Function points (FP)

Meaningful Metrics

Developed in the late 1970s by Allen Albrecht as a means of measuring software size in terms that are meaningful to end users, function points have become an international standard used tomeasure software size (ISO 20926:2003).

Function points are a logical size measure (as opposed to a physical size measure like lines of code or objects).

• Function points measure software size based on the functionality requested by and provided to the end user.

• Function points are derived from requirements and are applicable for measurement throughout the entire development life cycle.

• Function points are comprised of inputs, outputs, inquiries, internal data, and external interface data.

• Technology, platform, language independent

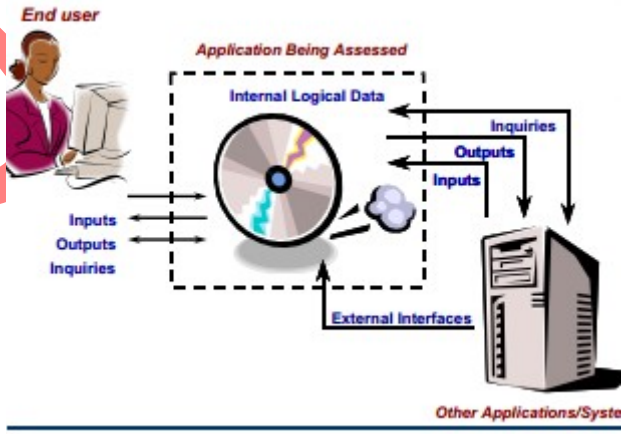
• Excellent as a basis of estimate for development cost and schedule

• Provides ability to normalize metrics for consistent analysis

• Fully documentable and repeatable

• Assists with requirements management; functional size traceable throughout entire life cycle

• Provides quantitative basis for earned value management



Advantages

Technology, platform, language independent

• Excellent as a basis of estimate for development cost and schedule

• Provides ability to normalize metrics for consistent analysis

• Fully documentable and repeatable

• Assists with requirements management; functional size

traceable throughout entire life cycle

• Provides quantitative basis for earned value management

Example Metrics

Effort per function point (productivity)

Defects per function point (quality)

Cost per function point (cost efficiency)

Why use Function Points

- Managing Your Software
- Managing Your Organization
- Function Points vs. Lines of Code

How to Count Function Points

Objectives of Function Point Analysis

Measures software by quantifying the functionality requested by and provided to the customer based primarily on logical design.

Measures software development and maintenance independently of technology used for implementation.

Measures software development and maintenance consistently across all projects and organizations.

Function Points are a Unit of Measure

(fig s6)

Why Use Function Points : Managing Your Software

Software Development Challenges

- Size of Requirements
- Changes to Requirements
- Estimation Based on Requirements
- Measuring and Improving Productivity and Quality

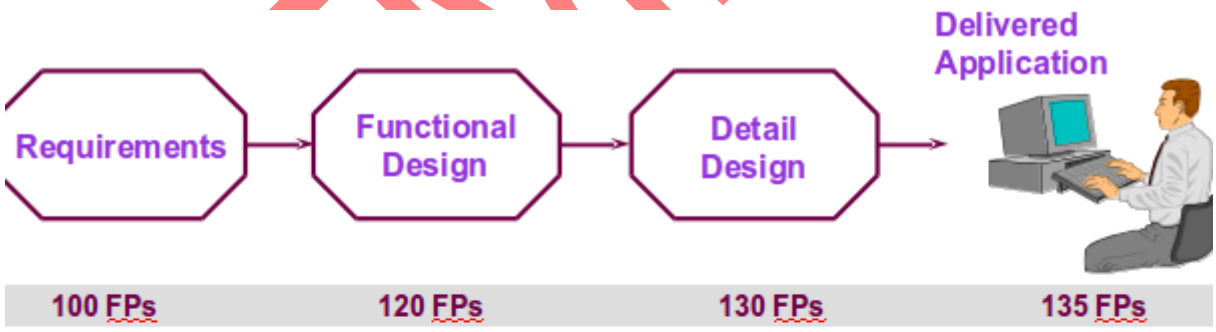
Size of Requirements

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

Changes to Requirements

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

Changes to Requirements (fig s11)



Estimation Based on Requirements

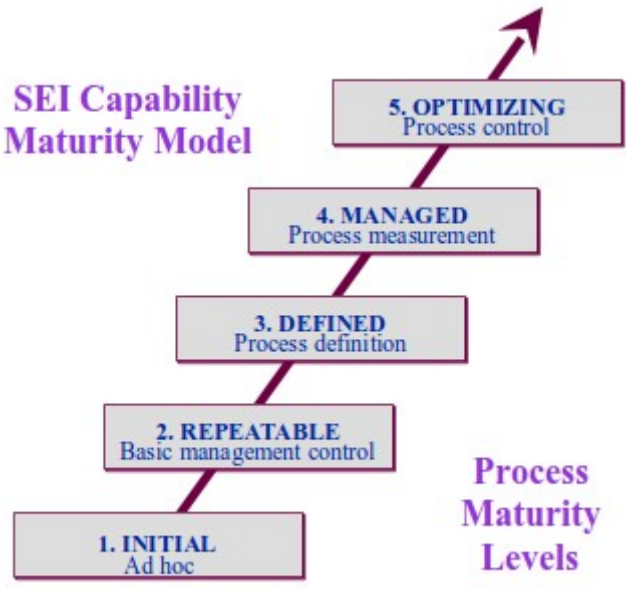
- Multiple Models
- Weighted Inputs:
  - Language
  - Skills
  - Methodology
  - Risk Factors
  - Size
- Historical Base

Estimating Examples Measuring and Improving Productivity (s13,s14,s15,s16)

Function Point Size	Project Variables	Project Estimate Based on Historical Data and/or Vendor Tool
Project A – 100 FPs	• On-line/database • New developm • C++ • Higly experienced development staff	Effort=5 months Schedule=3 months Cost(@$5K)=$25,000 KLOC=6 Delivered Defects=25 Productivity Rate= 20 FP/Months
Project B -100 FPs	Batch Enhancment Cobol Average experienced development staff	Effort=20 months Schedule=6 months Cost(@$5K)=$100,000 KLOC=10 Delivered Defects=10 Productivity Rate= 5 FP/Months

Why Use Function Points Asset Management Function Points and the CMM

Function Points are the metric of choice for many of the activities required in the SEI CMM Level 2 2 (s17)



With the next release of the CMM, metrics becomes a Key Process Area in its own right

Improving Customer Relations

- Predictable Time scales
- Predictable Costs
- Predictable Functionality

Organizational Improvement

- Process Measurement
- Project Management Metrics
  - Estimates
  - Productivity
  - Defect Densities
  - etc.
- Benchmarking

Function Points not Lines of Code

- Technology and platform independence
- Available from early requirements phase
- Consistent and objective unit of measure throughout the life cycle
- Objectively defines software application from the customer perspective
- Objectively defines a series of software applications from the customer's, not the technician's perspective
- Is expressed in terms that users can readily understand about their software

What is Wrong with Lines of Code?

- There is no standard for a line of code
- Lines of Code measure components not completed products
- Don't measure the panels produced; measure the number of cars assembled
- Measuring lines of code
- Rewards profligate design
- Penalizes tight design

- Positively misleading?

Classic Productivity Paradox

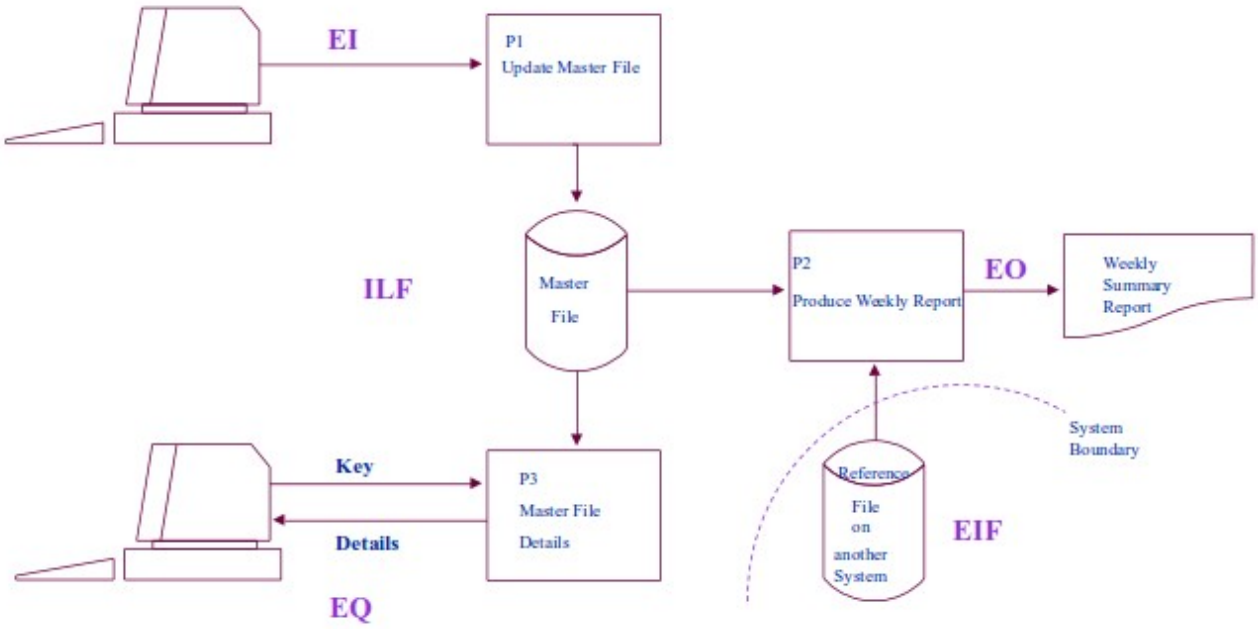
Lines of Code	10,000	3,000
Function Point	25	25
Total Month effort	25	15
Total Costs	$125,000	$75,000
Cost per Source Line	$12,50	$25
Lines per Person month	400	200
Fps per Person month	1.2	2
Cost per FP	$5,000	$3,000

How to Count Function Points

Steps in FP Counting

- Determine Type of Count
- Identify Counting Scope and Application Boundary
- Count Data Functions
- Count Transactional Functions
- Determine Unadjusted Function Point Count
- Determine Value Adjustment Factor
- Calculate Adjusted Function Point Count

FP Overview: What Is Counted (s27)



Data Storage

- 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)

Transactions

- External Input (EI)
  - Maintains ILF or passes control data into the application
- External Output (EO)
  - Formatted data sent out of application with added value (e.g., calculated totals
- External Query (EQ)
  - Formatted data sent out of application without added value

Functional Size (Unadjusted Function Size)

Function Type	Low	Average	High
EI	**x3**	**x4**	**x6**
EO	**x4**	**x5**	**x7**
EQ	**x3**	**x4**	**x6**

ILF	**x7**	**x10**	**x15**
EIF	**x5**	**x7**	**x10**

Value Adjustment Factor

Based on 14 General System Characteristics (User Business Constraints Independent of Technology)

Examples: data communications, response times, end user efficiency, multiple sites and flexibility

Adjusts FP count by up to + / - 35%

History - Early Days

1979 Function Points introduced by Allan Albrecht

1984 First formal Function Point Guidelines

1986 IFPUG elects first Board of Directors

1990 Function Point as Assets Manual

1991 Certification for training materials

tabela QSM

- Business Systems (Source Lines of Code benchmarks) - See more at: http://www.qsm.com/resources/performance-benchmark-tables#Business Function Point
- Business Systems: Function Point Benchmarks
- Engineering Systems
- Real Time Systems

Measures:

•Schedule: elapsed time (in months) from Requirements Determination (Phase 2) through the Initial Release (end of Phase 3)
•Schedule = (P2 Duration + P3 Duration) – P2 Overlap
•Effort: the number of Person Months expended during Requirements Determination (Phase 2) and Construct & Test (Phase 3)
•Effort = P2 PM + P3 PM
•Average Staff: the number of Full Time Equivalent employees for Phase 2 and Phase 3
•Average Staff = (P2 + P3 Effort)/ (P2+ P3 Duration)
•SLOC/ PM: the number of Source Lines of Code produced per Person Month of effort during Phase 3
•FP/ PM: the number of Function Points produced per Person Month during Phase 3

2,500

Language

QSM SLOC/FP Data

Avg Median Low High

ABAP (SAP) * 28 18 16 60

ASP* 51 54 15 69

Assembler * 119 98 25 320

Brio + 14 14 13 16

C * 97 99 39 333

C++ * 50 53 25 80

C# * 54 59 29 70

Excel * 209 191 131 315

Focus * 43 45 45 45

FoxPro 36 35 34 38

HTML * 34 40 14 48

J2EE * 46 49 15 67

Java * 53 53 14 134

JavaScript * 47 53 31 63

.NET * 57 60 53 60

Oracle * 37 40 17 60

PACBASE * 35 32 22 60

Perl * 24 15 15 60

SQL * 21 21 13 37

VB.NET * 52 60 26 60

Visual Basic * 42 44 20 60

The Strengths of Function Point Metrics

1. IFPUG function point metrics have more measured projects than all other metrics combined.
2. IFPUG function point metrics are endorsed by ISO/IEC standard 20926:2009.
3. Formal training and certification examinations are available for IFPUG function point counting.
4. Hundreds of certified IFPUG function point counters are available in most countries.
5. Counts of function points by certified counters usually are within 5% of each other.
6. IFPUG function point metrics are standard features of most parametric estimating tools such as KnowledgePlan, SEER, and Software Risk Master.
7. Function points are increasingly used for software contracts. The government of Brazil requires function points for all software contracts.

The Weaknesses of Function Point Metrics

1. Function point analysis is slow. Counting speeds for function points average perhaps 500 function points per day.
2. Due to the slow speed of function point analysis, function points are almost never used on large systems > 10,000 function points in size.
3. Function point analysis is expensive. Assuming a daily counting speed of 500 function points and a daily consulting fee of $1,500 counting an application of 10,000 function points would require 20 days and cost $30,000. This is equal to a cost of $3.00 for every function point counted.
4. Application size is not constant. During development applications grow at perhaps 2% per calendar month. After development applications continue to grow at perhaps 8% per calendar year. Current counting rules do not include continuous growth.
5. More than a dozen function point counting variations exist circa 2013 including COSMIC function points, NESMA function points, FISMA function points, fast function points, backfired function points, and a number of others. These variations produce function point totals that differ from IFPUG function points by perhaps + or – 15%.

Software Risk Master tool
Increasing Executive Awareness of Function Points for Economic Studies

Because of the slow speed of function point analysis and the lack of data from large applications function points are a niche metric below the interest level of most CEO's and especially CEO's of Fortune 500 companies with large portfolios and many large systems including ERP packages. In order for function point metrics to become a priority for C level executives and a standard method for all software contracts, some improvements are needed:

1. Function point size must be available in a few minutes for large systems; not after weeks of counting.
2. The cost per function point counted must be lower than $0.05 per function point rather than today's costs of more than $3.00 per function point counted.
3. Function point metrics must be able to size applications ranging from a low of 1 function point to a high of more than 300,000 function points.
4. Sizing of applications must also deal with the measured rates of requirements creep during development and the measured rates of post-release growth for perhaps 10 years after the initial release.
5. Function points must also be applied to maintenance, enhancements, and total costs of ownership (TCO).
6. Individual changes in requirements should be sized in real-time as they occur. If a client wants a new feature that may be 10 function points in size, this fact should be established within a few minutes.
7. Function points should be used for large-scale economic analysis of methods, industries, and even countries.
8. Function points should be used to measure consumption and usage of software as well as production of software.
9. Function points must be applied to portfolio analysis and system architecture.
10. Function points can and should be used to size and estimate collateral materials such as documentation, test case volumes, and reusable materials used for applications.

	1980 - 1989	1990 - 1999	2000 - 2009	2010 - present
Schedule (months)	17.8	11.4	10.2	11.7
Effort (person hours)	18,019	13,541	8,658	11,414
Team size (FTE staff)	7.3	6.7	6.7	6.9
New/Modified Code (KESLOC)	75.3	58.8	36.2	29.6
Productivity Index	14.7	16.2	12.9	13.2
Primary Language	COBOL	COBOL	JAVA	JAVA
Industry Sector	Financial	Utilities	Manufacturing	Financial