Software Sizing
Lines of Code and Beyond

Air Force Cost Analysis Agency

Corinne Wallshein
June 2009
Presentation Overview

- About software sizing…
  - Meaning
  - Sources
  - Importance
  - Description
  - Models
  - Current challenges
  - Conclusion
What is software size?

- Software sizing is a work sizing abstraction
- Determining mental efforts and social interactions in development, production and maintenance:
  - What is the purpose of the software?
    - Who cares about it?
  - What is the data?
    - Is the data correct?
  - What should the algorithms do?
    - Are the algorithms correct?
  - Who will use it?
    - Are the needed safeguards in place?
Sources for software size

- Cost Analysis Requirements Document
- Software schematic diagrams
- Software requirements specifications
- Sub-system requirements specifications
- Analogous (completed) system sizes
“Software size can be an important component of a productivity computation, a cost or effort estimate, or a quality analysis. More importantly, a good software size measure could...lead to a better understanding of the value being delivered by a software application...there is no agreement among professionals as to the right units for measuring software size or the right way to measure within selected units.”


STN 11-3, October 2008: New Directions in Software Estimation
Usage: cost driver for estimation


\[ \text{ESTIMATES} = \text{PROJECT SIZE} \times \text{PROJECT ATTRIBUTES} \]

Barry Boehm, et. al., “COCOMO Suite Methodology and Evolution,” 2005

\[ PM = A \times (\sum \text{Size})^B \times \prod(EM) \]

PM = person months.
A = calibration factor.
Size = measure(s) of functional size of a software module that has an additive effect on software development effort.
B = scale factor(s) that has an exponential or nonlinear effect on software development effort.
EM = effort multipliers that influence software development effort.

Each factor in the equation can be represented by a single value or multiple values.


*Integrity - Service - Excellence*
Measuring software size

- Stand-alone software
  - One software language
  - One software developer
  - One user
  - One component

- Integrated software
  - Multiple languages
  - Multiple developers
  - Multiple users
  - Multiple components

Past
- Easier to measure

Present
- Harder to measure

Integrity - Service - Excellence
Advent of new development paradigm: Object-Oriented (OO) coding

Source: Air Force Cost Analysis Agency (n = 220 military systems)
**Software languages by generation**

1. **1st Generation**
   - Machine Language
   - “0’s and 1’s” *Circuits*

2. **2nd Generation**
   - Assembly
   - “1 + 1 = 10; 0 + 0 = 0; 1 + 0 = 1; 0 + 1 = 1”
   - *Fix logic structures*

3. **3rd Generation**
   - Ada
   - Jovial
   - C series {C, C+, C++, C#}
   - COBOL
   - FORTRAN
   - Java
   - LISP
   - BASIC
   - PASCAL
   - ALGOL
   - Visual Basic
   - “Hello World” *User-friendly programming*

4. **4th Generation**
   - Better, faster, cheaper
   - Clipper
   - Cold Fusion
   - PowerBuilder
   - IBM Rational EGL
   - PL/SQL
   - R
   - SAS
   - IDL
   - FOCUS
   - RPG-II

5. **5th Generation**
   - Prolog
   - OPS5
   - Mercury
   - *Theory of constraints*

**Domain Languages - Specialized Specific Purpose** Languages like: OCL, QVT, CFML

**Integrity - Service - Excellence**
Software size units by decade

- Early user-computer interaction, before 1980
  - Punch cards
  - Lines of Code (LOC)
- Client-server environments, from 1980 to 1990
  - Function points (FP)
- Modular software, object-oriented methods, from 1990 to 2000
  - Object points
  - Documentation
- Component network software (systems of systems), from 2000 to today
  - Other size measures
    - Requirements
    - Use cases
    - UML diagrams
    - RICE objects
    - Integration of COTS, GFS, GOTS and OSS
Sizing method characterization

- Based on expert judgment
  - Pair-wise comparison
  - Analogous systems
  - Case based reasoning

- Based on measurable items
  - Supporting documentation
  - Other artifacts
  - Software documentation
  - Prototypes
  - Previous increments
  - Legacy system

Adapted from Pfleeger, Wu, and Lewis, *Software Cost Estimation and Sizing Methods*, 2005
Sizing units, models & techniques

SIZING MODELS / TECHNIQUES

SOURCE LINES OF CODE (SLOC)
- Algorithmic
- Analogy
- Expert-based
- Engineering Level

FUNCTION POINTS
- Traditional
- Variants
- Real-Time Adaptations

OBJECT POINTS
- Banker
- PRICE
- SEER-SEM object sizing

OTHER SIZING
- RICE
- Use Cases
- UML Diagrams

Adapted from Donald Reifer’s *Software Management*, published in 2002

Integrity - Service - Excellence
Software size and types

Size

- Lines of Code (LOC)
  - Source LOC (SLOC)
  - Thousands of LOC (KLOC)
  - Thousands of SLOC (KSLOC)
  - Effective SLOC (ESLOC)
  - Equivalent SLOC (ESLOC)
  - Delivered Source Instructions (DSI)
  - Maintainable Lines of Instruction (MLI)

Type

- New code
- Auto-generated code
  - Computer-aided software engineering (CASE) tools
  - Integrated development environment (IDE) tools
  - Model Driven Architecture (MDA) tools
- Adapted, leveraged, or reused code
  - Modified code
  - Unmodified code
  - Commercial-off-the-shelf (COTS)
  - Government furnished code (GFS)
  - Government-off-the-shelf (GOTS)
  - Open Source Software (OSS)

As Steve McConnell says in *Software Estimation: Demystifying the Black Art*, published in 2006:

“LOC…is a terrible way to measure software size, except that all the other ways to measure size are worse.”

*Integrity - Service - Excellence*
## SLOC comparison

<table>
<thead>
<tr>
<th>LOGICAL STATEMENTS</th>
<th>PHYSICAL LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROS</strong></td>
<td><strong>CONS</strong></td>
</tr>
<tr>
<td>Excludes dead code</td>
<td>Can be difficult to count</td>
</tr>
<tr>
<td>Excludes blanks and comments</td>
<td></td>
</tr>
<tr>
<td>Used in a number of estimating tools</td>
<td></td>
</tr>
<tr>
<td>Can be mathematically converted to FPs</td>
<td>May be ambiguous for reuse</td>
</tr>
<tr>
<td></td>
<td>Poor choice for full life cycle studies</td>
</tr>
<tr>
<td></td>
<td>Ambiguous for some visual languages</td>
</tr>
<tr>
<td></td>
<td>May be erratic for direct conversion to physical LOC metrics</td>
</tr>
<tr>
<td></td>
<td>Not extensively automated</td>
</tr>
<tr>
<td><strong>PROS</strong></td>
<td><strong>CONS</strong></td>
</tr>
<tr>
<td>Are easy to count</td>
<td>May include dead code</td>
</tr>
<tr>
<td>Used in a number of estimating tools</td>
<td>May include blanks and comments</td>
</tr>
<tr>
<td></td>
<td>Ambiguous for mixed language projects</td>
</tr>
<tr>
<td></td>
<td>Ambiguous for reuse</td>
</tr>
<tr>
<td></td>
<td>Poor choice for full life cycle studies</td>
</tr>
<tr>
<td></td>
<td>Does not work for some visual languages</td>
</tr>
<tr>
<td></td>
<td>Erratic for FP conversion</td>
</tr>
<tr>
<td></td>
<td>Erratic for logical statements conversion</td>
</tr>
<tr>
<td></td>
<td>Extensive automation</td>
</tr>
</tbody>
</table>

## Model size inputs

### COCOMO II Size Inputs

<table>
<thead>
<tr>
<th>New Software</th>
<th>SEER-SEM Size Inputs</th>
<th>True S Size Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Size</td>
<td>New Size</td>
<td>New Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>New Size Non-executable</td>
</tr>
</tbody>
</table>

### Adapted Software

<table>
<thead>
<tr>
<th>Adapted Size</th>
<th>Pre-existing Size</th>
<th>True S Size Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>% Design Modified (DM)</td>
<td>Redesign Required %</td>
<td>Adapted Size Non-executable</td>
</tr>
<tr>
<td>% Code Modified (CM)</td>
<td>Reimplementation Required %</td>
<td>% of Design Adapted</td>
</tr>
<tr>
<td>% Integration Modified (IM)</td>
<td>Retest Required %</td>
<td>% of Code Adapted</td>
</tr>
<tr>
<td>Assessment and Assimilation (AA)</td>
<td></td>
<td>% of Test Adapted</td>
</tr>
<tr>
<td>Software Understanding (SU)</td>
<td></td>
<td>Reused Size</td>
</tr>
<tr>
<td>Programmer Unfamiliarity (UNFM)</td>
<td></td>
<td>Reused Size Non-executable</td>
</tr>
<tr>
<td>Deleted Size</td>
<td></td>
<td>Deleted Size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code Removal Complexity</td>
</tr>
</tbody>
</table>

### Automatically Translated and Generated Code

<table>
<thead>
<tr>
<th>Adapted SLOC</th>
<th>Auto Generated Code Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Translation Productivity</td>
<td>Auto Generated Code Size Non-executable</td>
</tr>
<tr>
<td>% of Code Reengineered</td>
<td>Auto Translated Code Size</td>
</tr>
<tr>
<td></td>
<td>Auto Translated Size Non-executable</td>
</tr>
</tbody>
</table>

---

*From draft AFCAAA Software Cost Estimation Manual, as of June 2009*

*Integrity - Service - Excellence*
### COCOMO II size quantification

<table>
<thead>
<tr>
<th>Code Category</th>
<th>Percent Design Modified (DM) for new objectives and environment</th>
<th>Percent Code Modified (CM) for new objectives and environment</th>
<th>Percent of Integration Required for Adapted Software (IM) into an overall product</th>
<th>Assessment and Assimilation (AA) for reuse or integration of existing software into application</th>
<th>Software Understanding (SU) of the existing software by programmer</th>
<th>Programmer Unfamiliarity (UNFM) with the software</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>0% to 100%, normally &gt; 0%</td>
<td>0% to 100%, normally &gt; DM, must be &gt; 0%</td>
<td>0% to 100%, often moderate, can be &gt; 0%</td>
<td>0% to 8%</td>
<td>0% to 50%</td>
<td>0 - 1</td>
</tr>
<tr>
<td>Adapted</td>
<td>0% to 100%, normally &gt; 0%</td>
<td>0% to 100%, normally &gt; DM, must be &gt; 0%</td>
<td>0% to 100%, often moderate, can be &gt; 0%</td>
<td>0% to 8%</td>
<td>0% to 50%</td>
<td>0 - 1</td>
</tr>
<tr>
<td>Reused</td>
<td>0%</td>
<td>0%</td>
<td>0% to 100%, rarely 0%, could be very small</td>
<td>0% to 8%</td>
<td>Not applicable</td>
<td></td>
</tr>
</tbody>
</table>

From draft AFCAAA Software Cost Estimation Manual, as of June 2009
FP Standards

- Multiple standards for Function Point metrics
  - International Function Point Users Group (IFPUG) method’s functional size component
    - ISO/IEC 20926
  - Netherlands Software Metrics Association (NESMA) functional sizing measurement
    - ISO/IEC 24570
  - Common Software Measurement International Consortium (COSMIC) functional size metric
    - ISO/IEC 19761
  - MkII Function Point Analysis method
    - ISO/IEC 20968

ISO/IEC JTC1/SC7 decision to “Let the market decide” at http://www.cosmicon.com/historycs.asp
Accessed on May 28, 2009
### Unadjusted FP Conversion Factors to get Adjusted FP

<table>
<thead>
<tr>
<th>Type of Component</th>
<th>Complexity of Component</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>External inputs</td>
<td>___ x 3 =</td>
</tr>
<tr>
<td>External outputs</td>
<td>___ x 4 =</td>
</tr>
<tr>
<td>External queries</td>
<td>___ x 3 =</td>
</tr>
<tr>
<td>Internal logic files</td>
<td>___ x 7 =</td>
</tr>
<tr>
<td>External logic files</td>
<td>___ x 5 =</td>
</tr>
</tbody>
</table>

From Barry Boehm et al., *Software Cost Estimation with COCOMO II*, 2000
## SLOC per Unadjusted FP (UFP)

Capers Jones's *Programming Languages and Levels* – Languages identical to Barry Boehm’s

<table>
<thead>
<tr>
<th>PROGRAMMING LANGUAGE</th>
<th>LEVEL</th>
<th>AVERAGE SLOC PER UNADJUSTED FP</th>
</tr>
</thead>
<tbody>
<tr>
<td>5th Generation default</td>
<td>70</td>
<td>5</td>
</tr>
<tr>
<td>4th Generation default</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>3rd Generation default</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>2nd Generation default</td>
<td>3</td>
<td>107</td>
</tr>
<tr>
<td>1st Generation default</td>
<td>1</td>
<td>320</td>
</tr>
<tr>
<td>Machine language</td>
<td>0.5</td>
<td>640</td>
</tr>
</tbody>
</table>

*Integrity - Service - Excellence*
Impact

- Software size, effort, quality and schedule interrelated in numerous studies
- Different metrics for different application domains
  - Advent of domain languages
  - Advent of new development paradigms
- Proliferation of metrics and better ways to size software in numerous studies
- SLOC and FP are still the most common metrics

John Bailey, 2006 email: “...Too many projects estimate in function points or use cases, then afterwards compute their productivity using LOC...we have no idea whether they produced the expected number of functions so no calibration...is accomplished...”
Current challenges

- Comparing estimates when size metrics differ
- Translation tables between different size units
  - Established SLOC to UFP translations
  - Not established for other size units
    - AFCAA building RICE to requirements translation
- One DOD program building a specifications, business processes, and test plans translation
As sizing units change, estimation practices need to change with them.

Default size metrics are:
- Source Lines of Code (SLOC)
  - Physical SLOC
  - Logical SLOC
- Function Points (FP)
- Standards

Impact of new development paradigms: non-OO vs OO
Impact of software type: new, modified and reused
Impact of available components, including COTS, GFS, GOTS and OSS
Thank you for your attention