Physical and Functional Software Sizing

ICEAA SCEBoK
With input from Galorath, Cosmic, IFPUG and Nesma
Agenda

1. Overview of Lines of Code (LOC) Sizing
2. Overview of IFPUG & NESMA Function Points
3. Overview of COSMIC Function Points
Measurement of Source Lines of Code

Material provide by Galorath inc.
Many Viable Size Metrics

- SLOC is still viable for many items
  - Sometimes fails due to misunderstanding counting rules
- Functional sizes
- Use Cases
- Components
- Others

Source lines work for estimation in many environments...
Like any sizing metric...
If you use rigorous processes and definitions for counting / estimating them
Logical Vs Physical SLOC

Logical SLOC

- Language specific (but very similar & simple for most modern languages)
- Counts number of statements dependent on syntax
- Ignores blank lines & comments

Physical SLOC

- Count line feeds / carriage returns
- Not language specific
- Sensitive to formatting & style
- Often twice logical SLOC

Logical lines work best for effort estimation
Lines of Code Can Be Accurate If Definitions Are Consistent

Example C++ Program

```c++
extern double MessageMonitor(double dComplexity, double dSuccessRate);

//
// function: ExampleFunction
// purpose: Demonstrate counting of C code
// arguments: x [IN]: first argument
//            y [IN]: second argument
//            bar [IN]: third argument, an array of
// returns: return value

**/
double ExampleFunction(double x, double y, int* bar) {
    int n = (int) ((x + y) / 2);
    int SuccessfulAlert = 0;
    if (x < MessageMonitor(y, n))
        /* this is a comment */
        SuccessfulAlert = bar[n] + 5;
    else
        while (n > 0) {
            SuccessfulAlert += (int) MessageMonitor(x, n);
            n--;
        }
    return (x + SuccessfulAlert + bar[n]);
```
A Closer View....Lines of Code Counting Is Easy If You Know The Rules

Line Terminating Semicolons or Closing Curley Braces Make Automated Counting Simple For Many Languages
### SLOC Count Definitions

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Logical</th>
<th>NCSS</th>
<th>Physical</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In</td>
<td>In</td>
<td>In</td>
<td>Out</td>
</tr>
<tr>
<td>Executable</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

#### Nonexecutable

<table>
<thead>
<tr>
<th>Statement Type</th>
<th>Logical</th>
<th>NCSS</th>
<th>Physical</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In</td>
<td>In</td>
<td>In</td>
<td>Out</td>
</tr>
<tr>
<td>Declarations</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Compiler directives</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Comments</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>Blank lines</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>

Recommend USC (University of South California) code counter for counting existing software sizes.
Language used to develop a program will affect program size and effort

- Different languages require different numbers of lines to achieve the same functionality
- Different programming styles will result in different program sizes which do the same thing

Example of Ratios
(Statistical parameters not available)

<table>
<thead>
<tr>
<th>Source Language</th>
<th>Target Language</th>
<th>Gen 3GLs*</th>
<th>4GLs</th>
<th>Ada</th>
<th>Assembly</th>
<th>PL-1/Pascal</th>
</tr>
</thead>
<tbody>
<tr>
<td>General 3GLs</td>
<td>Gen 3GLs*</td>
<td>-65%</td>
<td>-18%</td>
<td>+22%</td>
<td>+248%</td>
<td>+338%</td>
</tr>
<tr>
<td>4GLs</td>
<td>4GLs</td>
<td>+185%</td>
<td>-71%</td>
<td>+22%</td>
<td>+435%</td>
<td>+19%</td>
</tr>
<tr>
<td>Ada</td>
<td>Ada</td>
<td>+185%</td>
<td>+248%</td>
<td>+22%</td>
<td>+1424%</td>
<td>+238%</td>
</tr>
<tr>
<td>Assembly</td>
<td>Assembly</td>
<td>+185%</td>
<td>+248%</td>
<td>+22%</td>
<td>+1424%</td>
<td>+238%</td>
</tr>
<tr>
<td>PL-1/Pascal</td>
<td>PL-1/Pascal</td>
<td>+185%</td>
<td>+248%</td>
<td>+22%</td>
<td>+1424%</td>
<td>+238%</td>
</tr>
</tbody>
</table>

* General 3GL includes Algol, Basic, FORTRAN, LISP, LOGO, C, C++, COBOL, JOVIAL, MODULA-2, PROLOG and other mixed languages.
Where SLOC Goes Wrong

- When code is autogenerated...
  - Count the source not the generated lines
- When code is generated from Graphical interface tool..
  - Use library of effective effort sizes
- When definitions are not known...
  - can convert from various definitions.. E.g. physical to logical ratios
- When people guess at SLOC rather than using a process and clear definitions
- When reuse is not properly accounted for...
- When Open Source or COTS is assumed to be free...
  - use effective size
- Not differentiating between logical and physical SLOC
Sometimes Size Exaggeration Occurs

**Actual Project Example:**
*Size Claimed = 1.6M Lines*
*True Size = 736,000 SLOC*

Why should we care: Poor sizing is a common reason for poor software estimates
Simplified Software Equation

Effort = a * fxsize^b

a is the scaler
b is the exponent
Fxsize is the estimated units of effective size
Effective Size Is What Should Be Used In Estimate, Not Total

Lines, Functions, Objects, Use Cases, Etc.

New

Pre-existing

Redesign
Reimplementation
Retest

Estimated Effective Size Units

Estimated Effective lines if using SLOC
Total vs. Estimated Effective Lines

- Many programs include a combination of new and existing software
  - New code needs to be designed, coded, tested, integrated
  - Existing code
    - May require some degree of design change, recoding, retesting, reintegration
    - May also require “reverse engineering” to discover how it works

- Total software size reflects the total functionality of the program
  - Newly developed functionality
  - Existing functionality (integrated)
  - COTS functionality

- Estimated “Effective Size” is more directly related to the amount of effort required to develop the software, considering
  - Redesign, reimplementation and/or retesting of existing code
  - Use of existing design or concept
  - Use of code generators, GUI Builders, etc.
Size Growth

• Whether SLOC or other size measures software usually grows from initial estimates due to
  • Bias
  • Functional growth
• Using a range to describe the estimated size can mitigate this.
SLOC: Conclusions

• Lines of code are viable for expressing software scope
• Takes care and rigor
• Don’t let people just guess
Agenda

1. Overview of Lines of Code (LOC) Sizing
2. Overview of IFPUG & NESMA Function Points
3. Overview of COSMIC Function Points
ISO Standards on Functional Size Measurement (FSM) Methods

- **Generic - ISO 14143:** FSM methods must:
  - Be based on a representation of the Functional User Requirements from the perspective of the users.
  - Not be derived from the effort required to develop the software.
  - Be independent of the methods used to develop the software being measured.
  - Be independent of the physical components of the software being measured.
  - Exclude software quality & technology characteristics.

- **Specific FSM recognized by ISO & IEC (International Electronic Commission):**
  - ISO 19761: COSMIC Function Points
  - ISO 20926: IFPUG Function Points
  - ISO 24570: NESMA Function Points
  - ISO 20968: MKII Function Points
  - ISO 29981: FISMA Function Points
Introduction to the IFPUG-NESMA Function Point sizing methods

Material provided by IFPUG and Nesma
What Are Function Points

- Function points are a normalized functional size measure first introduced in 1979

- Standard method for quantifying the software deliverable based upon the user view

- User View is the Functional User Requirements as perceived by the user

- User is any person or thing that communicates or interacts with the software at any time – person, system, application

- Projects typically have functional and non-functional requirements

- When we count a project, we are sizing the functional requirements (adds, changes and deletes) to an application

- Professional counters earn the Certified Function Point Specialist (CFPS) designation through testing and experience
What is function point analysis (FPA)?

A consistent method to quantify the amount of requested functionality that an information system offers to its users.

Main advantage

Objective (ISO/IEC standard), repeatable, verifiable and therefore defensible measurement of functionality:

- IFPUG: ISO 20926
- NESMA: ISO 24570

Therefore it is possible to use functional size in software project estimation, project control, productivity measurement, benchmarking, pricing and contracting.
Function
FPA bases itself on the functionality that an application provides to a user. Because users see only the "outside" or the boundary of an application, FPA examines the specifications that describe the application's exchange of information with its environment. Functionality is derived from incoming and outgoing information flows (these can be both data or control information), as well as from the logical files that an application contains or uses. The functionality of an application is measured by identifying data functions and transactional functions.

Point
After identifying the user functions (user transactions and logical files), the complexity of each of these is determined by using the counting guidelines. Each user function is worth a number of function points, depending on its complexity.

Analysis
Refers to the skills and expertise an FPA analyst needs to correctly identify the user transactions and logical files in scope of a certain measurement and the expertise and skills to determine the right complexity of each of the user transactions and logical files identified.

Before an FPA is carried out, first the purpose of the FPA should be determined and the scope (functionality inside / outside the boundaries) of the FPA should be determined.
The scope of the analysis is the set of functional requirements/specifications to be included in the function point analysis.

An application boundary is: the boundary between the (developing or developed) application and its environment (users and/or other applications).
An application in this International Standard refers to:

- an automated information system. This is an application that collects, saves, processes, and presents data by means of a computer.

The following guidelines can help in determining the application boundary:

- The application demarcated by the application boundary should make up an independent whole that can function separately from other applications to a large degree.

- Establish whoever the owner or main user is. If there are several owners or key users, it often means that there are several applications.

- Look at the application through the eyes of the user, that is only at the part of the application the user can actually observe. Use the specifications that describe or define the outside of the application, seen from the user perspective. This is called the application context, and it can be represented in a context diagram, among other ways. Determine what is located inside and outside the application. Only those things that the user requests and that are relevant to him are significant to the function point analysis.

- Think of an application as a group of programs maintained as a whole. The application boundary encloses this group of programs. All functions within this boundary are identified.
Application FPA

The number of function points that measures the amount of functionality that an application is to supply or has already supplied to a user.

Project FPA

The number of function points that measures the amount of functionality of a new application or of changes to an existing application. Changes to an existing application pertain to adding, changing, and/or deleting functions.

Application 1.0
500 FP

Project Y:
Add 100 FP,
Change 20 FP,
Delete 10 FP

Application 1.1
590 FP

Project Y:
130 FP
The software deliverable is sized based upon the functionality delivered.

Five key components are identified based on logical user view:

- External Inputs
- External Outputs
- External Inquiries
- Internal Logical Files
- External Interface Files
Identify the Data Functions

- Data functions represent the functionality provided to the user to meet internal and external data requirements
  - Internal Logical Files (ILFs)
  - External Interface Files (EIFs)
- The term *file* here does not mean file in the traditional data processing sense; in this case, file refers to a logically related group of data and not the physical implementation of those groups of data
- Identify all logically related and user recognizable data or control information within the counting scope.
  - Logically related typically means dependence vs independence of data storage and usage
Measure Transactional Functions

A transactional function is an elementary process that provides functionality to the user to process data and is one of the following:

**External Input (EI)** – an elementary process that processes data or control information sent from outside the boundary

**External Inquiry (EQ)** – is an elementary process that sends data or control information outside the boundary (using data retrieval only)

**External Output (EO)** – is an elementary process that sends data or control information outside the boundary and includes additional processing beyond that of an external inquiry.
### FPA Complexity & Weights

**Inputs - Matrix**

<table>
<thead>
<tr>
<th>FTR’s</th>
<th>DATA ELEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-4</td>
</tr>
<tr>
<td>0-1</td>
<td>Low</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
</tr>
<tr>
<td>3 or more</td>
<td>Ave</td>
</tr>
</tbody>
</table>

**Output & Enquiries - Shared Matrix**

<table>
<thead>
<tr>
<th>FTR’s</th>
<th>DATA ELEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-5</td>
</tr>
<tr>
<td>0-1</td>
<td>Low</td>
</tr>
<tr>
<td>2-3</td>
<td>Low</td>
</tr>
<tr>
<td>&gt; 3</td>
<td>Ave</td>
</tr>
</tbody>
</table>

**Rating**

<table>
<thead>
<tr>
<th>Rating</th>
<th>VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EO</td>
</tr>
<tr>
<td>Low</td>
<td>4</td>
</tr>
<tr>
<td>Average</td>
<td>5</td>
</tr>
<tr>
<td>High</td>
<td>7</td>
</tr>
</tbody>
</table>

Transactions: weights in FP (Function Points)

© ICEAA Physical and Functional sizing
Examples

EXTERNAL INPUTS (EI)
- SCREENS (ADDS, CHANGES, DELETES)
- INPUT FILES/TRANSACTIONS
  - BATCH INTERFACES
  - SCANS
  - XML INPUTS
  - WEB SERVICES, API'S
  - SIGNALS, ALARMS, MESSAGES
  - CONTROL INFORMATION

APPLICATION BOUNDARY
- INTERNAL LOGICAL FILES (ILF)
  - TABLES
  - FILES
  - QUEUES
  - DATA STORAGE

EXTERNAL INTERFACE FILES (EIF)
External Files and Tables Used from Other Applications

EXTERNAL QUERIES (EQ)
- QUERIES, EXTRACTS, DATA VIEWS

EXTERNAL OUTPUTS (EO)
- OUTPUT FILES/TRANSACTIONS
  - BATCH INTERFACES
  - REPORTS
  - XML
  - WEB SERVICES
  - MESSAGES
  - SIGNALS
  - LETTERS
  - NOTICES
  - ALARMS
  - GRAPHICS; PICTURES

Graphic Copyright DCG Software Value 2017 Used with permission
Estimation with FPA

Analogy: painting a wall

• Measure surface: 20 m² (= size)
• Use small brush (1 m² per hour): 20 hours
• Use paint-roller (5 m² per hour): 4 hours
• → Amount of time needed depends on size and productivity

Developing an information system

• Measure the size in function points: 120 FP
• Productivity from experience or from benchmarking database
  • Java: 10 hours/FP: 12 hours
  • .Net: 12 hours/FP: 10 hours
• Amount of time needed depends on size and productivity
Agenda

1. Overview of Lines of Code (LOC) Sizing
2. Overview of IFPUG & NESMA Function Points
3. Overview of COSMIC Function Points
Introduction to COSMIC Function Points
ISO 19761

Material provided by COSMIC
(Common Software Measurement Consortium)
All software does this:

Functional Users types:
1. Humans
2. Hardware devices
3. Other software

The ‘Data Movement of 1 data group’ is the unit of measurement: 1 CFP
(1 CFP = 1 COSMIC Function Point)
COSMIC measurement units - CFP

Largest observed functional processes:
- In avionics >100 CFP
- In banking > 70 CFP

A single CFP exists & is well defined

No max
The task of each functional process is to respond to an event that has happened in the world of the software’s functional users.

The size of a functional process is the count of its data movements (1CFP each)
Example 1: Intruder Alarm System - Requirements

Input devices (functional users)
- Keypad
- Power voltage detector
- Front door sensor
- Movement detectors

Software Boundary
- The embedded alarm software

Output devices (functional users)
- External alarm
- Internal alarm
- 2 x LED’s

Persistent storage
Functional process: Possible intruder detected.
Triggering event: Door opens whilst alarm system is activated.

- * Green LED Switch ‘on’ command (after successful entry of PIN)
- * Red LED Switch ‘off’ command
- Internal siren Stop noise command (after successful entry of PIN)
- External siren Start noise command (after three unsuccessful PIN entries, or if the PIN is not entered in time)
- External siren Stop noise command (after 20 minutes, a legal requirement)

Size = 9 CFP (COSMIC Function Points)
• Business applications
• Real-time software
• Data Warehouse software
• SOA software  (SOA: Service Oriented Architecture)
• Mobile apps
• Agile Development

All COSMIC documents free on the web at:
www.cosmic-sizing.org
COSMIC in Agile

User Story (new &/or re-work) -> Sprint -> Iteration -> Release

COSMIC size usable for:

- Early Total System sizing & effort estimation
- US, Sprint etc. sizing & estimation
- Progress control at any level
Quality of the documentation of a functional process at measurement time

<table>
<thead>
<tr>
<th>Functional Process Quality Level</th>
<th>Quality of the functional process definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completely defined</td>
<td>Functional process and its data movements are completely defined</td>
</tr>
<tr>
<td>Documented</td>
<td>Functional process is documented but not in sufficient detail to identify the data movements</td>
</tr>
<tr>
<td>Identified</td>
<td>Functional process is listed but no details are given of its data movements</td>
</tr>
<tr>
<td>Counted</td>
<td>A count of the functional processes is given, but there are no more details</td>
</tr>
<tr>
<td>Implied (A ‘known unknown’)</td>
<td>The functional process is implied in the actual requirements but is not explicitly mentioned</td>
</tr>
<tr>
<td>Not mentioned (An ‘unknown unknown’)</td>
<td>Existence of the functional processes is completely unknown at present</td>
</tr>
</tbody>
</table>

Input to Sizing or Approximation

Quality of the documentation of a functional process at measurement time

Presented at the ICEAA 2017 Professional Development & Training Workshop - www.iceaonline.com/portland2017
COSMIC Guideline presents 8 approximation techniques (including reported use, strengths & weaknesses):

1. Average functional process approximation
2. Fixed size classification approximation
3. Equal size bands approximation
4. Average use case approximation
5. Early & quick COSMIC approximation
6. Easy function points approximation
7. Approximation from informally written texts
8. Approximation using fuzzy logic - EPCU
COSMIC Function Points allow:

1. Meaningfull benchmarking with an ISO standard
2. Early & Quick sizing
3. Estimation
Thanks for attending this workshop
Questions ?