>RESL</td>
<td>little (20%)</td>
<td>some (40%)</td>
<td>often (60%)</td>
<td>generally (75%)</td>
<td>mostly (90%)</td>
<td>full (100%)</td>
</tr>
<tr>
<td>TEAM</td>
<td>very difficult interactions</td>
<td>some difficult interactions</td>
<td>basically cooperative interactions</td>
<td>largely cooperative</td>
<td>highly cooperative</td>
<td>seamless interactions</td>
</tr>
<tr>
<td>PMAT</td>
<td>weighted sum of KPA achievement levels</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- PREC: \( W \)
- FLEX: \( W \)
- RESL: \( W \)
- TEAM: \( W \)
- PMAT: \( W \)
Effort Multipliers

4 Categories

• Product
• Platform
• Personnel
• Project
Example - Product Factor
RELY (Reqd. S/W Reliability)

Required Software Reliability (RELY)
Measures the extent to which the software must perform its intended function over a period of time. Ask: what is the effect of a software failure?

<table>
<thead>
<tr>
<th>RELY</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
<th>Extra High</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELY</td>
<td>slight inconvenience</td>
<td>low, easily recoverable losses</td>
<td>moderate, easily recoverable losses</td>
<td>high financial loss</td>
<td>risk to human life</td>
<td></td>
</tr>
</tbody>
</table>
Example - Product Factor RELY (Reqd. S/W Reliability)
# Accuracy Results:

## Effort Prediction

<table>
<thead>
<tr>
<th></th>
<th>Before Stratification By Organization</th>
<th>After Stratification By Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRED(.20)</td>
<td>46%</td>
<td>49%</td>
</tr>
<tr>
<td>PRED(.25)</td>
<td>49%</td>
<td>55%</td>
</tr>
<tr>
<td>PRED(.30)</td>
<td>52%</td>
<td>64%</td>
</tr>
</tbody>
</table>

## Schedule Prediction

<table>
<thead>
<tr>
<th></th>
<th>Before Stratification By Organization</th>
<th>After Stratification By Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRED(.20)</td>
<td>48%</td>
<td>52%</td>
</tr>
<tr>
<td>PRED(.25)</td>
<td>54%</td>
<td>61%</td>
</tr>
<tr>
<td>PRED(.30)</td>
<td>61%</td>
<td>62%</td>
</tr>
</tbody>
</table>
Presentation Outline:

• Steps in Software Estimation

  ➤ COCOMO II
    – Overview and Objectives
    – Post-Architecture Model
  ➤ Reuse Model
    – Other Ongoing Research and Plans

• Information Sources
COCOMO REUSE MODEL

- A nonlinear estimation model to convert adapted (reused or modified) software into equivalent size of new software

\[ AAF = 0.4(DM) + 0.3(CM) + 0.3(IM) \]

\[ ESLOC = \frac{ASLOC[AA + AAF(1 + 0.02(SU)(UNFM))]}{100}, AAF \leq 0.5 \]

\[ ESLOC = \frac{ASLOC[AA + AAF + (SU)(UNFM)]}{100}, AAF > 0.5 \]
COCOMO REUSE MODEL (cont)

ASLOC - Adapted Source Lines of Code
ESLOC - Equivalent Source Lines of Code
AAF - Adaptation Adjustment Factor
DM - Percent Design Modified. The percentage of the adapted software’s
design which is modified in order to adapt it to the new objectives and
environment.
CM - Percent Code Modified. The percentage of the adapted software’s
code which is modified in order to adapt it to the new objectives and
environment.
IM - Percent of Integration Required for Modified Software. The
percentage of effort required to integrate the adapted software into an
overall product and to test the resulting product as compared to the
normal amount of integration and test effort for software of comparable
size.
AA - Assessment and Assimilation effort needed to determine whether a
fully-reused software module is appropriate to the application, and to
integrate its description into the overall product description. See table.
SU - Software Understanding. Effort increment as a percentage. Only
used when code is modified. See table.
UNFM - Unfamiliarity. The programmer’s relative unfamiliarity with
the software which is applied multiplicatively to the software
understanding effort increment (0-1).
Presentation Outline:

- Steps in Software Estimation
  \[\text{COCOMO II}\]
  - Overview and Objectives
  - Post-Architecture Model
  - Reuse Model
  \[\text{Other Ongoing Research and Plans}\]
- Information Sources
Ongoing Research (PhD topics)

- Effects of Process Maturity on Effort (Brad Clark)
- COTS Model (Chris Abts)
- Cost/Quality Tradeoff Model (Sunita Devnani-Chulani)

Future Work:

- Stratify data based on Language Level and Application Type
- Effort distribution based on activities
- Enhancement of COCOMO II database to continuously update the model
Presentation Outline:

• Steps in Software Estimation
• COCOMO II
  – Overview and Objectives
  – Post-Architecture Model
  – Reuse Model
  – Other Ongoing Research and Plans

→ Information Sources
Information Sources:

- Phone: (213) 740-6470
- Email: cocomo-info@sunset.usc.edu
- Web site:
  http://sunset.usc.edu/COCOMOII/Cocomo.html
  - Affiliate Prospectus
  - Model Definition Manual (ver. 1.4)
  - Data Collection Form (ver. 1.6)
  - Java COCOMO
  - Little Expert COCOMO Calculator

Tech Report on COCOMO II.1997 calibration:
  http://sunset.usc.edu/TechRpts/electronicopy.html