

Presented to 18th International Forum on **COCOMO** and Software Cost Modeling
Center for Software Engineering, University of Southern California
October 21-24, 2003, Los Angeles, CA

Evolving Standards in Function Point/ Lines of Code Ratios

Presented by

Koni Thompson Houston

Managing Senior Consultant, The David Consulting Group, Inc.

Member of IFPUG Board of Directors, 2002 – 2005

Member of IFPUG Counting Practices Committee since 1996

The David Consulting Group, Inc

Achieving Software Excellence

Agenda

- Framework for Sizing – A Historical Perspective
- Functional Sizing using IFPUG CPM 4.1.1
- LOC Comparisons in Software Sizing
- Comparison of IFPUG Method vs. COSMIC Method
- Impact of Backfiring to Functional Sizing
- IFPUG Current Activities
- Summary



Framework for Functional Sizing

Historical Perspective

- Major Milestones in functional size measurement:
 - late 1970s - Allan Albrecht of IBM analyzed a number of software development projects in an attempt to find a way of measuring their output.
 - 1979, he summarized his work in a paper on function point analysis.
 - 1984, his concept was extended in *IBM CIS & A Guideline 313, AD/M Productivity Measurement and Estimate Validation*, dated November 1, 1984.
- Function points were originally devised as a method of measuring, and hence estimating, productivity.
- A secondary objective was to devise a measure that was meaningful to non-technical users.
- This was achieved by analyzing the “function value delivered”.

IFPUG Counting Practices Manual

- IFPUG Counting Practices Manual (CPM) evolved from the method first presented by Allan Albrecht in 1979
- The CPM continues to evolve to meet the ever-changing needs of practitioners, while retaining many of the original concepts.
- “Framework for Functional Sizing” (FFS) - Newest document by IFPUG Counting Practices Committee (CPC)
 - explores the different types of requirements
 - provides the basis for identifying functional user requirements and sizing software
 - “provides a means of maintaining that practice while satisfying the requirements of ISO/IEC 14143-1

Evolving ISO Standards

- In 1993, Joint Technical Committee 1 on Information Technology (JTC1) initiated a project to create an ISO standard “which gives a complete and concise description of the software sizing technique, Function Point Analysis”.
- The name of the project was later changed to “functional size measurement”, partly because the measurement of the functional size was better understood than the measurement of “technical size” and “quality size”.
- 1998, the International Standards for information technology were first proposed, ISO and IEC formed a joint technical committee to develop and manage them.
- 1999 - functional size measurement standard was published. This standard defined the Functional Size as “a size of software derived by quantifying the Functional User Requirements”.
 - The focus moved towards measuring what the user required with estimation becoming a secondary objective.
 - This reprioritization grew from a realization that these two objectives were not always complementary.

ISO – 14143-1

- In 1998, ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) jointly published the first part in a series of International Standards for Functional Size Measurement (FSM).
 - Standard - ISO/IEC 14143-1:1998 contains provisions to which FSM Methods must conform.
 - Significant focus - FSM Method must measure “functional” requirements and must not measure “technical” or “quality” requirements.
- In 1999, IFPUG initiated a process to create an International Standard from the IFPUG Counting Practices Manual Release 4.1 Series (CPM) to be published as ISO/IEC 20926:2003.
 - With its publication, this version of the IFPUG CPM conforms to ISO/IEC 14143-1:1998.
 - To ensure that future versions of the CPM can be published as revisions to this International Standard, the CPM has to remain conformant to ISO/IEC 14143-1

ISO/IEC 14143-1:1998

- *ISO/IEC 14143-1:1998* includes the following definitions in an attempt to clarify what functional size is intended to measure:
 - Functional User Requirements –
 - a sub-set of the user requirements.
 - represent the user practices and procedures that the software must perform to fulfill the users' needs.
 - They exclude Quality Requirements and any Technical Requirements
 - Quality Requirements
 - any requirements relating to software quality as defined in ISO 9126:1991
 - Technical Requirements
 - requirements relating to the technology and environment, for the development, maintenance, support and execution of the software
 - NOTE - Examples of Technical Requirements include programming language, testing tools, operating systems, database technology and user interface technologies

Aligning CPM to 14143-1

- Function points were originally designed to measure requirements without considering whether they were functional, technical or quality requirements.
 - Users had a limited understanding of technology; therefore, their specified requirements were generally functional in nature.
 - At that time, the maturity of the technology meant that there were very few options to delivering the required solution, and these were usually represented as “design alternatives”.
 - Specified requirements, therefore, rarely included what are now referred to as “technical and quality requirements”.

Requirements Categorization

- Alternative categorization of requirements from 2 additional sources
 - Institute of Electrical and Electronics Engineers Inc. (IEEE)
 - Estimation model from Practical Software Measurement (PSM) that merges some of the international standards with industry practice.

IEEE Definition of Requirements

IEEE Definition of Requirement	Requirement Type	
	IEEE	ISO
Specifies or constrains the design of a system or system component	design	technical
Specifies a function that a system or system component must be able to perform	functional	functional
Specifies or constrains the coding or construction of a system or system component	implementation	technical
Specifies an external item with which a system or system component must interact	interface	functional
Sets forth constraints on formats, timing or other factors caused by such interaction	interface	technical
Imposes conditions on a functional requirement; e.g., specifies the speed, accuracy, or memory usage with which a given function must be performed	performance	technical
Specifies a physical characteristic that a system or system component must possess; e.g., material, shape, size or weight	physical	technical

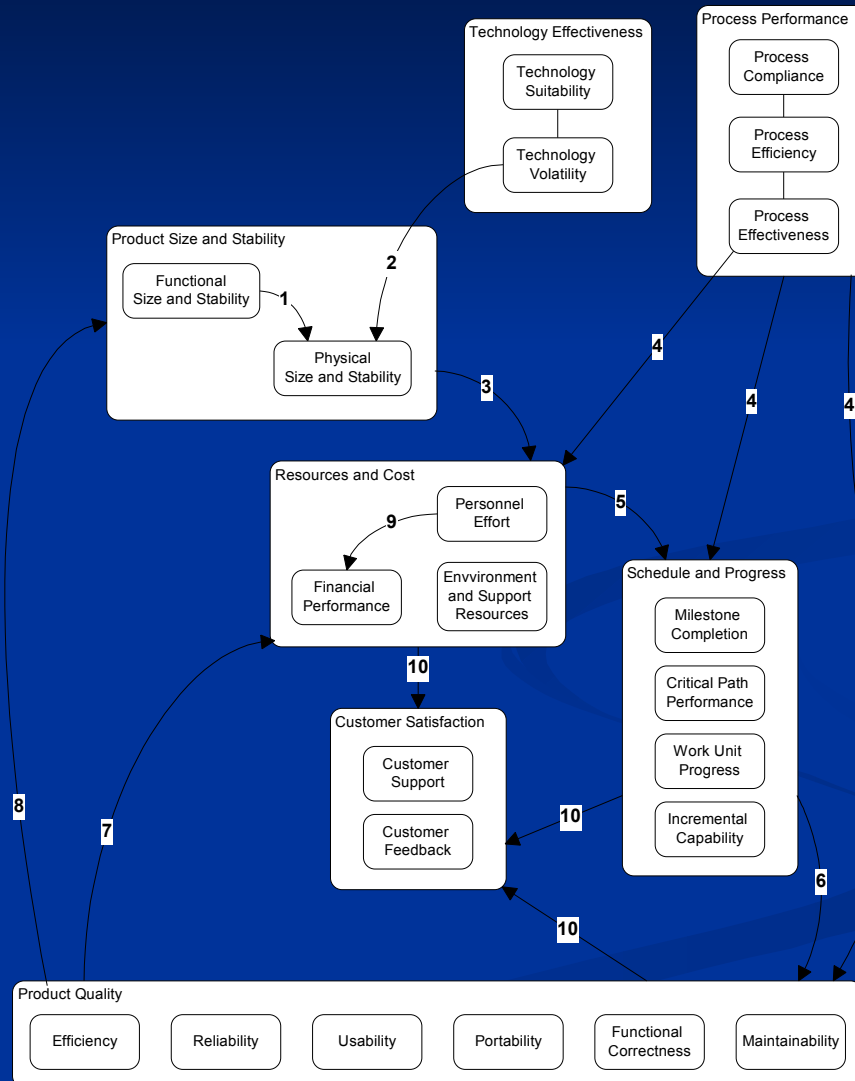
IEEE Standard

- *IEEE Std 610.12-1990* also includes two definitions of quality:
 - The degree to which a system, component, or process meets specified requirements.
 - The degree to which a system, component, or process meets customer or user needs or expectations.

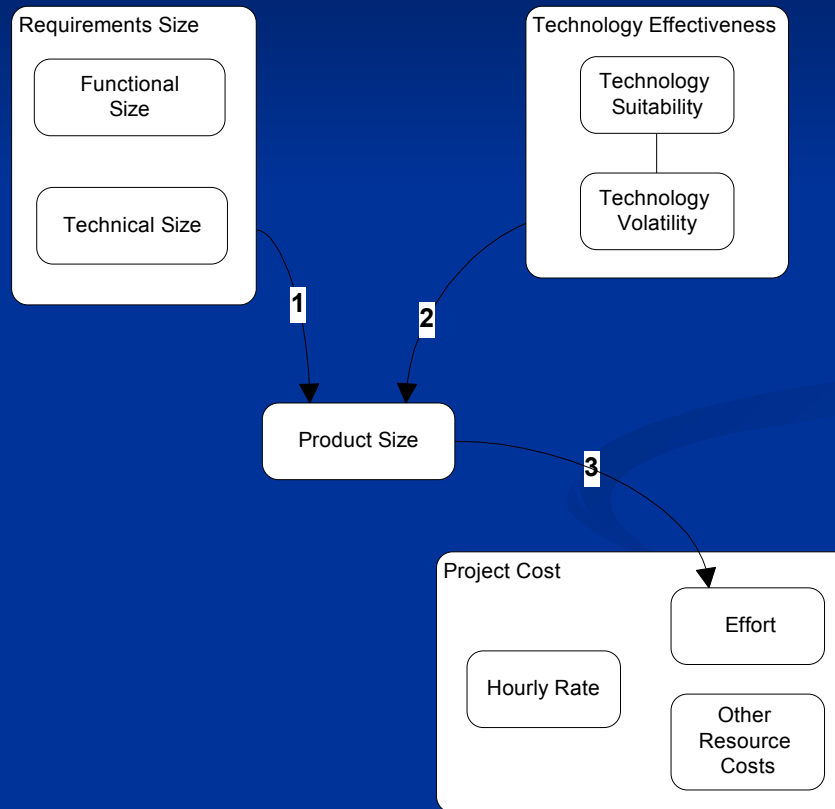
Practical Software Measurement

- PSM - Supported by the US Department of Defense, PSM has been gaining increasing acceptance in industry.
- The project was initiated in the mid 1990s, and was most recently incarnated in book published in 2002.
- PSM provides a foundation for objective project management in the software and systems arena.
- As such, its scope is far wider than functional size measurement but also allows us to view the functional size in a wider context.
-

PSM Detailed Integrated Analysis Model



Conceptual Sizing Model



Analyzing Requirements

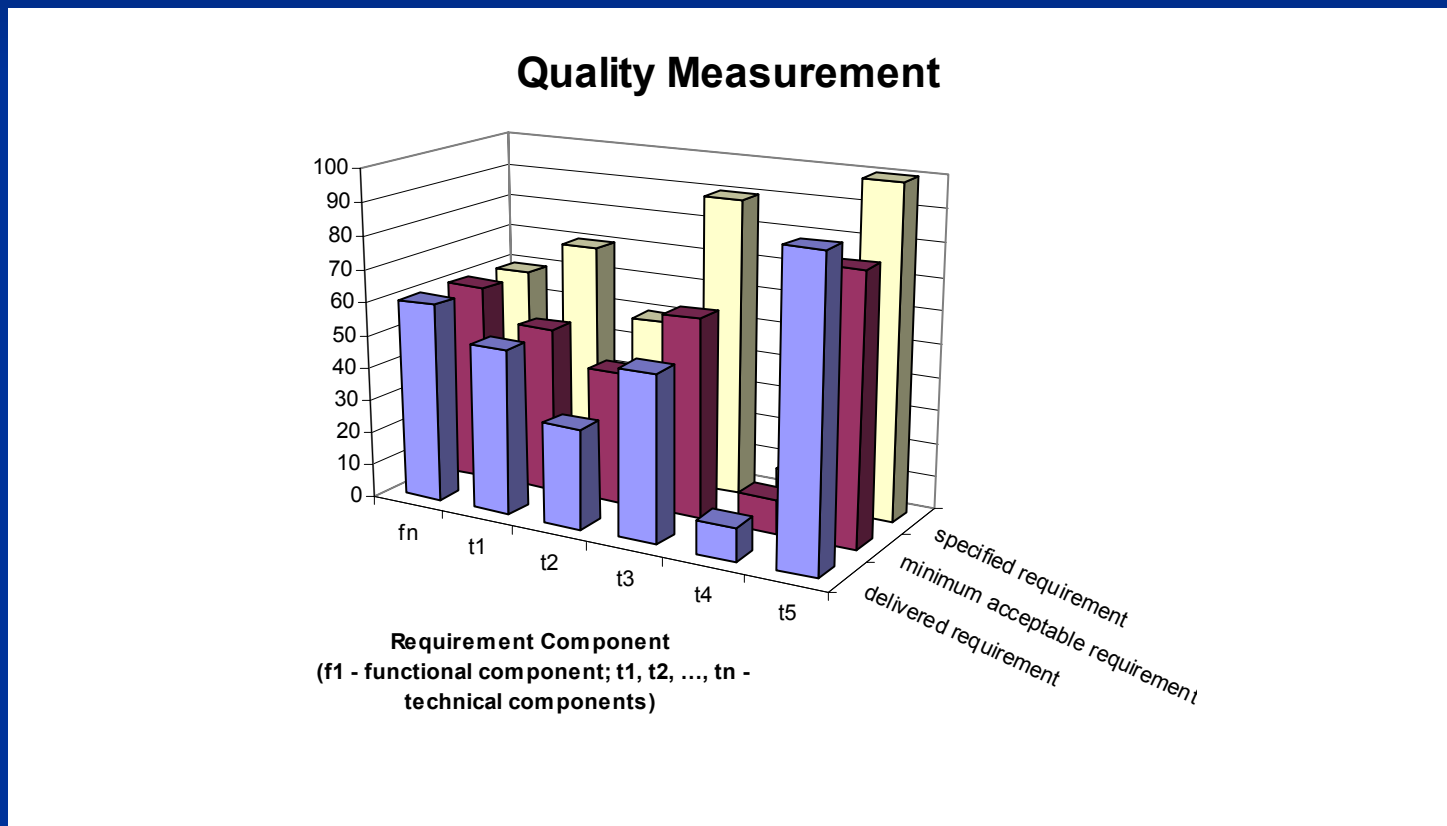
- 5Ws and an H technique – provides refinement of functional/ technical requirements
- Functional – what must be done, by whom and when (only if timing affects the value of the result)
- Technical – how it must be done, where and when (if timing does not affect the value of the result)

Analyzing Requirements

Question	Manifestation in User Requirements	Functional	Technical
WHO?	The organizational unit, job type or client type to which the requirement relates	X	
WHAT?	The task that needs to be performed; a functional requirement	X	
WHERE?	The machine or machine type (client, server, etc.) on which the processing is required to occur		X
WHEN?	There are several possibilities; e.g.,		
	<ul style="list-style-type: none"> Performance – the task must be completed within one hour of initiation 		X
	<ul style="list-style-type: none"> Order – the task must be initiated after completion of the weekly reconciliation and must be completed before initiating the weekly reports 	X	
	<ul style="list-style-type: none"> Period – the processing logic depends on the period of time to which the transaction applies 	X	
WHY?	The rationale behind the requirement; asking "why" a requirement exists, results in answers to the "who", "what", "where", "when" and "how"	Not directly applicable	Not directly applicable
HOW?	The manner in which the task is initiated, performed or in which it delivers its results; asking "how", determines the methods used to deliver the functionality; the "how" can affect the data entry, processing and data delivery mechanisms, but does not affect the value of the result (i.e., the "what")		X

Model of Software Quality

Figure 4 is a model of a single user requirement. The requirement has a functional component (the functional requirement) and a number of technical components (the technical requirements). These are shown in the third row, labeled “specified requirement”.



Requirements Size vs. Product Size

- The *requirements size* (a combination of the functional and technical sizes) can be used to quantify the specified requirements.
 - The technology effectiveness contributes those technical attributes that have not been specified by the user.
- The *product size* represents the combination of the specified requirements and the technology effectiveness.
 - If the functional and technical requirements (and hence technology effectiveness) can be measured on the same scale, it will be possible to produce a single number to represent the *product size*.
 - Since the technology effectiveness can embody such tools as reuse and package customization, it should not be assumed that the resulting *product size* will always be greater than the *requirements size*.
- In summary, the *product size* quantifies what needs to be built and/or customized.

FSM Summary

- Functional size is not the answer to all existing estimation problems.
- There is also a need to measure technical size. This will allow calculation of *requirements size* and a *product size*.
- IFPUG recognizes this need and is working on several solutions.
- Currently, this takes the form of applying the General Systems Characteristics (GSCs) to the functional size.

Current Usage of FSM

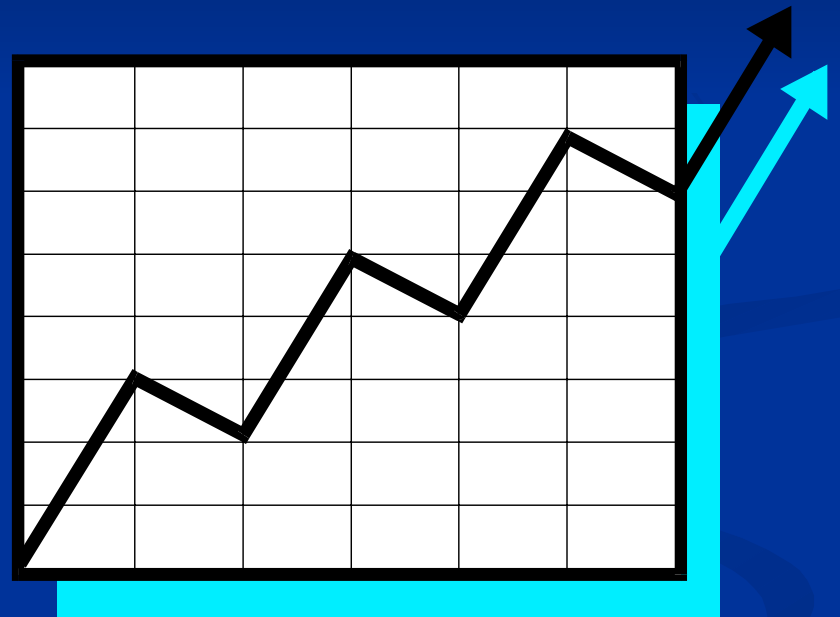
- Software Development Contracts
- Outsourcing of Maintenance and Enhancement
- Benchmarking

Functional Sizing using IFPUG CPM 4.1.1



Software Development Challenges

- Requirements
- Estimation
- Change Management



Development Challenges

Requirements

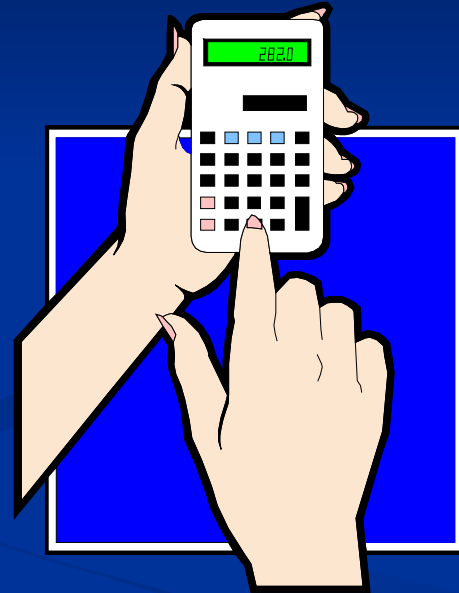
- Complete
- Business Terms
- Mutual Understanding
- Document Assumptions
- Size



Development Challenges

Estimation

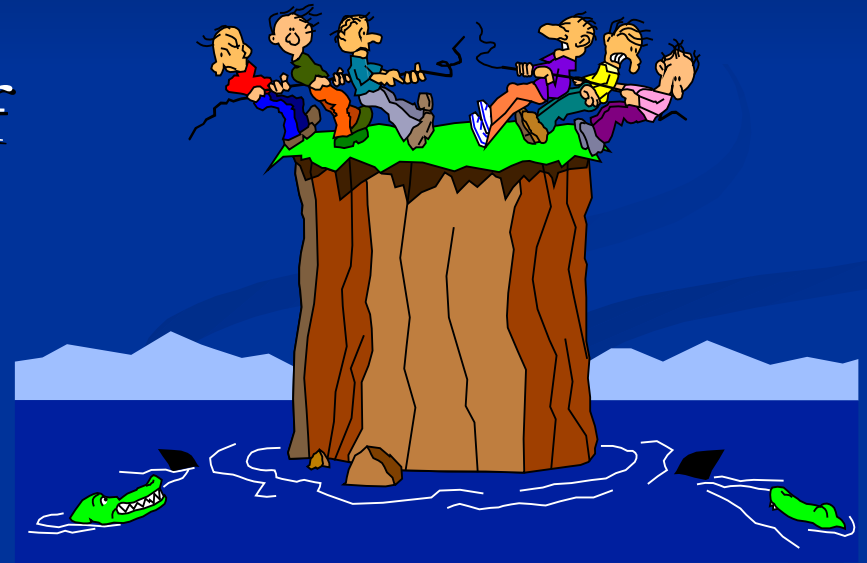
- Multiple Models
- Weighted Inputs:
 - Language
 - Skills
 - Methodology
 - Risk Factors
 - Size
- Historical Base



Development Challenges

Change Management

- Change Inevitable
- Trade-offs
- Customer Definition of Quality
- Size

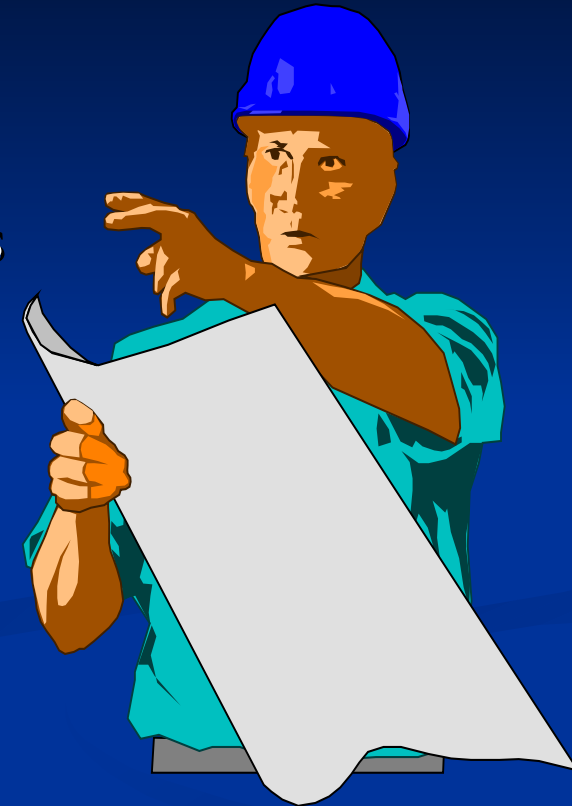


Function Point Definition

- “A software size measure.
- They measure the amount of information processing functionality contained within a software product.
- They are derived early in the software life cycle from requirements or design specifications, and are applied across diverse application domains and technology platforms.”
 - *Practical Software Measurement, NUWC 1997, P.381*

Function Point (FP) Overview

- Size of Logical User Requirements
 - Floor Plan Square Feet
- Developers take Requirements through Installation via 100's of tasks
 - Blueprints to Final Construction



FP Objectives

Source IFPUG CPM 4.1

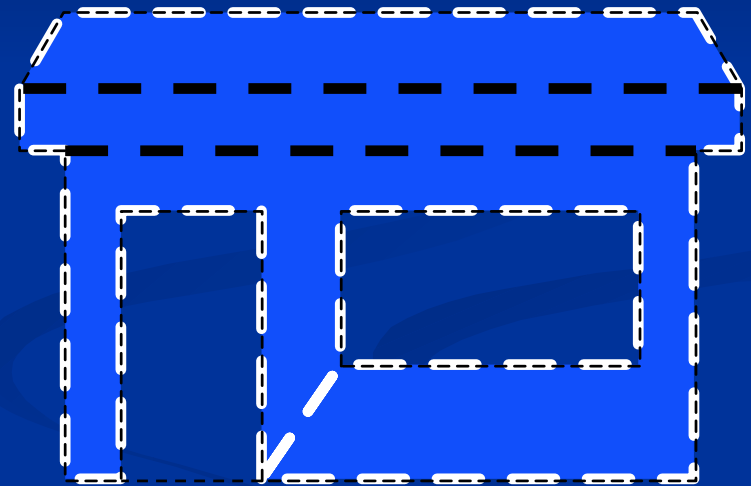
- Measure functionality the user requests and receives
- Measure independent of implementation technology
- Provide a normalization factor for software measurement



Types of Counts

Development (Project) FP

- Construction Project size
- User functionality in the project
- Less conversion FP, equals Application Base



Types of Counts

Application (Base) FP

- Static (point in time) size of an Installed Application
- Current functions of the application
- Re-analyze after enhancements



Types of Counts

Enhancement (Project) FP

- Size of Renovation / Modification Project
- Enhancement of existing software
- Sum of New FP + Changed FP + Removed FP + Conversion FP



FP Overview

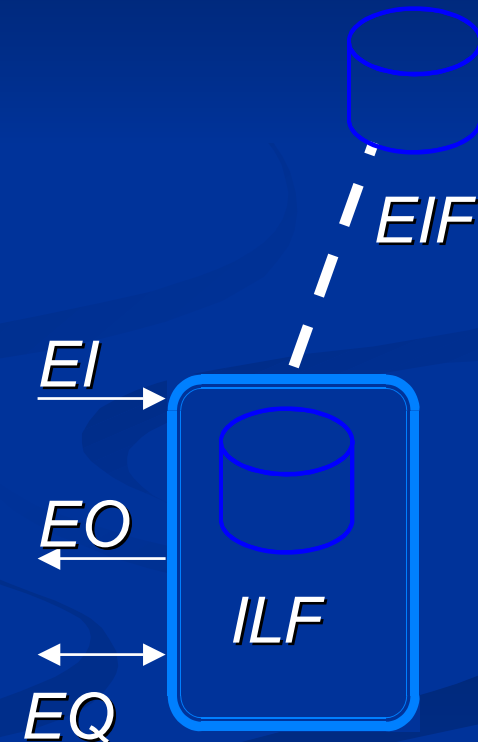
Source IFPUG CPM 4.1

Data Functions:

- Internal Logical File ILF
- External Interface File EIF

Transactional Functions:

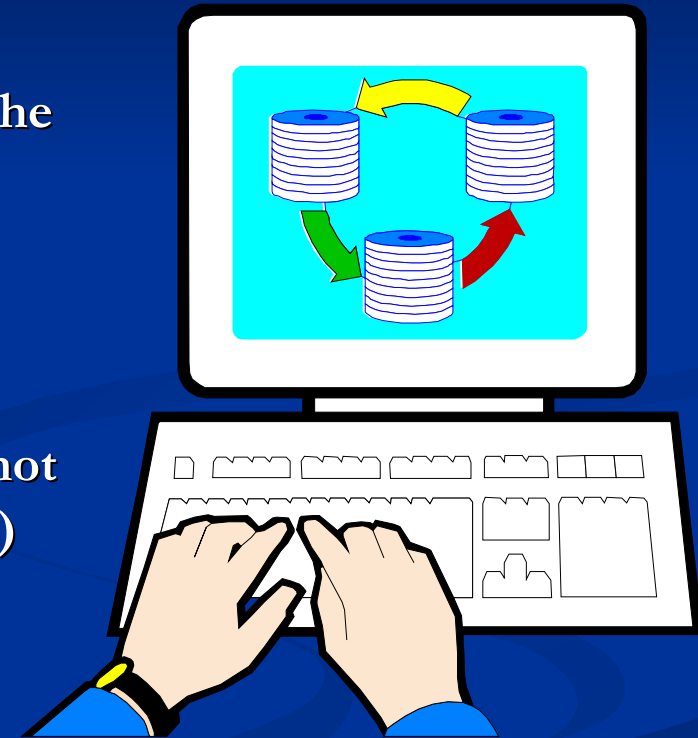
- External Input EI
- External Output EO
- External Query EQ



FP Overview

Source IFPUG CPM 4.1

- Internal Logical File (ILF)
Logical group of data maintained by the application (e.g., Employee file)
- External Interface File (EIF)
Logical group of data referenced but not maintained (e.g., Global state table)



FP Overview

Source IFPUG CPM 4.1

- External Input (EI)

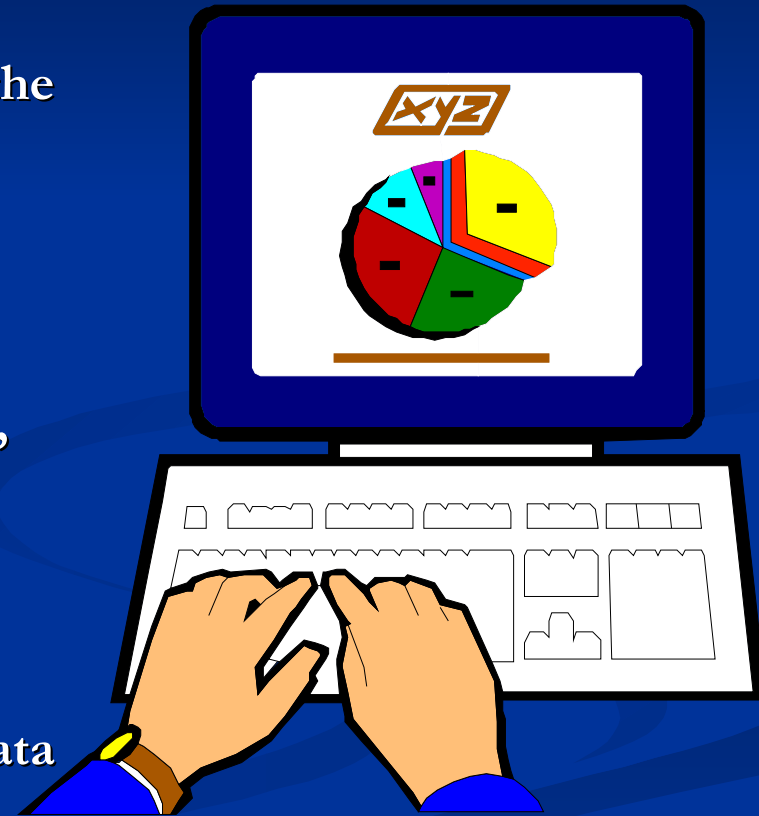
Maintains ILF or passes control data into the application.

- External Output (EO)

Presents info to user; includes calculations, derivations or updates ILF.

- External Query (EQ)

Presents information to user thru simple data Retrieval.



FP Overview

Source IFPUG CPM 4.1

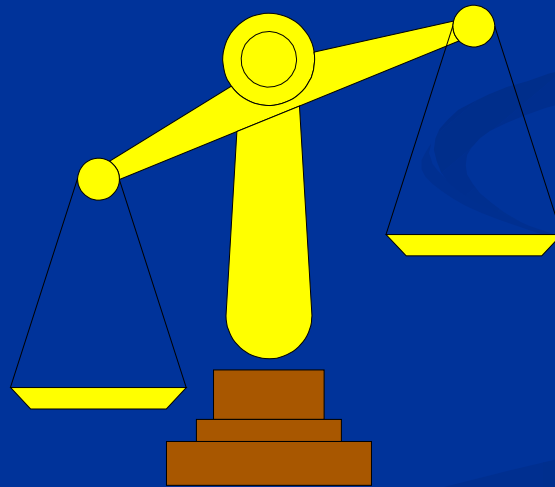
■ Function Weights

Function Type	Low	Average	High
EI	x 3	x 4	x 6
EO	x 4	x 5	x 7
EQ	x 3	x 4	x 6
ILF	x 7	x 10	x 15
EIF	x 5	x 7	x 10

FP Overview

Source IFPUG CPM 4.1

- Adjust FP based on 14 User Business Constraints (Independent of Technology) by up to + / - 35%



Logical versus Physical (Examples)

LOGICAL:

- 1. The application must store Employee information.*
- 2. Users need the ability to add, change and delete employees.*

PHYSICAL:

- 1. Implemented using monthly Employee DB2 tables.*
- 2. Payroll system sends in a TX file with new, updated & terminated employees to our application.*

FP Counting Aids

- Requirement Document(s)
- Data Models
- Data Flow Diagram (DFD)
- Entity Relationship Diagram (ERD)
- Flow Charts
- Interface Descriptions
- Live Application
- Report Layouts
- Screen Layouts
- On-line Tutorial
- System expert (for the system at hand)
- File Layouts
- User Manual



FP Based Measurement

- Requirements
 - Quantify Size based on User Functions
- Estimating
 - Objective Size of Work Product
- Change Management
 - Size / Impact of Changes
- Combination Metrics

Productivity Metrics Samples

- Enhancement or delivery rate

FP's project / work effort

- Application support rate

FP's base / work effort

- Application maint load per person

FP's base / # of maintenance people (FTE's)

Quality Metrics Samples

- Defect ratio

of defects / Project FP's. Use with Mean Time to Repair ratio (Elapsed time / # of problems)

- Repair cost ratio

*(Total hours to repair defects * hourly costs) / FP's*

- Application reliability

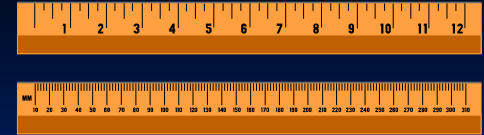
1 - (Application failures / Application FP's)

- Stability ratio

1 - (# of changes / FP's)

Summary

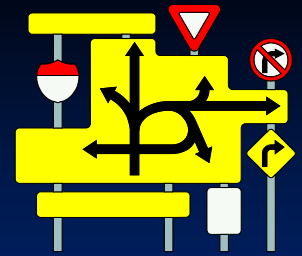
What FP are



- Additional Fact-Based Measure for Project Management
- Functional Size of Software Based on Evaluation of User Requirements
- Independent of Tools, Technology, and Other Project Attributes
- Used as a Size Normalizer across Technologies, Platforms, etc.

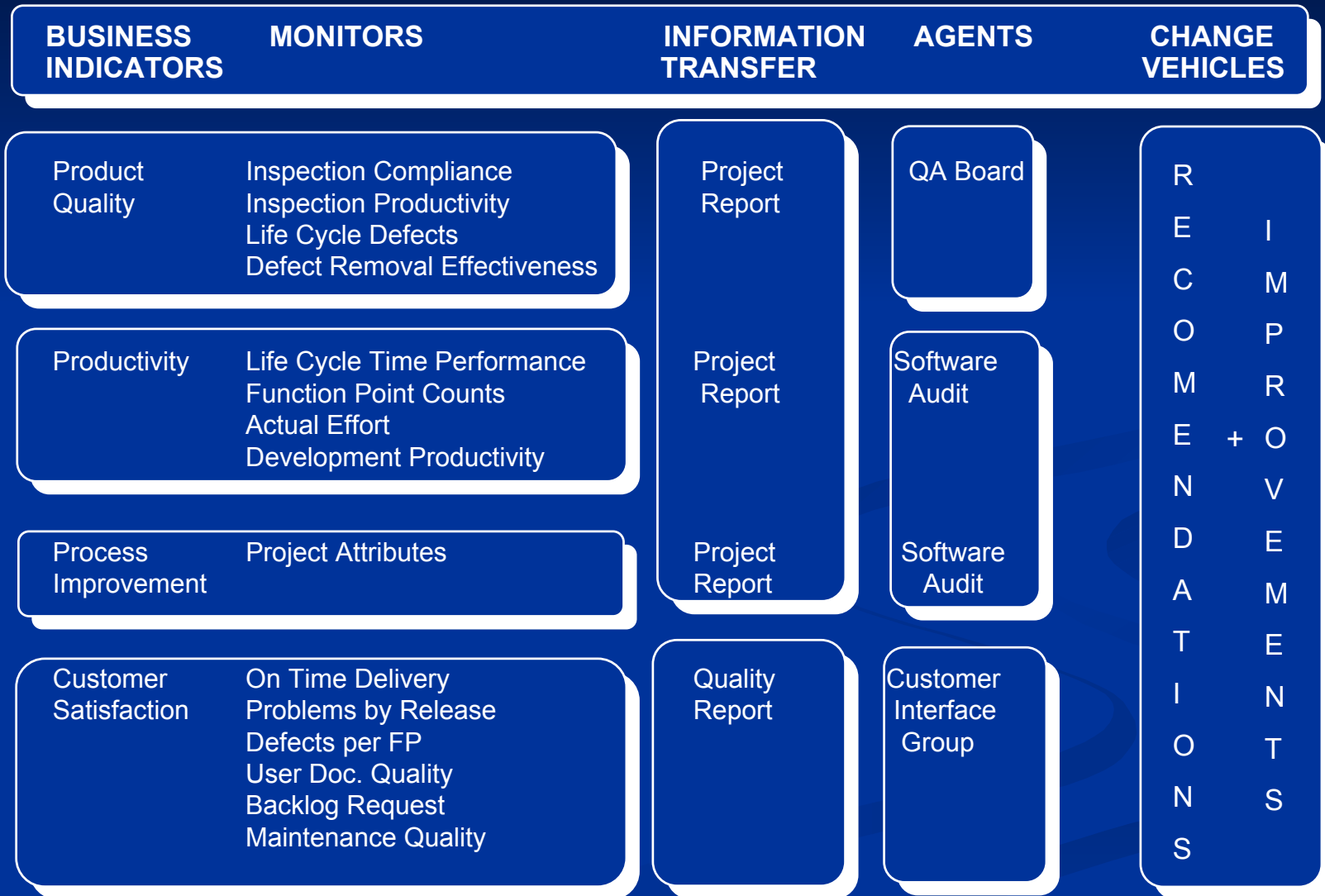
Summary

What FP are Not

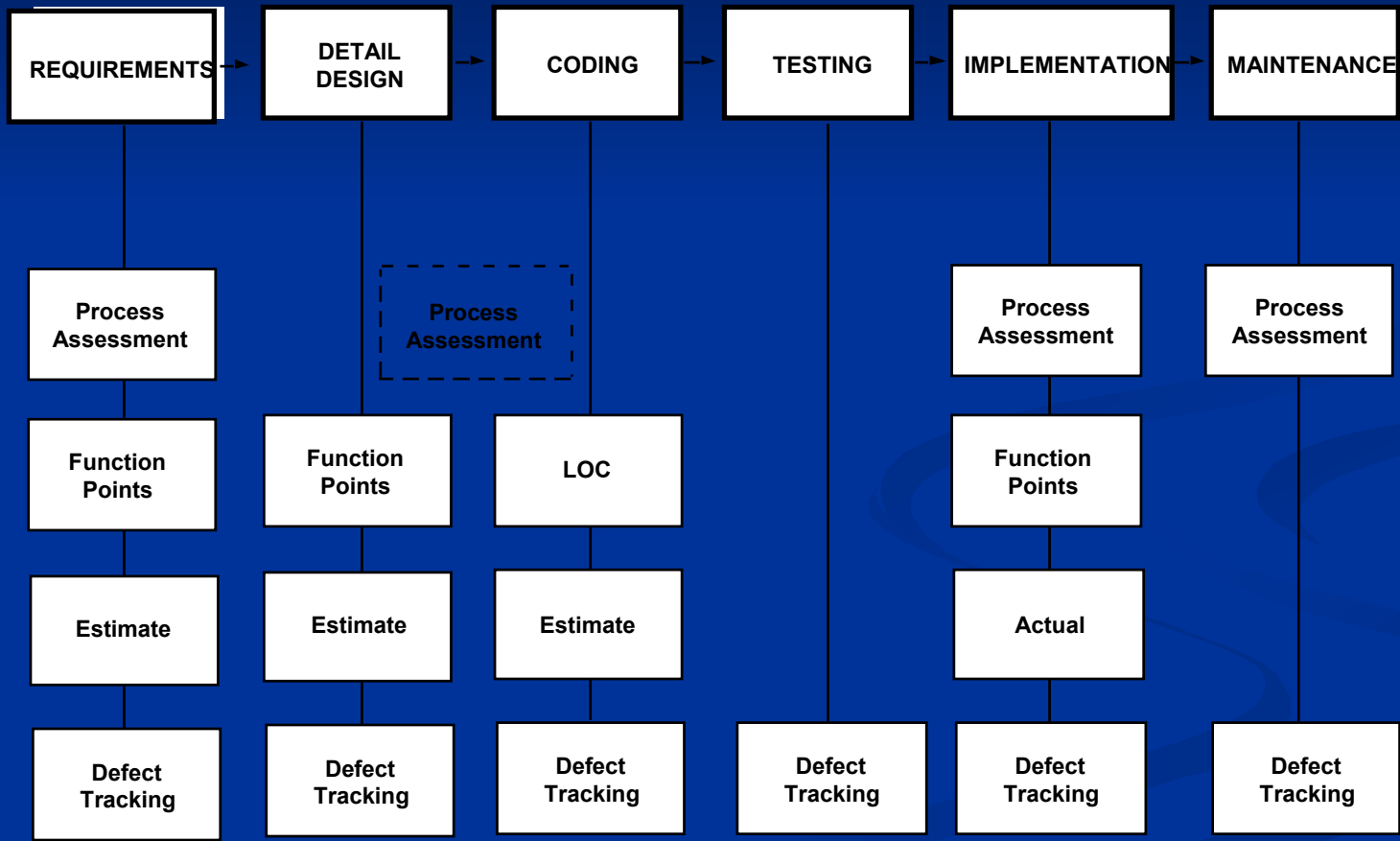


- Equal to Effort, Productivity, or Internal Application Complexity
- The “Silver Bullet” or Swiss Army Knife of Metrics
- A Process Improvement “Quick Fix” or Road Map
- A Substitute for Project Attributes

STRATEGIC IMPROVEMENT PROCESS



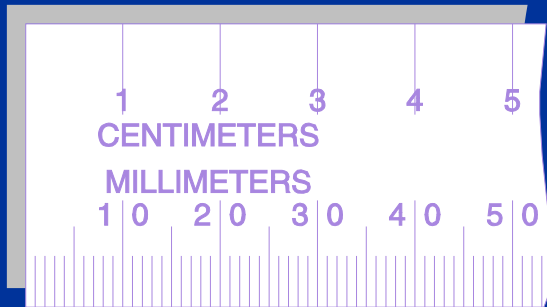
MEASUREMENT ACTIVITIES IN THE LIFE CYCLE



LOC comparisons in Software Sizing

Early Software Size Options

- Function Points
(Logical)
- SLOC (Physical)
- McCabe or Halstead
Metric (Code
Complexity or
Density)



Sizing Options

- **Halstead/McCabe Complexity**
 - Measures code density and complexity
 - Used for testing and code maintenance
 - Counted at end of Coding phase
- **SLOC**
 - Counted at end of Coding phase
 - Inconsistent, limited rules, not valid for comparisons across projects
- **FP**
 - Counted at Requirements, Design, Coding, Implementation
 - Consistent, Rules based, ISO standards
 - Basis for industry comparison

Investigating SLOC and FPs

- SLOC don't measure user functionality
- No agreement on the definition of a SLOC
- SLOC are not comparable across different programming languages
 - Example: Cobol 108 SLOC = C++ 53 SLOC
- SLOC are more of a measure of coding phase productivity and quality not the entire SDLC
- SLOCS are not available early enough in the requirements phase for estimating purposes

Investigating SLOC and FPs

- FPs measure what the user paid for
- IFPUG has defined what an FP is and how it is measured
- FPs are independent of technology used
- FPs are a measure of the complete development lifecycle
- FPs can be estimated very early in development lifecycle

SIZING SOFTWARE USING “LINES OF CODE”

Varying Definitions of “Lines of Code”

- ↓ Executable lines
- ↓ Executable lines + data definitions
- ↓ Executable lines + data definitions + comments + JCL
- ↓ Delivered lines + support software + test cases
- ↓ New lines
- ↓ New lines + changed lines
- ↓ New lines + changed lines + residual
- ↓ “Line” meaning a physical line
- ↓ “Line” meaning statements between delimiters

EXAMPLES OF LINE COUNTING VARIANCE:

EXAMPLE 1

If the counting rules are:

- **Only executable lines**
- **Delivered lines only**
- **New lines only**
- **Macros counted once**
- **“Line” meaning delimiters**

Then a program might equal 10,000 lines of source code

EXAMPLES OF LINE COUNTING VARIANCE

EXAMPLE 2

But if the counting rules are:

- **All lines (executable, data comments, JCL)**
- **Delivered + support + test cases + scaffold code**
- **New + changed + residual lines**
- **Macros counted for each usage**
- **“Line” meaning physical lines**

Then the same program might equal 100,000 lines of source code

Evolving Software Size Options

MEASURE	ADVANTAGES	DISADVANTAGES
Function Points (FP)	Technology independent Logical user view International acceptance (IFPUG)	Requires training
Mark II Function Points	Technology independent Logical user view	Acceptance isolated-UK Proprietary (until recently) Dependent on implementation
Lines of Code (KLOC)	Can be done from physical implementation	Technology dependent Inconsistent rules Counter indicator of productivity Unavailable until coded
Feature Points (Capers Jones - early 1990's)	Enhanced FP for algorithmically intensive	Experimental
FP Backfired from LOC	Derive FP where limited documentation	Less accurate than FP Same limitations as LOC

CHARACTERISTICS OF AN EFFECTIVE SIZING METRIC

- Meaningful to developer and user
- Defined (industry recognized)
- Consistent (methodology)
- Easy to learn and apply
- Accurate, statistically based
- Available when needed (early)

CANDIDATES FOR SIZING METRICS

Metrics

- Lines of code
- **Function Points**
- Mark II
- COSMIC
- Object Points
- Artifacts

Criteria

	Meaningful	Defined	Consistent	Easy to Learn	Accurate	Available
			✓			
✓	✓	✓	✓	✓	✓	✓
✓	✓	✓		✓		
✓	✓	✓				✓
✓	✓	✓				✓
✓			✓			✓

MEASURING WITH LINES OF CODE

The **Pros** and Cons of Using SLOC

- Can be counted consistently within organization
- Available late in the life cycle
- Lines of code not always available
- No industry standard for counting lines of code
- Limited ability to compare among various language groups
- Internal use only, not meaningful to customer

SIZING FROM ARTIFACTS

Determine size (and estimate) based on an evaluation of existing or planned artifacts

Artifacts may include:

- ↗ Document size
- ↗ Number of screens, reports
- ↗ Programs/modules
- ↗ WBS

FUNCTION POINTS

Function Point Analysis is a standardized method for measuring the functionality delivered to an end user.

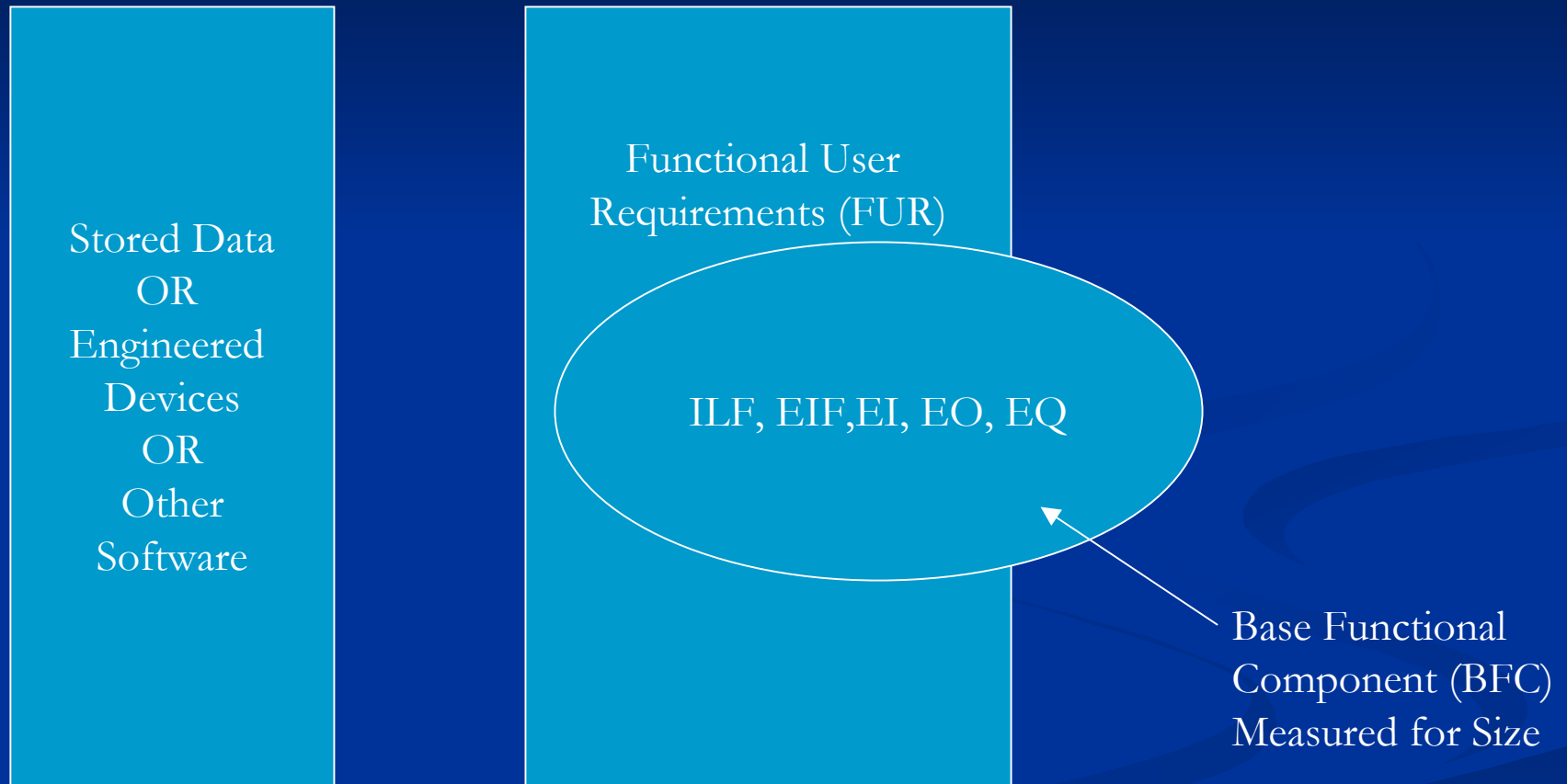
- Consistent method
- Easy to learn
- Available early in the lifecycle
- Acceptable level of accuracy
- Meaningful internally and externally

Comparison of IFPUG Method vs. COSMIC Method

Comparison of IFPUG Method vs. COSMIC Method

- ISO/IEC/JTC1/SC7 Standard 14143-1 (1998) definitions:
 - Functional Size – a size of the software derived by quantifying the functional user requirements
 - Functional Sizing Measurement (FSM) the process of measuring functional size
 - FSM Method – A specific implementation of FSM defined by a set of rules, which conforms to the mandatory features of ISO/IEC 14143 (e.g., IFPUG 4.1 Unadjusted; COSMIC-FFP)

Base Functional Component



BFC Types

■ IFPUG

- Input, Output, Enquiry
- Internal Logical File
- External Interface File
- 5 BFCs

■ COSMIC

- Entry
- Read
- Write
- Exit
- 4 BFCs

Similarities

- IFPUG and COSMIC methods
 - Both recognize
 - Elementary process as functional unit
 - Data moving in/out of process as contributing factor to size
 - Data accesses to persistent data as contributing factor to size

Differences

- IFPUG
- Measures Process and Data
- 3 Process BFC types
 - Min size 3 fps
 - Max size 7 fps
 - Aggregates DETs crossing the boundary
 - Counts individual DETs
 - Counts access to persistent data once per process (read/write count as 1)
- COSMIC
 - Only measures process
 - 4 sub process BFC types
 - Min size – 2 CFSU
 - Max size – infinite
 - Aggregates DET groups
 - Counts DET groups
 - Counts access to persistent data and Read (one) and/or Write (counts one/twice)

Advantages of COSMIC

- Fast consistent counting as all processes counted using same rules
- Explicit rules for sizing software in layered architecture

Disadvantages of COSMIC

- No available 'quick' short cut of counting
- Needs fairly detailed specifications
- Inconsistency in way people identify layers
- Provide very few extra guidelines for counting different environments or examples of counts
- Very limited supporting resources, training, tools and industry data
- Only 30 FFP projects in ISBGS vs. over 1500 with IFPUG

Advantages of IFPUG

- Easy counting of ranges of DETs and RET/FTRs
- Quick rough counting – default to BFCs to industry averages
- Long term industry evidence to good correlation to effort
- Readily available skills, training, tools, resources and historical data
- International certification
- “the Microsoft of the Functional Sizing World”

Disadvantages to IFPUG Method

- BFC often inconsistently appraised
 - Where processes do not have a clear primary intent
 - Processes are part of multi-layered architecture

Impact of Backfiring to Functional Sizing

Summary of Software Size Options

MEASURE	ADVANTAGES	DISADVANTAGES
Function Points (FP)	<ul style="list-style-type: none">• Technology independent• Logical user view (supports ISO Functional Sizing Method)• International acceptance (IFPUG)• 90% of all documented FS based on IFPUG FP Counting Rules	<ul style="list-style-type: none">• Requires training• Not intuitive
FP Backfired from LOC	<ul style="list-style-type: none">• Derive FP where limited documentation	<ul style="list-style-type: none">• Less accurate than FP• Same limitations as LOC

SIZING STUDY

Software Sizing: Alternative Methods

BASELINE COUNTS

Level 1 - Detailed

All functions are identified, sized and weighted (IFPUG)

Resources required: Client / Consultant

Source: Knowledgeable on application, system expert

Documentation Required:

User guide (on-line)

System flow (batch)

Data model/schema

Effort: 1 – 3 days (average)

Deliverable: Function Point Report

Level 2 - Limited

All functions are identified, sized and average weightings are applied

Resources required: Client / Consultant

Source: Knowledgeable on application, system expert

Documentation Required (limited):

User guide (on-line)

System flow (batch)

Data model/schema

Effort: 1 – 3 days (Limited counts take up to 35% less time to complete than detailed counts)

Deliverable: Function Point Report (approximated)

Note: Effort reflects consulting time only. In each scenario there will be client preparation time and participation time required

APPROXIMATED BASELINE COUNTS

Level 1 - Robust

Identifies all physical components, translated into logical entities

Resources required: Client / Consultant

Source: Knowledge of functional components,
knowledge of I/P/O

Documentation Required: existing documentation

Effort : One half day

Deliverable: Approximated Baseline Function Point Size

Level 2 - Ratio

Uses industry accepted ratio based estimating; statistically based

One file rule (31 FP for every ILF); ILF equates to 22% of total (ISBSG)

Resources required: Client/Consultant

Source: Identify the logical data for the application

Documentation Required: logical data model

Effort : 2 – 4 hours

Deliverable: Approximated Baseline Function Point Size

Note: Effort reflects consulting time only. In each scenario there will be client preparation time and participation time required

APPROXIMATED BASELINE COUNTS

Level 3 - Expert

Package systems or common applications

Resources required: Client/Consultant

Source: Functionality installed is identified

Documentation Required: User Guide

Effort : 2 – 4 hours

Deliverable: Approximated Baseline Function Point Size

Level 4 – Delphi

Application is positioned relative to “like” applications

Resources required: Client / Consultant

Source: Size of comparative applications

Documentation Required:

Effort : 1 hour

Deliverable: Approximated Baseline Function Point Size

Note: Effort reflects consulting time only. In each scenario there will be Client preparation time and participation time required

BACKFIRED BASELINE COUNTS

Level 1 - Calibration

Backfire values are calibrated to client experience

Resources required: Client/Consultant

Source: Lines of code, system expert,

Documentation Required: System

Effort : Varies

Deliverable: Calibrated Backfired Baseline Size

Level 2 - Calculation

Industry backfire values are applied

Resources required: Client / Consultant

Source: lines of code

Documentation Required: Counting guidelines

Effort : 1 hour

Deliverable: Backfired Baseline Size

Note: Effort reflects consulting time only. In each scenario there will be client preparation time and participation time required. Effort to obtain appropriate line of code counts may be significant

COST COMPARISONS FOR ESTABLISHING AN APPLICATION PORTFOLIO

Count Type	Accuracy	Cost
Baseline	+/- 5%	1 - 3 days
Baseline - Limited	+/- 25%	1 - 3 days
Robust	+/- 35%	1/2 day
Ratio	+/- 50%	< 1/2 day
Expert	+/- 50%	< 1/2 day
Delphi	+/- 100%	< 1/4 day
Backfire	+/- 100%-400%*	varies

* Variation occurs based upon language levels

Note: The cost is based on an average size application (250 - 1200 FP) and will vary for applications outside of that size range.

Backfiring FP

- Chart showing conversion of FP to LOC published in 1990
- Based on CPM, Release 3.0 (1990)
- Not updated since CPM has changed

Accuracy Comparisons - Counting vs. Backfiring

COUNT TYPE	ACCURACY
IFPUG COUNTING	+/- 5%
FP BACKFIRING TO LOC	+/- 100% to +/- 400%

Based on DCG studies from 1994 to 2002

DCG's LOC Conversion Ratio

- If you are planning to use the backfire method, you can increase your accuracy (relatively speaking) by using current backfire values (based upon IFPUG Counting Practices Manual 4.x)
- The conversion rates which follow reflect changes and clarifications to counting rules that have occurred since CPM Release 3.0 (1990)

DCG's LOC Conversion Ratio

(for Backfired FP Counts)

- If you reference other backfire numbers, ensure that they have been updated from those numbers published prior to 4.x
- DCG does not consider backfiring as a recommended approach for sizing software project deliverables
- We would use backfiring only when sizing an organization's installed applications
- DCG validates our LOC conversion ratios by sampling:
 - performing actual Function Point counts and comparing them to Lines of Code
 - validating applications within the installed base at that organization

DCG's LOC Conversion Ratio *

Based on Release 4.1 of the IFPUG Counting Practices Manual

LANGUAGE	Conversion Ratio (LOC Per Function Point)
Basic Assembly	575
JCL	400
Macro Assembly	400
C	225
Cobol 74 (Cobol I)	220
FORTRAN	210
Cobol 85 (Cobol II)	175
Pascal	160
PL/1	126
RPG I	120
RPG II/III	110
Natural	100
C++	80

LANGUAGE	Conversion Ratio (LOC Per Function Point)
Java	80
dBase III	60
Focus	60
Clipper	60
Oracle	60
Sybase	60
dBase IV	55
Perl	50
JavaScript	50
VBScript	50
Shell Script	50
SAS	50
APL	50

Higher Level Languages are not included because of their significantly high divergent rates.

***NOTE: Differences between DCG's ratio and other published conversion rates are based on the changes in counting rules from early 1990s. DCG's Conversion Ratios has made those adjustments.**

Other Software Sizing Options

MEASURE	ADVANTAGES	DISADVANTAGES
Mark II Function Points	<ul style="list-style-type: none"> • Technology independent • Logical user view (Minimally supports ISO Functional Sizing Method) 	<ul style="list-style-type: none"> • Acceptance isolated use only in UK • Proprietary (until recently) • Dependent on implementation
Lines of Code (KLOC)	<ul style="list-style-type: none"> • Easy to compute • Can be done from physical implementation • Intuitive for developers • Granular • Natural byproduct of process 	<ul style="list-style-type: none"> • Technology dependent (does not support ISO Functional sizing method) • Inconsistent rules • Counter indicator of productivity • Unavailable until coded • Penalizes efficient code • Difficult to make valid comparisons across languages
Feature Points (Capers Jones - early 1990's)	<ul style="list-style-type: none"> • Enhanced FP for algorithmically intensive 	<ul style="list-style-type: none"> • Experimental – no statistical comparison to products • Not supported by the sizing community • No longer endorsed by Capers Jones, as of 1996)

IFUG Current Activities



- International Function Point User Group, publishes book with over 40 authors: IT Measurement: Practical Advice from the Experts, Addison Wesley, April, 2002
- ISO approves IFPUG CPM 4.1 Unadjusted as Publically Available Standard (PAS), ISO #20926

IFPUG Current Activities

- Establishing new Certification in Software Measurement
- Establishing Advanced Certified Function Point Specialist Certification
- Expanding Product Size vs. Requirement Size
- Continued support of International Standards Benchmarking Group (ISBGS)

Current Activities

- CPC New White Papers:
 - Practical Guidelines in Counting Logical Files
 - Practical Guidelines in Counting Code Data
 - Practical Guidelines in Counting Enhancements
 - Practical Guidelines in Counting Shared Data
 - Framework for Functional Sizing
- CPM will be released in 2004 to include latest guidelines and be in ISO format

Summary

- IFPUG Functional Sizing Method celebrates 25 years of usage in 2004
- Significant statistics continue to validate the model as the world standard for sizing software

Contact Information

- Koni Thompson Houston
 - The David Consulting Group
 - www.davidconsultinggroup.com
 - Koni_thompson@msn.com
 - 770-529-5753
- IFPUG
 - www.ifpug.org
- ISBGS
 - www.isbgs.org