COCOMO - Constructive Cost Modeling
The COCOMO model

- A family of empirical models based on analysis of projects of different companies
- Extended to cover different development processes and other aspects, such as quality (COQUALMO)
The COCOMO model

• COCOMO is based on a physical measure (source lines of code)

• Estimations become more precise as we move with development

• Estimation errors:
  – Initial estimations can be wrong by a factor of 4x
  – As we move with the development process, estimations become more precise
    (and the model takes into account more detailed parameters)
COCOMO: General Structure

\[ \text{OUTPUT} = A \cdot (\text{size})^B \cdot M \]

- All COCOMO models have the same basic structure
- OUTPUT can be effort or time
- The fundamental measure is code size (expressed in source lines of code)
- Code size has an exponential effect on effort and size (although very close to 1)
- Various adjustment factors are used to make the model more precise
COCOMO 81
COCOMO 81: Introduction

- Combination of three models with different levels of detail and complexity:
  - **BASIC**: quick estimation, early development stage
  - **INTERMEDIATE**: more accurate, needs some product characteristics, more mature development stage
  - **ADVANCED**: most detailed, requires more information

- In all COCOMO models:
  - 1 person month = 152 work-hours
  - SLOC is DSI (delivered source instructions)
    (only the code delivered to the client. E.g. unit testing, conversion code, utilities, ... do not count)
COCOMO 81: Types of Projects

- COCOMO 81 distinguishes among three different types of projects:
  
  - **ORGANIC**
    * small teams, familiar environment, well-understood applications, simple non-functional requirements (EASY)
  
  - **SEMI DETACHED**
    * project team may have experience mixture, system may have more significant non-functional constraints, organization may have less familiarity with application (HARDER)
  
  - **EMBEDDED**
    * tight constraints, including local regulations and operational procedures; unusual for team to have deep application experience (HARD)
COCOMO 81: Basic Model

\[ PM = A_{PM} \cdot (KSLOC)^{B_{PM}} \]

\[ TDEV = A_{TDEV} \cdot (PM)^{B_{TDEV}} \]

where

– KSLOC: thousands of delivered source lines of code
– M is equal to 1 (and therefore it does not appear in the formulae)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>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>Semi-detached</td>
<td>3</td>
<td>1.12</td>
<td>2.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.2</td>
<td>2.5</td>
<td>0.32</td>
</tr>
</tbody>
</table>

“spm” - ©2014 adolfo villafiorita - introduction to software project management
COCOMO exponential effect (vs. linear)
Application Example

• Estimation of 50 KDSI for an organic project
  – PM \(= 2.4 \times (50)^{1.05} \approx 146 \text{ mm}\)
  – TDEV \(= 2.5 \times (371.54)^{0.38} \approx 16 \text{ month}\)
  – Team \(= 371.54 / 23.69 \approx 9 \text{ person}\)

• The effect of different project parameters

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>B</th>
<th>PM</th>
<th>TDEV</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organic</td>
<td>2.4</td>
<td>1.05</td>
<td>2.5</td>
<td>0.38</td>
<td>146</td>
<td>16.6</td>
<td>8.8</td>
</tr>
<tr>
<td>Semi-detached</td>
<td>3</td>
<td>1.12</td>
<td>2.5</td>
<td>0.35</td>
<td>240</td>
<td>17.0</td>
<td>14.1</td>
</tr>
<tr>
<td>Embedded</td>
<td>3.6</td>
<td>1.2</td>
<td>2.5</td>
<td>0.32</td>
<td>394</td>
<td>16.9</td>
<td>23.3</td>
</tr>
</tbody>
</table>
Intermediate COCOMO

- It uses a more fine grained characterization, which uses **attributes** (effort multipliers) to take into account:
  - functional and non-functional requirements
  - project attributes

- The **effort multipliers** are organized in 4 classes and 15 sub-items.

- The importance of each attribute is **qualitatively evaluated** between 1 (very low) and 6 (extra high)

- Each value corresponds to multiplier, in the range $[0.7, 1.66]$ (multiplier $< 1$ implies reduced cost)

- All the values are multiplied together to modulate effort
COCOMO 81: Intermediate Model

\[
PM_{nominal} = A_{PM} \cdot (KSLOC)^{B_{PM}}
\]

\[
PM = PM_{nominal} \cdot \prod_{i=1}^{15} EM_i
\]

\[
TDEV = A_{TDEV}(PM)^{B_{TDEV}}
\]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>A</th>
<th>B</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>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>2.8</td>
<td>1.2</td>
<td>2.5</td>
<td>0.32</td>
</tr>
</tbody>
</table>
## Intermediate Model: Parameters

<table>
<thead>
<tr>
<th>Effort Adjustment Factors</th>
<th>Very_Low</th>
<th>Low</th>
<th><strong>Nominal</strong></th>
<th>High</th>
<th>Very_High</th>
<th>Extr_High</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required Software Reliability</td>
<td>RELY</td>
<td>0.75</td>
<td>0.88</td>
<td>1.00</td>
<td>1.15</td>
<td>1.40</td>
</tr>
<tr>
<td>Database Size</td>
<td>DATA</td>
<td>0.94</td>
<td>1.00</td>
<td>1.08</td>
<td>1.16</td>
<td></td>
</tr>
<tr>
<td>Product Complexity</td>
<td>CPLX</td>
<td>0.70</td>
<td>0.85</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
</tr>
<tr>
<td><strong>Computer Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Execution Time Constraints</td>
<td>TIME</td>
<td>1.00</td>
<td>1.11</td>
<td>1.30</td>
<td>1.66</td>
<td></td>
</tr>
<tr>
<td>Main Storage Constraints</td>
<td>STOR</td>
<td>1.00</td>
<td>1.06</td>
<td>1.21</td>
<td>1.56</td>
<td></td>
</tr>
<tr>
<td>Virtual Machine Volatility</td>
<td>VIRT</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td>Computer Turnaround Time</td>
<td>TURN</td>
<td>0.87</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td></td>
</tr>
<tr>
<td><strong>Personnel Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyst Capability</td>
<td>ACAP</td>
<td>1.46</td>
<td>1.19</td>
<td>1.00</td>
<td>0.86</td>
<td>0.71</td>
</tr>
<tr>
<td>Applications Experience</td>
<td>AEXP</td>
<td>1.29</td>
<td>1.13</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
</tr>
<tr>
<td>Programmer Capability</td>
<td>PCAP</td>
<td>1.42</td>
<td>1.17</td>
<td>1.00</td>
<td>0.86</td>
<td>0.70</td>
</tr>
<tr>
<td>Virtual Machine Experience</td>
<td>VEXP</td>
<td>1.21</td>
<td>1.10</td>
<td>1.00</td>
<td>0.90</td>
<td></td>
</tr>
<tr>
<td>Programming Language Experience</td>
<td>LEXP</td>
<td>1.14</td>
<td>1.07</td>
<td>1.00</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td><strong>Project Attributes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use of Modern Programming Practices</td>
<td>MODP</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.82</td>
</tr>
<tr>
<td>Use of Software Tools</td>
<td>TOOL</td>
<td>1.24</td>
<td>1.10</td>
<td>1.00</td>
<td>0.91</td>
<td>0.83</td>
</tr>
<tr>
<td>Required Development Schedule</td>
<td>SCED</td>
<td>1.23</td>
<td>1.08</td>
<td>1.00</td>
<td>1.04</td>
<td>1.10</td>
</tr>
</tbody>
</table>
Attributes

- Attributes:
  
  **PRODUCT** = RELY * DATA * CPLX
  **COMPUTER** = TIME * STOR * VIRT * TURN
  **PERSONNEL** = ACAP * AEXP * PCAP * VEXP * LEXP
  **PROJECT** = MODP * TOOL * SCED

- The impact of the parameters is between [0.09, 73.28]

- The PM (or team) estimate the values of parameters to predict actual effort

- Example:
  
  - If the “required software reliability” is low, the predicted effort is 0.88 of the one computed with the basic formula
**COCOMO 81: Detailed Model**

• The detailed model:
  – has more detailed multipliers for each development phase
  – organizes the parameters hierarchically, to simplify the computation of systems made of several modules

• Projects are organized in four phases:
  * Requirements Planning and Product Design (PRD)
  * Detailed Design (DD)
  * Code and Unit Test (CUT)
  * Integration Test (IT)

• EM are given and estimated per phase

• Phase data is then aggregated to get the total estimation
COCOMO 81: Advanced Model

- Example of parameter:

<table>
<thead>
<tr>
<th>Cost Driver</th>
<th>Rating</th>
<th>RPD</th>
<th>DD</th>
<th>CUT</th>
<th>IT</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACAP</td>
<td>Very Low</td>
<td>1.8</td>
<td>1.35</td>
<td>1.35</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>Nominal</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>0.75</td>
<td>0.9</td>
<td>0.9</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>Very High</td>
<td>0.55</td>
<td>0.75</td>
<td>0.75</td>
<td>0.7</td>
</tr>
</tbody>
</table>
COCOMO: Maintenance Phase

• The COCOMO model can also be applied to predict effort during system maintenance (system maintenance = small updates and repairs during the operational life of a system)

• Most of development parameters apply both to development and maintenance (some do not: SCED, RELY, MODP)

• One essential input is an estimation of the ACT (annual change traffic)
COCOMO 81: Maintenance

\[ ACT = \frac{\%\text{Added} + \%\text{Modified}}{100} \]

\[ PM = ACT \cdot PM_{nom} \cdot EAF_{maint} \]
COCOMO II
COCOMO II

- COCOMO II builds upon COCOMO 81 to take into account:
  - New development processes (e.g., spiral)
  - Increased flexibility in software development (e.g. reuse, automatic code generation)
  - Need for decision making with incomplete information
  - New data about projects (not really a need, rather an opportunity) (161 projects vs. 61)
COCOMO II: Models

- COCOMO II incorporates a range of sub-models that produce increasingly detailed software estimates.

- The sub-models in COCOMO II are:
  - **Application Composition Model.** For prototyping
  - **Early Design Model.** Used when requirements are available but design has not yet started.
  - **Post-architecture model.** Used once the system architecture has been designed and more information about the system is available.

- Moreover:
  - **Reuse model.** Used to compute the effort of integrating reusable components.
COCOMO II: Model Stages

Concept Ready
- Object Points

Requirements Ready
- Revised COCOMO (13 pars.)

Design Ready
- Revised COCOMO (23 pars.)

System Development
COCOMO II: ED and PA Models

\[ PM_{NS} = 2.94 \cdot (SIZE)^E \cdot \prod_{i=1}^{5} EM_i \]

\[ E = 0.91 + 0.01 \cdot \sum_{j=1}^{5} SF_i \]

\[ TDEV_{NS} = 3.67 \cdot (PM_{NS})^F \]

\[ F = 0.28 + 0.01 \cdot \sum_{j=1}^{5} SF_i \]

All constants can (need to) be adjusted with organization-dependent values.

(Strongly recommended: 2.94, effort multiplier and 3.67, schedule multiplier)

The difference between ED and PA is the number of parameters

The exponent depends on adjustment factors (rather than being just a constant as in COCOMO '81)
COCOMO II: Effort Multipliers

- From 7 (Early Design) to 17 (Post Architecture) according to the level of detail needed
- For instance:

<table>
<thead>
<tr>
<th>Early Design cost drivers</th>
<th>Post-Architecture cost drivers (Counterpart combined)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product reliability and complexity</td>
<td>RCPX</td>
</tr>
<tr>
<td>Required reuse</td>
<td>RUSE</td>
</tr>
<tr>
<td>Platform difficulty</td>
<td>PDIF</td>
</tr>
<tr>
<td>Personnel capability</td>
<td>PERS</td>
</tr>
<tr>
<td>Personnel experience</td>
<td>PREX</td>
</tr>
<tr>
<td>Facilities</td>
<td>FCIL</td>
</tr>
<tr>
<td>Required Development Schedule</td>
<td>SCED</td>
</tr>
</tbody>
</table>

Source: [http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf](http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf)
COCOMO II: Scale Factors

- The exponent is computed by providing qualitative answers to the following factors:
  - **Precedentedness**: how novel the project is for the organization
  - **Flexibility**: development flexibility (e.g. rigidity of compliance to requirements)
  - **Design/Risk**: thoroughness of design and risk resolution
  - **Team Cohesion**
  - **Process Maturity**: maturity with respect to the CMMI questionnaire

<table>
<thead>
<tr>
<th>Scale Factors (Wi)</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 average of “Yes” answers to CMM Maturity Questionnaire</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: [http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf](http://www.ifi.uzh.ch/req/courses/seminar_ws02/reports/Seminar_4.pdf)
Algorithmic Techniques
Conclusions
COCOMO Considerations

• A series of progressively more complex models

• COCOMO computes both $D$ and $E$ and manpower is derived from $D$ and $E$.
  (we often estimate $E$, decide $M$, and compute only $D$)

• Project cost estimates may be self-fulfilling: the estimate defines the budget and the product is adjusted to meet the budget

• A precise application requires organizations to setup their own measurement programs (to fine-tune parameters)

• Models need to be adapted to changing technologies and the technology changes fast... it might be difficult to keep it up to date
Algorithmic Techniques: Recap

• Based on system characteristics and productivity metrics collected about past projects

• Different models
  – Function Points:
    Req \(\leftarrow\) UFP \(\rightarrow\) FP, MM/FP
    Req \(\leftarrow\) UFP \(\rightarrow\) SLOC/UFP and COCOMO
  – Object Points
    Screens, Reports, Modules \(\leftarrow\) NOP, NOP/Month
  – COCOMO
    SLOC \(\leftarrow\) PDEV, TDEV \(\rightarrow\) Team Size = PDEV / TDEV
The Improvement “factory”

Development

Product Specification

Estimation

estimation not ok

Development

Measurement Program

Estimations Collection

Estimations Collection

Actual Data Collection

Improvement Program

Model Tuning

Organization Tuning