Counting Software Size: Is It as Easy as Buying A Gallon of Gas?

October 22, 2008

NDIA - 11th Annual Systems Engineering Conference
Lori Vaughan and Dean Caccavo
Northrop Grumman Mission Systems
Office of Cost Estimation and Risk Analysis
Agenda

• Introduction
• Standards and Definitions
• Sample
• Implications
• Summary
Introduction

- In what ways is software like gasoline?
- In what ways is software not like gasoline?
Industry Data Suggests...

- A greater percentage of the functions of the DoD Weapon Systems are performed by software

<table>
<thead>
<tr>
<th>Weapon System</th>
<th>Year</th>
<th>% of Functions Performed in Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-4</td>
<td>1960</td>
<td>8</td>
</tr>
<tr>
<td>A-7</td>
<td>1964</td>
<td>10</td>
</tr>
<tr>
<td>F-111</td>
<td>1970</td>
<td>20</td>
</tr>
<tr>
<td>F-15</td>
<td>1975</td>
<td>35</td>
</tr>
<tr>
<td>F-16</td>
<td>1982</td>
<td>45</td>
</tr>
<tr>
<td>B-2</td>
<td>1990</td>
<td>65</td>
</tr>
<tr>
<td>F-22</td>
<td>2000</td>
<td>80</td>
</tr>
</tbody>
</table>

Source: PM Magazine

System Functionality Requiring Software

- Increased amount of software in Space Systems and DoD Weapon Systems – Ground, Sea and Space/Missile
- Increased amount of software in our daily lives:
  - Cars, Cell Phones, iPod, Appliances, PDAs...

The amount of software used in DoD weapon systems has grown exponentially
Is There a Standard for Counting Software?

• Since, increasing percent of our DoD systems are reliant on software we need to be able to quantify the software size
  - Historical data collection
  - Estimation and planning
  - Tracking and monitoring during program performance

• Software effort is proportional to the size of the software being developed
  - *SW Engineering Economics* 1981 by Dr. Barry Boehm

• “Counting” infers there is a standard

• Experience as a prime integrator
  - Do not see a standard being followed

There are software counting standards but the message isn’t out or it is not being followed consistently
Source Line of Code definition

From Wikipedia, the free encyclopedia

“Source lines of code (SLOC) is a software metric used to measure the size of a software program by counting the number of lines in the text of the program’s source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or effort once the software is produced.”

• Variety of Software Languages in which source code is written
  – A to Z
    • Ada, Assembler, C, C++, C#, COBOL, Fortran, Java, JavaScript, Pascal, Perl and SQL to name just a few
Source Line of Code definition: Physical and Logical

- Software Engineering Institute (SEI) has developed checklist as part of a system of definition checklists to support measurement definitions. Software Size Measurement: A Framework for Counting Source Statements.

- Physical SLOC: One physical SLOC is corresponding to one line starting with the first character and ending by carriage return or an end of file marker of the same line and which excludes the blank and comment line.

- Logical SLOC: Lines of code intended to measure “statements” which normally terminated with a semicolon or a carriage return. Logical SLOC are not sensitive to format, style and conventions, but they are language dependent.
Source Line of Code Samples

```
for (i=0; i<100; ++i) printf("hello"); /* How many lines of code is this? */
   - 1 Physical Line of Code LOC
   - 2 Logical Lines of Code LOC (for statement and `printf` statement)
   - 1 Comment Line
```

```
for (i=0; i<100; ++i)
{
    printf("hello");
}
/* Now how many lines of code is this? */
   - 4 Physical Lines of Code LOC (Is placing braces work to be estimated?)
   - 2 Logical Line of Code LOC (What about all the work writing non-statement lines?)
   - 1 Comment Line (Tools must account for all code and comments regardless of comment placement.)
```

Note the logical count is independent of the programming style and conventions
Implications of SLOC Counts

Typical Simplified Software Cost Estimation Formula

\[ \text{Size} \div \text{Productivity (Size / Time unit)} = \text{Effort / Cost} \]

- Suppose you were given this simplified software cost formula and you received data from two separate contractors and were asked to determine relative development costs?

- What would that impact?
  - Size
  - Productivity
  - Hours
### Implication Illustration – Historical

**Contractor A**

<table>
<thead>
<tr>
<th>Physical Coordinate Perspective</th>
<th>Logical Coordinate Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLOC Count</td>
<td>SLOC Count</td>
</tr>
<tr>
<td>500 KSLOC</td>
<td>312.5 KSLOC</td>
</tr>
<tr>
<td>Effort</td>
<td>Effort</td>
</tr>
<tr>
<td>2500 Person Months (PM)</td>
<td>2500 (PM)</td>
</tr>
<tr>
<td>Productivity</td>
<td>Productivity</td>
</tr>
<tr>
<td>500 KSLOC ÷ 2500 PM = 200 ESLOC/PM</td>
<td>312.5 KSLOC ÷ 2500 PM = 125 ESLOC/PM</td>
</tr>
</tbody>
</table>

Without understanding the basis of the Software SLOC count, it looks like Contractor A is more productive. Is this correct?
**Implication Illustration - Estimate Comparison**

<table>
<thead>
<tr>
<th>Contractor A</th>
<th>Contractor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Size</td>
<td>Estimated Size</td>
</tr>
<tr>
<td>600 KSLOC</td>
<td>600 KSLOC</td>
</tr>
<tr>
<td>Historical Productivity</td>
<td>Historical Productivity</td>
</tr>
<tr>
<td>200 ESLOC/PM</td>
<td>125 ESLOC/PM</td>
</tr>
<tr>
<td>Estimated Effort</td>
<td>Estimated Effort</td>
</tr>
<tr>
<td>3,000 PM</td>
<td>4,800 PM</td>
</tr>
<tr>
<td>Estimated Cost</td>
<td>Estimated Cost</td>
</tr>
<tr>
<td>3,000 PM X $20K = $60 M</td>
<td>4,800 PM X $20K = $96 M</td>
</tr>
</tbody>
</table>
Attributes of a good code counter
- Non Proprietary
- Available to the public
- Platform independent
- Support multiple programming languages
- Count both physical and logical SLOC
- Limited Public License or “Copyleft” type agreement

http://sunset.usc.edu/research/CODECOUNT/

Sample 1.0::SLOC Counting

The Totals

<table>
<thead>
<tr>
<th>Total Lines</th>
<th>Blank Lines</th>
<th>Comments Lines</th>
<th>Compiler Whole Embedded Lines</th>
<th>Data Exec. Direct. Decl. Instr. Lines</th>
<th>Number of Files</th>
<th>File SLOC Type</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>33991</td>
<td>3855</td>
<td>8465</td>
<td>19</td>
<td>250 6815 14606</td>
<td>336</td>
<td>21671 CODE Physical</td>
<td></td>
</tr>
<tr>
<td>33991</td>
<td>3855</td>
<td>8465</td>
<td>19</td>
<td>250 2775 10667</td>
<td>336</td>
<td>13692 CODE Logical</td>
<td></td>
</tr>
<tr>
<td>1135</td>
<td>42</td>
<td>0</td>
<td>0</td>
<td>0 1093 0</td>
<td>47</td>
<td>1093 DATA Physical</td>
<td></td>
</tr>
</tbody>
</table>

Number of files successfully accessed......................... 383 out of 383

Ratio of Physical to Logical SLOC............................ 1.58
USC CSSE CodeCount™

• What programming languages are covered today
  – Ada, Assembler(s), Jovial, Pascal, COBOL, Fortran, MUL – Markup Language, Java, C/C++, C#, JavaScript, Visual Basic and Visual Basic Script

• What is included for each language
  – Read me file
  – Logical Standard (word table)
  – C source code of language specific counter
  – Sample input, source files and output file

USC Center for Systems and Software Engineering (CSSE) CodeCount™ suite supports many languages
Imagine Software Code Counting...

- As an integral part of your program’s change management system
- Improving your ability to perform Root cause Analysis
- Normalized code counts of existing software that are automatically uploaded to your historical database
- A historical repository of software size that could be used for estimation purposes and parametric model calibration
- Improving the representative nature of Parametric and Predictive Modeling
- Being consistent...
Summary

• Recognize underlying implications of Physical and Logical software sizing

• Assess appropriateness and magnitude of code count measurement

• Consider widespread standardization and integration into acquisition process
NORTHROP GRUMMAN
Defining the future