Software Cost Estimation

J. C. Huang
Department of Computer Science
University of Houston
Houston, TX 77204-3010

On Precision

It is the mark of an instructed mind to rest satisfied with the degree of precision which the nature of subject admits, and not to seek exactness when only an approximation of the truth is possible... -Aristotle

An Observation

Estimation of resources, cost, and schedule for a software development effort requires experience, access to good historical information, and the courage to commit to quantitative measures when qualitative data are all that exist.

Another Observation

Estimation for software project carries inherent risk because of uncertainty in project complexity, project size, and structure (of requirements and problem to be solved).

Major Factors

Major factors that influence software cost:
• product size and complexity
• programmer ability
• available time
• required reliability
• level of technology

Three Approaches

Estimation can be done by using
• experience and historical data
• decomposition techniques
• empirical models
Decomposition Techniques

To obtain an estimation, we can decompose the problem to be solved, or decompose the process.

Decomposition

Decomposition should be done in such a way that
1. size can be properly estimated,
2. cost or effort required for each component can be accurately estimated,
3. the team's ability to handle the components is well known, and
4. the estimated values will be relatively unaffected by changes to the requirements.

Problem-Based Estimation

1. Based on the software scope, decompose the software into problem functions that can be estimated individually.
2. Estimate LOC or FP of each function.
3. Make optimistic ($s_{op}$), most likely ($s_m$), and pessimistic ($s_{pe}$) estimates for each item. Then compute the expected value:
   \[ EV = \frac{s_{op} + 4s_m + s_{pe}}{6} \]
4. Apply baseline productivity metrics to compute estimated cost or effort.

Process-Based Estimation

1. Decompose the process into a set of tasks or activities.
2. Estimate the cost or effort required for each.

Empirical Estimation Models

- An estimation model provides empirically derived formulas to predict effort as a function of LOC or FP.
- The data used to support these models are derived from a limited sample. Thus no model is appropriate for all classes of software.

Structure of Estimation Model

\[ E = A + BX^C \]

where A, B, and C are empirically derived constants, E is the effort in person months, and X is the estimation variable, either in LOC or FP.
LOC-based Model

Walston-Felix Model \( E = 5.2(KLOC)^{0.91} \)
Bailey-Basili Model \( E = 5.5 + 0.73(KLOC)^{1.05} \)
Boehm Model (simple) \( E = 3.2(KLOC)^{1.05} \)
Doty Model for KLOC > 9 \( E = 5.288(KLOC)^{1.047} \)

FP-based model

Albrecht and Gaffney Model \( E = -13.39 + 0.0545(FP) \)
Kemerer Model \( E = 60.62 + 7.728(FP)^{3.10^{-8}} \)
Matson, Barnett & Mellichamp Model \( E = 585.7 + 15.12(FP) \)

COCOMO

The COntstructive COst MOdel

It is LOC-based.

There are three models:
- basic,
- intermediate, and
- advanced.

Three Classes of Software Project

- **Organic** -- a relatively small simple project in which small teams with good application experience work to a set of less than rigid requirements.
- **Semi-detached** -- an intermediate project in which teams with mixed experience must meet a mix of rigid and less than rigid requirements.
- **Embedded** -- a project that must meet tight hardware, software, and operational constraints.

COCOMO Model (continued)

The basic equations
\[
E = a(KLOC)^b \\
T = cE^d
\]
where \( E \) is the effort required in person-months, \( T \) is the required development time in chronological months, KLOC is the estimated size of software in thousand lines of delivered source code. The constants \( a, b, c, \) and \( d \) are listed below:
The COCOMO (continued)

Effort equation for the intermediate model:

\[ E = a(KLOC)^b(EAF) \]

where EAF is the effort adjustment factor that ranges from 0.9 to 1.4, and constants a and b are

<table>
<thead>
<tr>
<th>type of project</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>organic</td>
<td>3.2</td>
<td>1.05</td>
</tr>
<tr>
<td>semi-detached</td>
<td>3.0</td>
<td>1.12</td>
</tr>
<tr>
<td>embedded</td>
<td>2.8</td>
<td>1.20</td>
</tr>
</tbody>
</table>

The Software Equation

\[ E = ((LOC)B^{0.333}/P)^{(1/t^4)} \]

where

- E = effort in person-months
- t = project duration in months
- B = special skill factor ranging from 0.16 to 0.39
- P = productivity parameter

(ref. www.qsm.com)

Major Factors

Major factors that influence software cost:

- product size and complexity
- programmer ability
- available time
- required reliability
- level of technology

Product Complexity

Three categories of products:

- Application programs: those developed in the environment by a language compiler, such as C++.
- Utility programs: those written to provide user processing environments and make sophisticated use of the operating system facilities.
- System programs: those interact directly with hardware, and often involve concurrent processing with time constraints.
**Required Effort**

Given KDSI, thousand lines of deliverable code, Required programmer-months:

- application programs: \( PM = 2.4 \times (KDSI)^{1.05} \)
- utility programs: \( PM = 3.0 \times (KDSI)^{1.12} \)
- system programs: \( PM = 3.6 \times (KDSI)^{1.20} \)

**Required Development Time**

Required development time:

- application programs: \( TDev = 2.5 \times (PM)^{0.38} \)
- utility programs: \( TDev = 2.5 \times (PM)^{0.35} \)
- system programs: \( TDev = 2.5 \times (PM)^{0.32} \)

**Major Factors**

Major factors that influence software cost:
- product size and complexity
- programmer ability
- available time
- required reliability
- level of technology
How programmers spend their time.

- Writing programs: 13%
- Reading programs and manuals: 16%
- Job communications: 32%
- Personal: 13%
- Miscellaneous: 15%
- Training: 6%
- Mail: 5%

Based on Bell Lab Study conducted in 1964 on 70 programmers.

Programmer’s Ability

Variations in programmers abilities

<table>
<thead>
<tr>
<th>performance measure</th>
<th>worst/best ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>program #1</td>
</tr>
<tr>
<td>debugging hours</td>
<td>28:1</td>
</tr>
<tr>
<td>CPU time</td>
<td>8:1</td>
</tr>
<tr>
<td>coding hours</td>
<td>16:1</td>
</tr>
<tr>
<td>program size</td>
<td>6:1</td>
</tr>
<tr>
<td>run time</td>
<td>5:1</td>
</tr>
</tbody>
</table>

Programmer’s Ability (continued)

By eliminating extreme performance in both directions, a variability of 5 to 1 in programmer productivity can be expected.

Available Time

Software projects require more total effort if development time is compressed or expanded from the optimal time.

Available Time (continued)

According to Putnam, a schedule compression of 0.86 will increase required staff by a factor of 1.82.

Available Time (continued)

It is commonly agreed that there is a limit beyond which a software project cannot reduce its schedule by adding more personnel and equipment. This limit occurs roughly at 75% of the normal schedule.
COCOMO Effort Multipliers

Product attributes
- Required reliability: 0.75 to 1.40
- Data-base size: 0.94 to 1.16
- Product complexity: 0.70 to 1.65

COCOMO Effort Multipliers (continued)

Computer attributes
- Execution time constraint: 1.00 to 1.66
- Main storage constraint: 1.00 to 1.56
- Virtual machine volatility: 0.87 to 1.30
- Computer turn-around time: 0.87 to 1.15

Personnel attributes
- Analyst capability: 1.46 to 0.71
- Programmer capability: 1.42 to 0.70
- Applications experience: 1.29 to 0.82
- Virtual machine experience: 1.21 to 0.90
- Programming language experience: 1.14 to 0.95

Staffing Level Estimation

A software project typically starts with a small group of capable people to do planning and analysis, a larger, but still small group to do architectural design. The size of required personnel increases in each successive phase, peaks at the implementation and system testing phase, and decreases in the maintenance phase.

Staffing-Level Estimation

The personnel level of effort required throughout the life cycle of a software product can be approximated by the following equation, which describes a Rayleigh curve:
Staffing-Level Estimation (continued)

\[ FSP = PM \frac{0.15T_{dev} + 0.7t}{0.25(T_{dev})^2} e^{-\frac{(0.15T_{dev} + 0.7t)^2}{0.5(T_{dev})^2}} \]

where \( FSP \) is the no. of full-time software personnel required at time \( t \),

\( PM \) is the estimated programmer-months for product development, excluding planning and analysis, and

\( T_{dev} \) is the estimated development time.

Skills most lacking in entry level programmers

- Express oneself clearly in English
- Develop and validate software requirements and specifications
- Work within applications area
- Perform software maintenance
- Perform economic analyses
- Work with project management techniques
- Work in group