Headquarters U.S. Air Force

Integrity - Service - Excellence

Software Sizing Lines of Code and Beyond

Air Force Cost Analysis Agency

Corinne Wallshein

June 2009

U.S. AIR FORCE

U.S. AIR FORCE

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?

U.S. AIR FORCE

Sources for software size

- Cost Analysis Requirements Document
- Software schematic diagrams
- Software requirements specifications
- Sub-system requirements specifications
- Analogous (completed) system sizes

U.S. AIR FORCE

Why sizing is important

"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."

Arlene Minkiewicz, PRICE Systems LLC, "The Evolution of Software Size: A Search for Value"

STN 11-3, October 2008: *New Directions in Software Estimation*

U.S. AIR FORCE

Usage: cost driver for estimation

Capers Jones, *Estimating Software Costs*, 2007

PROJECT SIZE	x	PROJECT ATTRIBUTES	=	ESTIMATES -Schedule -Effort -Costs -Deliverables

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

$$PM = A \times (\Sigma\ Size)^{B} \times \Pi(EM)$$

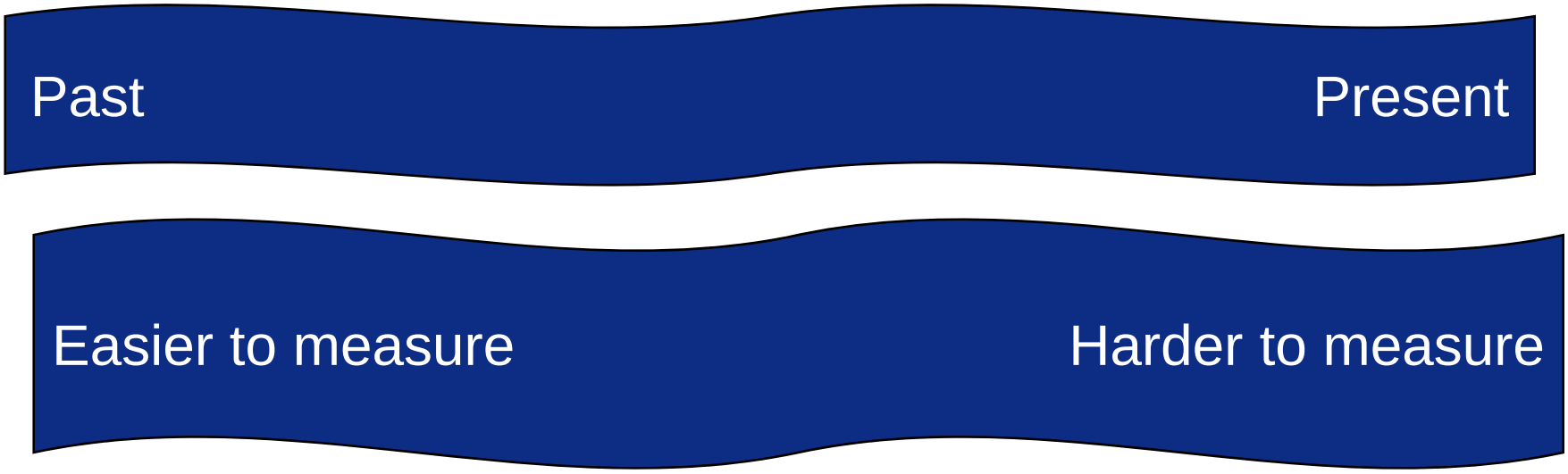
PM = person months.
A = calibration factor.
Size = measure(s) of functional size of a software module that has an additive effect on software development effort.
B = scale factor(s) that has an exponential or nonlinear effect on software development effort.
EM = effort multipliers that influence software development effort.
Each factor in the equation can be represented by a single value or multiple values

http://www.stsc.hill.af.mil/CrossTalk/2005/04/0504Boehm.html **accessed on 6-11-2009**

U.S. AIR FORCE

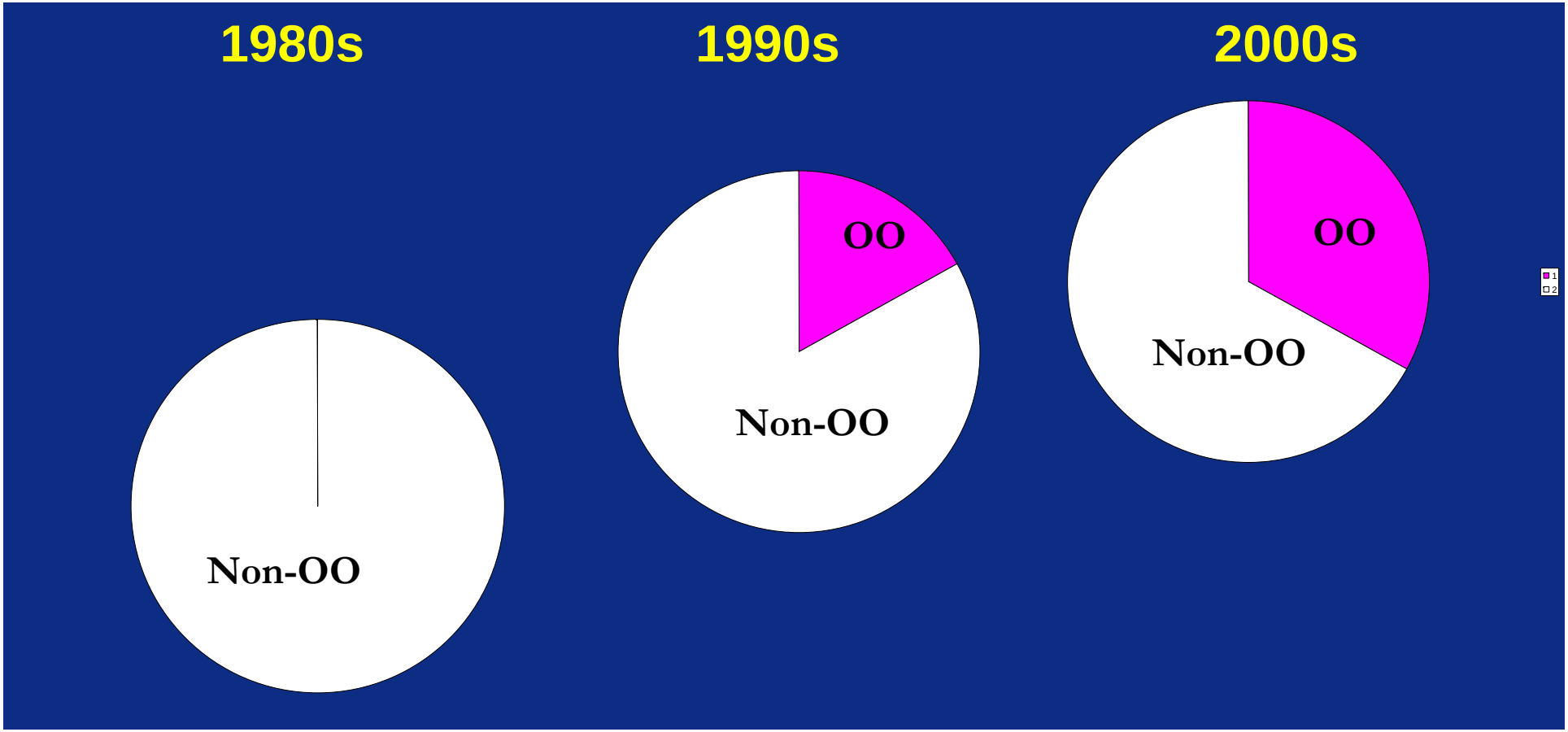
Measuring software size

- **Stand-alone software**
  - **One software language**
  - **One software developer**
  - **One user**
  - **One component**
- **Integrated software**
  - **Multiple languages**
  - **Multiple developers**
  - **Multiple users**
  - **Multiple components**



U.S. AIR FORCE

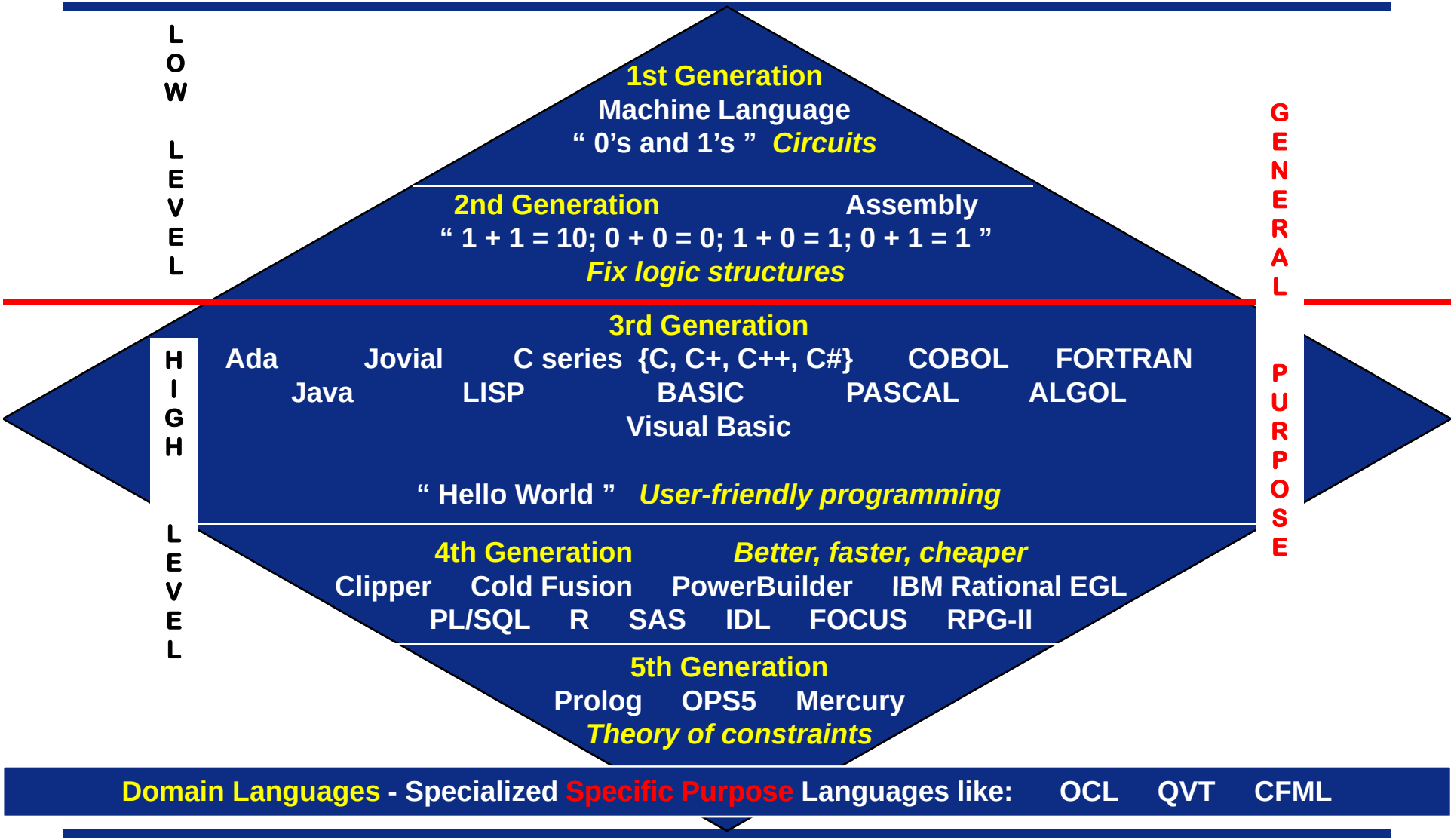
Advent of new development paradigm: Object-Oriented (OO) coding



Source: Air Force Cost Analysis Agency (n = 220 military systems)

U.S. AIR FORCE

Software languages by generation



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

U.S. AIR FORCE

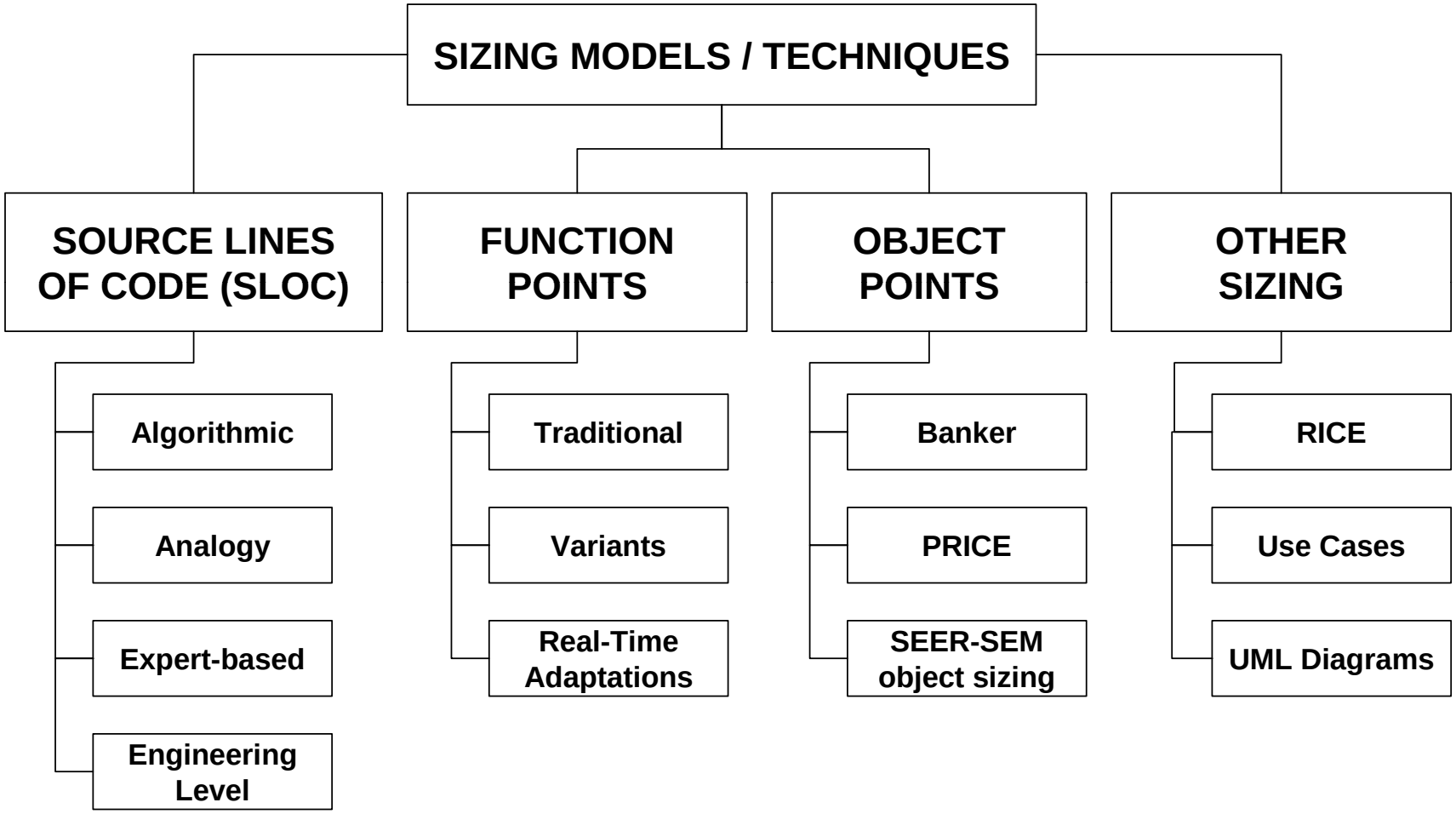
Sizing method characterization

- 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

U.S. AIR FORCE

Sizing units, models & techniques



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

U.S. AIR FORCE

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."

U.S. AIR FORCE

SLOC comparison

LOGICAL STATEMENTS	
PROS	**CONS**
Excludes dead code	Can be difficult to count
Excludes blanks and comments	
Used in a number of estimating tools	
	May be ambiguous for reuse
	Poor choice for full life cycle studies
	Ambiguous for some visual languages
Can be mathematically converted to FPs	
	May be erratic for direct conversion to physical LOC metrics
	Not extensively automated

PHYSICAL LOC	
PROS	**CONS**
Are easy to count	May include dead code
	May include blanks and comments
Used in a number of estimating tools	Ambiguous for mixed language projects
	Ambiguous for reuse
	Poor choice for full life cycle studies
	Does not work for some visual languages
	Erratic for FP conversion
	Erratic for logical statements conversion
Extensive automation	

Capers Jones, *Estimating Software Costs*, published in 2007

U.S. AIR FORCE

Model size inputs

COCOMO II Size Inputs	SEER-SEM Size Inputs	True S Size Inputs
New Software		
New Size	New Size	New Size New Size Non-executable
Adapted Software		
Adapted Size % Design Modified (DM) % Code Modified (CM) % Integration Modified (IM) Assessment and Assimilation (AA) Software Understanding (SU) Programmer Unfamiliarity (UNFM)	Pre-existing Size Redesign Required % Reimplementation Required % Retest Required % Deleted Size	Adapted Size Adapted Size Non-executable % of Design Adapted % of Code Adapted % of Test Adapted Reused Size Reused Size Non-executable Deleted Size Code Removal Complexity
Automatically Translated and Generated Code		
Adapted SLOC Automatic Translation Productivity % of Code Reengineered		Auto Generated Code Size Auto Generated Code Size Non-executable Auto Translated Code Size Auto Translated Size Non-executable

From draft AFCAA Software Cost Estimation Manual, as of June 2009

U.S. AIR FORCE

COCOMO II size quantification

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

From draft AFCAA Software Cost Estimation Manual, as of June 2009

U.S. AIR FORCE

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

U.S. AIR FORCE

Unadjusted FP Conversion Factors to get Adjusted FP

Type of Component	Complexity of Component			
	Low	Average	High	Total
External inputs	___ x 3 =	___ x 4 =	___ x 6 =	
External outputs	___ x 4 =	___ x 5 =	___ x 7 =	
External queries	___ x 3 =	___ x 4 =	___ x 6 =	
Internal logic files	___ x 7 =	___ x 10 =	___ x 15 =	
External logic files	___ x 5 =	___ x 7 =	___ x 10 =	

From Barry Boehm et al., *Software Cost Estimation with COCOMO II*, 2000

U.S. AIR FORCE

SLOC per Unadjusted FP (UFP)

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

PROGRAMMING LANGUAGE	LEVEL	AVERAGE SLOC PER UNADJUSTED FP
5th Generation default	70	5
4th Generation default	16	20
3rd Generation default	4	80
2nd Generation default	3	107
1st Generation default	1	320
Machine language	0.5	640

U.S. AIR FORCE

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…"

U.S. AIR FORCE

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

Conclusion

- 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

U.S. AIR FORCE

Thank you for your attention