Function Point Analysis

Introduction

- Function point metrics, developed by Alan Albercht of IBM, were first published in 1979
- In 1984, the International Function Point Users Group (IFPUG) was set up to clarify the rules, set standards, and promote their use and evolution

Introduction (Cont’d)

- Function point metrics provide a standardized method for measuring the various functions of a software application.
- Function point metrics, measure functionality from the users point of view, that is, on the basis of what the user requests and receives in return

Objectives of FPA

- Function point analysis **measures software by quantifying the functionality** the software provides to the user based primarily on logical design. With this in mind, the objectives of function point analysis are to:
  - Measure functionality that the user requests and receives
  - Measure software development and maintenance independently of technology used for implementation
- In addition to meeting the above objectives, the process of counting function points should be:
  - Simple enough to minimize the overhead of the measurement process
  - A consistent measure among various projects and organizations

There are several other important notes about the FPA process that need to be introduced at this time:

- **Measured from the user's perspective** The size of the application being measured is based on the user's view of the system. It is based on what the user asked for, not what is delivered. It's based on the way the user interacts with the system, including the screens that the user uses to enter input, and the reports the users receive as output. Finally, it's also based on their understanding of the data that needs to be stored and processed by the system.
- **Technology-independent** It does not matter what technology you are using to implement your application. It doesn't matter if it's a Web application written in Java, PHP, ColdFusion, or .Net; or a client-server app written in Java, Delphi, VB; or even an AS/400 RPG application. Just show me your screens and your data tables and I'll derive "number of function points" from there.
- **Low cost** Adding FPA to your software development portfolio is also very easy. Historically, adding the process of counting FPs to your development process results in a cost increase of only 1%.
- **Repeatable** Studies have shown that multiple function point counters can independently count the same application to within 10% accuracy of each other. Repeatability is very important, because without it we could not begin to trust the data from the hundreds of applications that are stored in repositories around the world.
- **Work well with use cases** This process works extremely well with use cases, and can even work with the concept of "stories" in Extreme Programming.

Benefits of FPA

- Organizations can apply function point analysis as:
  - A tool to determine the size of a purchased application package by counting all the functions included in the package
  - A tool to help users determine the benefit of an application package to their organization by counting functions that specifically match their requirements
  - A tool to measure the units of a software product to support quality and productivity analysis
  - A vehicle to estimate cost and resources required for software development and maintenance
  - A normalization factor for software comparison

- FP Counting Process involves the following steps −
- **Step 1** − Determine the type of count.
- **Step 2** − Determine the boundary of the count.
- **Step 3** − Identify each Elementary Process (EP) required by the user.
- **Step 4** − Determine the unique EPs.
- **Step 5** − Measure data functions.
- **Step 6** − Measure transactional functions.
- **Step 7** − Calculate functional size (unadjusted function point count **UFPC**).
- **Step 8** − Determine Value Adjustment Factor (**VAF**).
- **Step 9** − Calculate adjusted function point count.

Determine the Type of Count

There are three types of function point counts −

- **Development Function Point Count**: associated with new development work and may include the prototypes, which may have been required as temporary solution, which supports the conversion effort.
- **Application Function Point Count:** exclude any conversion effort (prototypes or temporary solutions) and existing functionality that may have existed.
- **Enhancement Function Point Count:** When changes are made to software after production, they are considered as enhancements

FPA Overview

- Counting boundary: The border between the application or project being measured and external applications or the user domain.
- A boundary establishes ***which functions are included in the function point count***

Identify Each Elementary Process Required by the User

- Compose and/or decompose the ***functional user requirements into the smallest unit of activity***, which satisfies all of the following criteria –
- Is meaningful to the user.
- Constitutes a complete transaction.
- Is self-contained.
- Leaves the business of the application being counted in a consistent state.

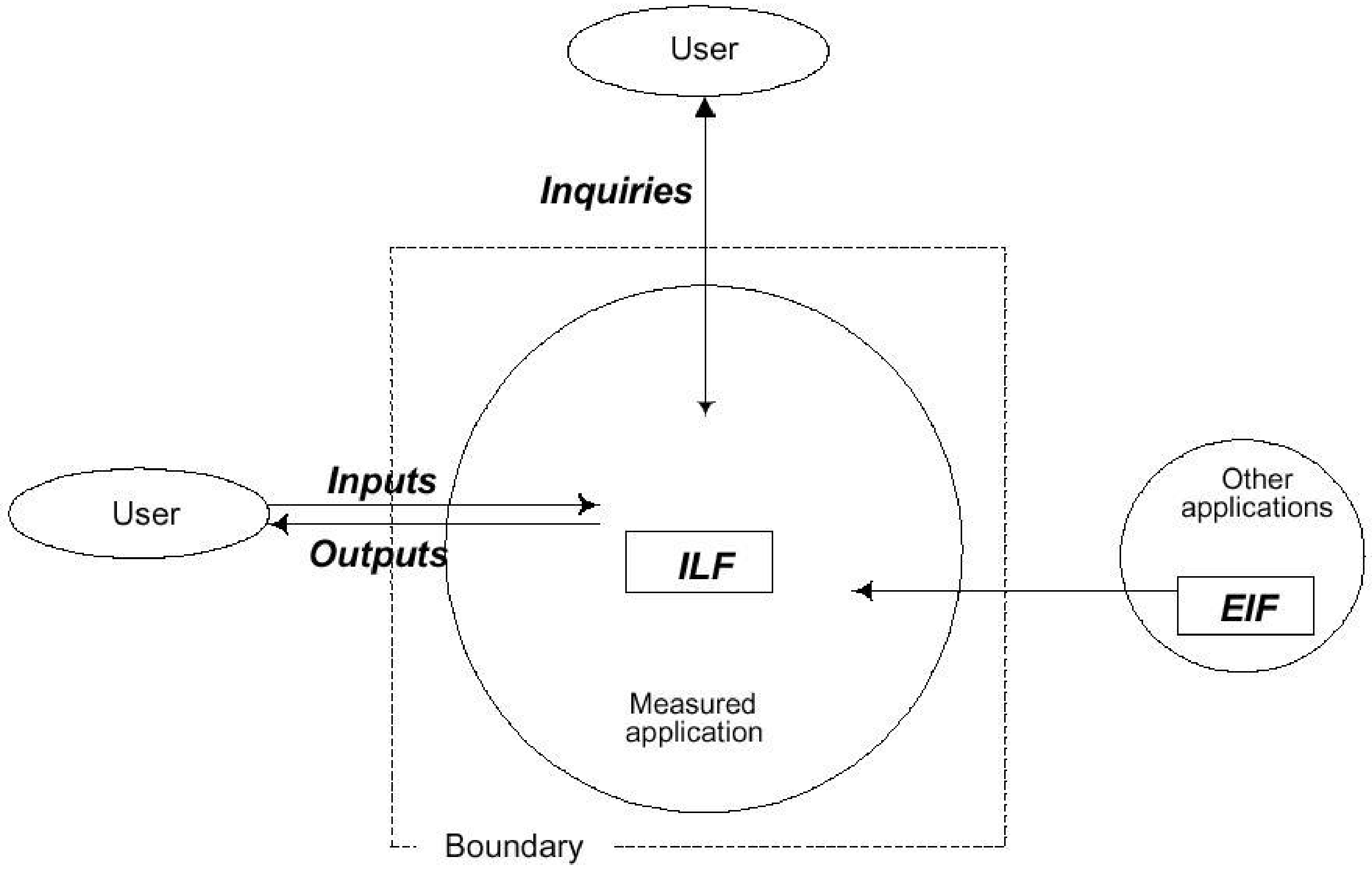
For example, the Functional User Requirement – ***"Maintain Employee information"*** can be decomposed into smaller activities such as ***add employee, change employee, delete employee***, and inquire about employee.

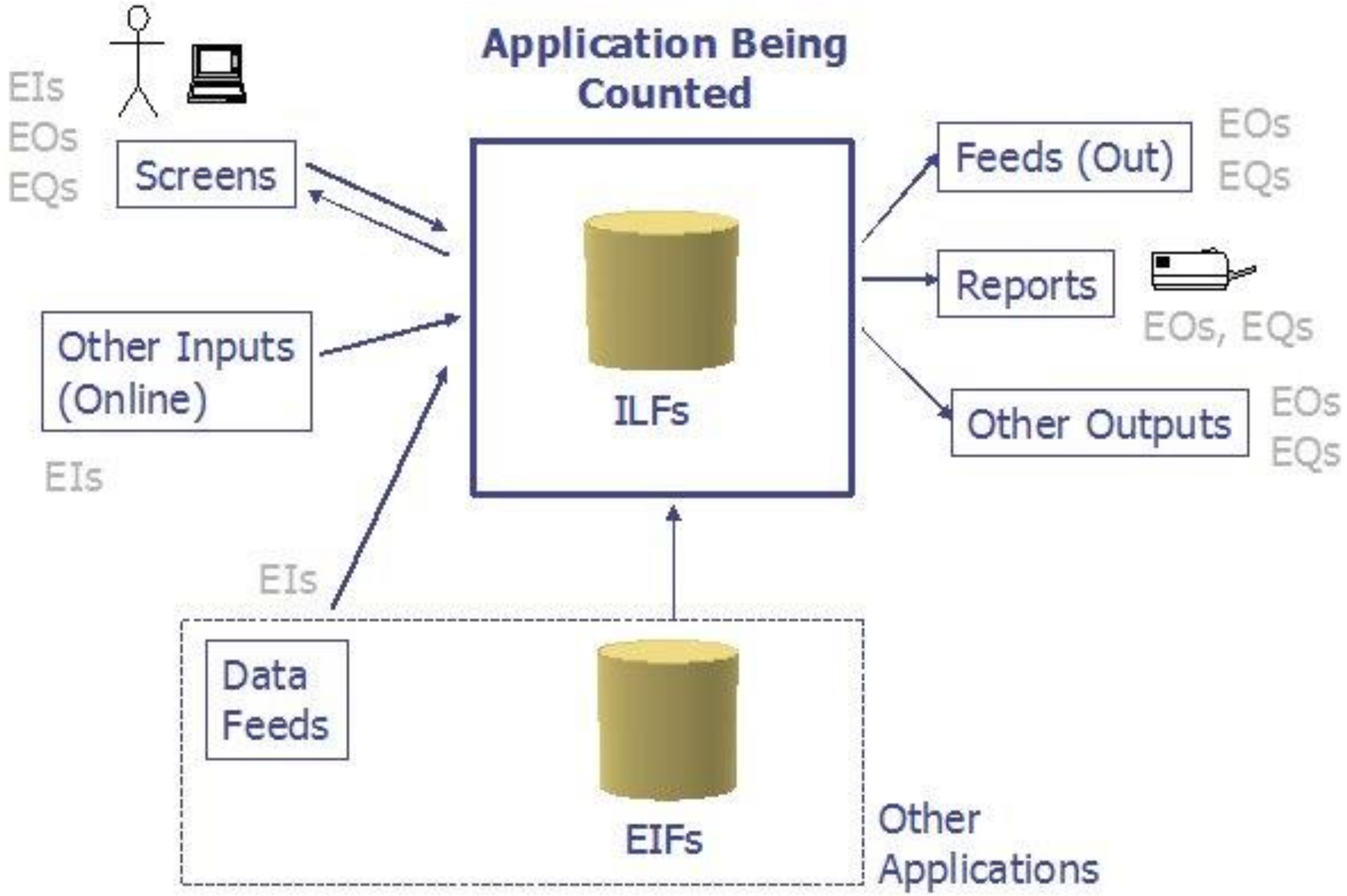
- Each unit of activity thus identified is an Elementary Process (EP).

Determine the Unique Elementary Processes

- Comparing two EPs already identified, ***count them as one EP*** (same EP) if they –
- Require the same set of DETs.
- Require the same set of FTRs.
- Require the same set of processing logic to complete the EP.
- Do not split an EP with multiple forms of processing logic into multiple Eps.
- For e.g., if you have identified ‘Add Employee’ as an EP, it should not be divided into two EPs to account for the fact that an employee may or may not have dependents. The EP is still ‘Add Employee’, and there is variation in the processing logic and DETs to account for dependents.

FPA Components




Application Being Counted
EIs
EOs
EQs
Screens
Other Inputs (Online)
EIs
EIs
Data Feeds
ILFs
EIFs
Other Applications
Feeds (Out)
EOs
EQs
Reports
EOs, EQs
Other Outputs
EOs
EQs

FPA Overview (Cont’d)

- The next step is determining the unadjusted function point (UFP) count
- UFP reflects the specific countable functionality provided to the user by the project or application

Some Terms

- ***DET***: a *data element type* is a unique user recognizable, non-repeated ***field.*** For example, an ***account number*** that is stored in multiple fields is counted as one DET.
- ***RET***: A *record element type* is a user recognizable ***subgroup of data elements*** within an ILF or EIF. E.g personal info.

Some Terms (Cont’d)

- ***FTR:*** A *file type referenced* is
  - An internal logical file read or maintained by a transactional function or
  - An external interface file read by a transactional function

Calculation of the UFP

- This calculation begins with the counting of the five function types of a project or application:
  - Two data function types
  - Three transactional function types

Data Function Types

- **Internal Logical File (ILF)**: a user identifiable group of logically related data or control information maintained within the boundary of the application. OR
- ILFs represent data that is *stored and maintained within the boundary of the application* you are counting.

- **External Interface File (EIF)**: a user identifiable group of logically related data or control information referenced by the application, but maintained within the boundary of another application. OR
- EIFs represent the data that your application will use/reference, but *data that is not maintained by your application*.
  - This means that EIF counted for an application, must be an ILF in another application

Examples of ILFs :

- Tables in a relational database.
- Flat files.
- Application control information, perhaps things like user preferences that are stored by the application.
- LDAP data stores.
- **Example of EIF:**

it may be necessary for a pilot to reference position data from a satellite or ground-based facility during flight. The pilot does not have the responsibility for updating data at these sites but must reference it during the flight. Groupings of data from another system that are used only for reference purposes are defined as External Interface Files (EIF).

Transactional Function Types

- ***External Input (EI)***: An EI processes data or control information that **comes from outside the application’s boundary**. The EI is an elementary process.
  - Elementary process: The smallest unit of activity that is meaningful to the end user in the business

- EI allows a user to maintain Internal Logical Files (ILFs) through the ability to add, change and delete the data.
- For example, a ***pilot can add, change and delete navigational information prior to and during the mission***. In this case the pilot is utilizing a transaction referred to as an External Input (EI). An External Input gives the user the capability to maintain the data in ILF's through adding, changing and deleting its contents.

Transactional Function Types

- ***External Output (EO)***: An EO is an elementary process that generates data or control information sent outside the application’s boundary.

- gives the user the ability to produce outputs. The primary intent of an external output is to ***present information to a user through processing logic*** other than, or in addition to, the retrieval of data or control information . The processing logic must contain at least one ***mathematical formula or calculation***, create derived data maintain one or more ILFs or alter the behavior of the system.

- For example a pilot has the ability to separately display ground speed, true air speed and calibrated air speed. The results displayed are derived using data that is maintained and data that is referenced

- ***External Inquiry (EQ)***: An EQ is an elementary process made up of an *input-output* combination that results in data retrieval.
- In this situation there is ***no manipulation of the data***. It is a ***direct retrieval*** of information contained on the files. For example if a pilot displays terrain clearance data that was previously set, the resulting output is the direct retrieval of stored information. These transactions are referred to as External Inquiries (EQ).

FPA Overview (Con’d)

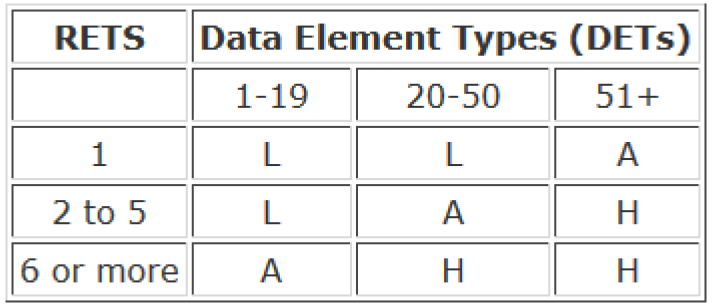
- These 5 function types are then ranked according to their complexity: Low, Average or High, using a set of prescriptive standards.
- Organizations that use FP methods, develop criteria for determining whether a particular entry is Low, Average or High.
- Nonetheless, the determination of complexity is somewhat subjective.

FPA Overview (Con’d)

- After classifying each of the five function types, the UFP is computed using predefined weights for each function type

Measure Data Functions

1. **Count the DETs for Each Data Function**
2. **Count the RETs for Each Data Function**
3. **Determine the Functional Complexity for Each Data Function**



RETS	Data Element Types (DETs)		
	1-19	20-50	51+
1	L	L	A
2 to 5	L	A	H
6 or more	A	H	H

4. **Measure the Functional Size for Each Data Function**

Complexity	Points
Low	7
Average	10
High	15

Table 3: ILF weights

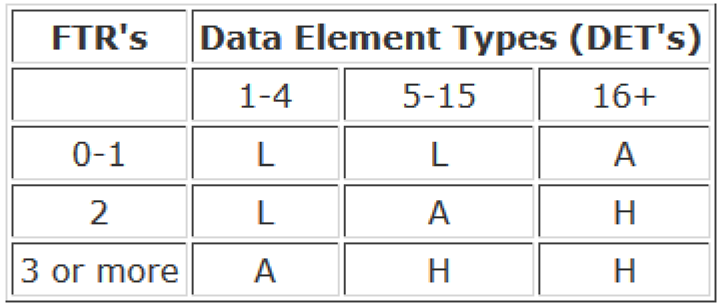
EIF Weights:

Value	No. or Function Points
Low	5
Average	7
High	10

Measure Transactional Functions

First classify each Transactional Function

1. **Count the DETs for Each Transactional Function**
2. **Count the FTRs for Each Transactional Function**
3. **Determine the Functional Complexity for Each Transactional Function**



FTR's	Data Element Types (DET's)		
	1-4	5-15	16+
0-1	L	L	A
2	L	A	H
3 or more	A	H	H

4. **Measure the Functional Size for Each Transactional Function**

Complexity	Points/Weight
Low	3
Average	4
High	6

Table 7: EI weights

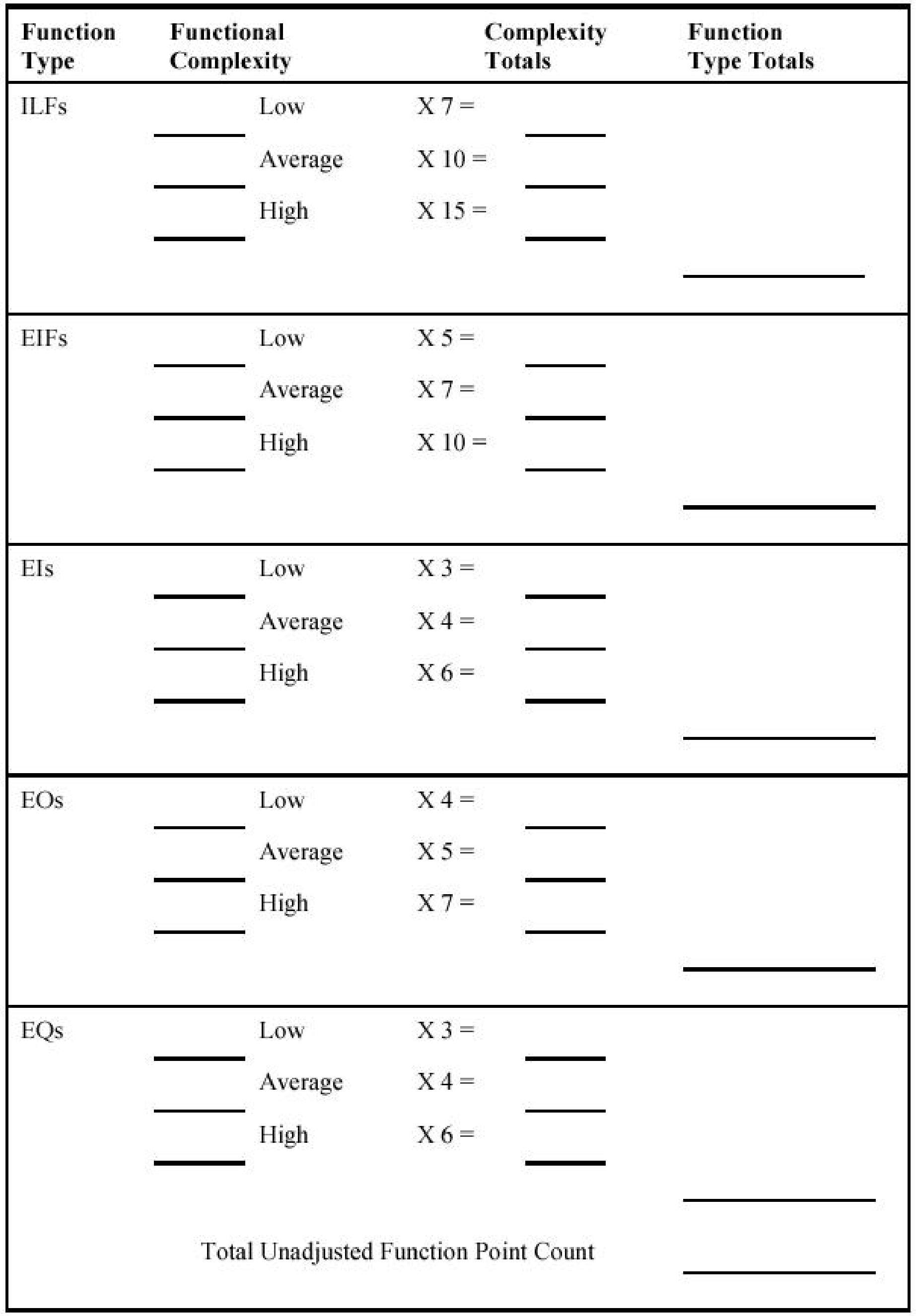
Complexity	Points/Weight
Low	4
Average	5
High	7

Table 9: EO weights

Complexity	Points/Weight
Low	3
Average	4
High	6

Table 11: EQ weights

UFP Calculation Table



Function Type	Functional Complexity			Complexity Totals	Function Type Totals
ILFs		Low	X 7 =		
		Average	X 10 =		
		High	X 15 =		
EIFs		Low	X 5 =		
		Average	X 7 =		
		High	X 10 =		
EIs		Low	X 3 =		
		Average	X 4 =		
		High	X 6 =		
EOs		Low	X 4 =		
		Average	X 5 =		
		High	X 7 =		
EQs		Low	X 3 =		
		Average	X 4 =		
		High	X 6 =		
	Total Unadjusted Function Point Count				

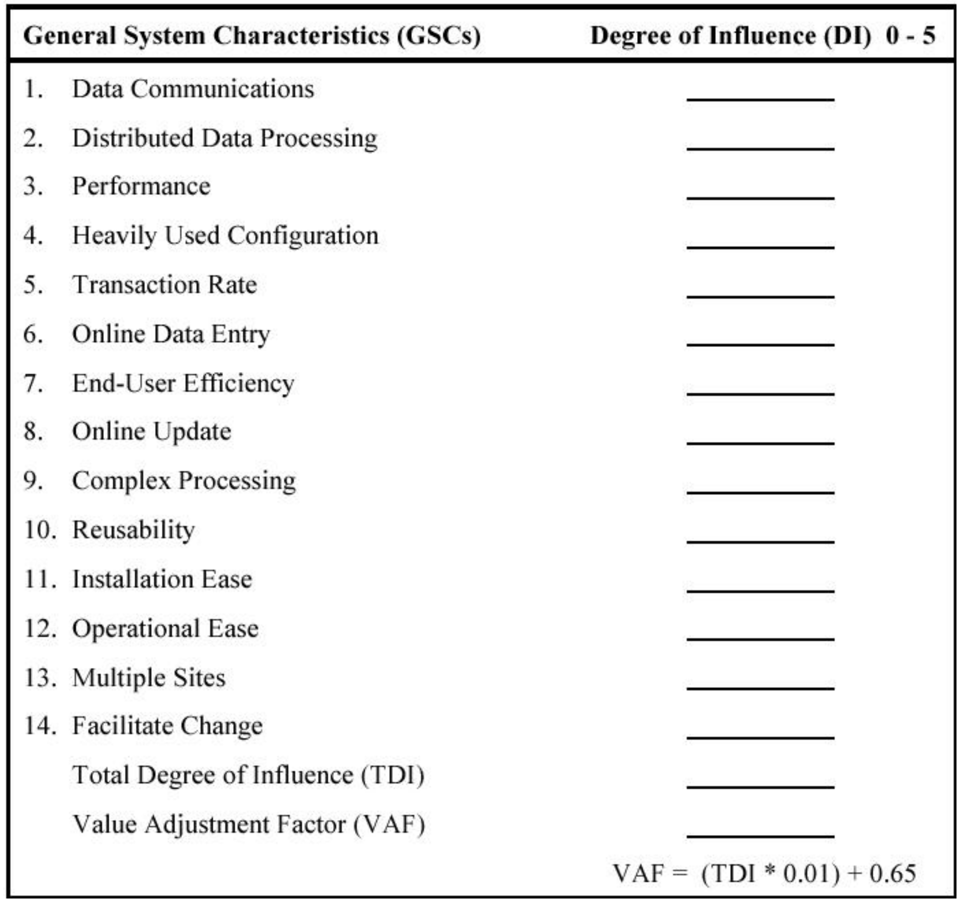
FPA Overview (Con’d)

- The last step involves assessing the environment and processing complexity of the project or application as a whole.
- In this step, the impact of 14 general system characteristics is rated on a scale from 0 to 5 in terms of their likely effect on the project or application

Value Adjustment Factor (VAF)

Calculation Table

- 0 = No Influence
- 1 = Incidental
- 2 = Moderate
- 3 = Average
- 4 = Significant
- 5 = Essential



General System Characteristics (GSCs)	Degree of Influence (DI) 0 - 5
1. Data Communications	________
2. Distributed Data Processing	________
3. Performance	________
4. Heavily Used Configuration	________
5. Transaction Rate	________
6. Online Data Entry	________
7. End-User Efficiency	________
8. Online Update	________
9. Complex Processing	________
10. Reusability	________
11. Installation Ease	________
12. Operational Ease	________
13. Multiple Sites	________
14. Facilitate Change	________
Total Degree of Influence (TDI)	________
Value Adjustment Factor (VAF)	________

VAF = (TDI * 0.01) + 0.65

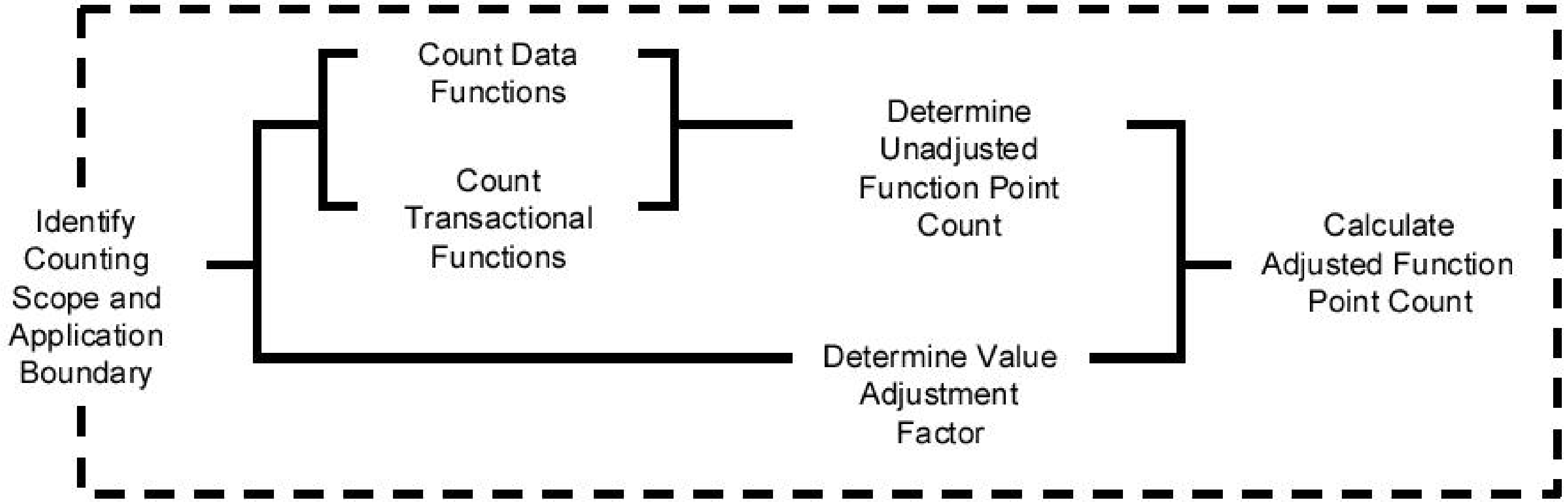
FPA Overview (Con'd)

- On the whole:

$$FP = UFP \times VAF$$

- The constant values in the equation and the weighting factors are determined empirically

FPA Procedure at a Glance

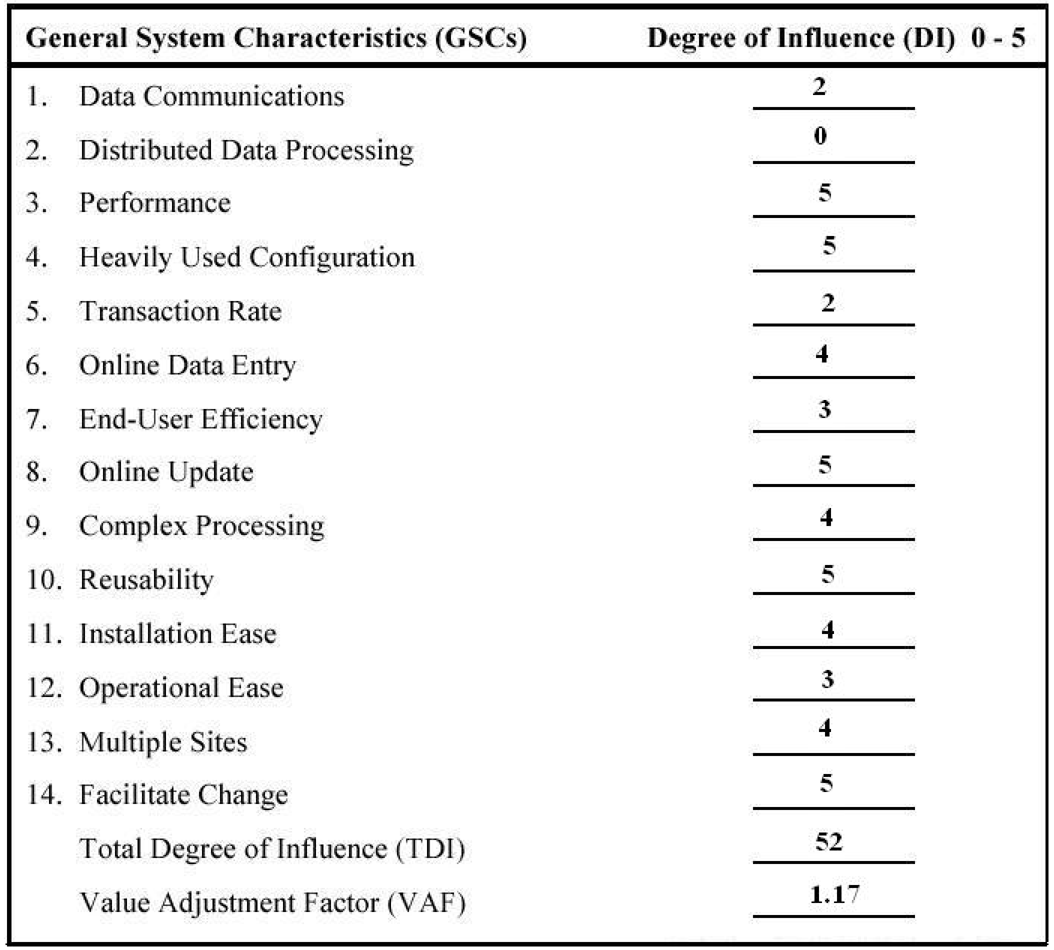

Identify Counting Scope and Application Boundary
Count Data Functions
Count Transactional Functions
Determine Unadjusted Function Point Count
Determine Value Adjustment Factor
Calculate Adjusted Function Point Count

An Example

Function Type	Estimated Count	Weight	FP-Count
EI	24	(Average) 4	96
EO	16	(Average) 5	80
EQ	22	(Average) 4	88
ILF	4	(Average) 10	40
ELF	2	(Average) 7	14
UFP count			**318**

An Example (Cont’d)

$\text{VAF} = 52 * 0.01 + 0.65$
$= 1.17$
$FP_{\text{estimated}} = 318 \times 1.17$
$= 372$



General System Characteristics (GSCs)	Degree of Influence (DI) 0 - 5
1. Data Communications	2
2. Distributed Data Processing	0
3. Performance	5
4. Heavily Used Configuration	5
5. Transaction Rate	2
6. Online Data Entry	4
7. End-User Efficiency	3
8. Online Update	5
9. Complex Processing	4
10. Reusability	5
11. Installation Ease	4
12. Operational Ease	3
13. Multiple Sites	4
14. Facilitate Change	5
Total Degree of Influence (TDI)	52
Value Adjustment Factor (VAF)	1.17

Function Point Analysis

A simple example:

inputs

3 simple X 2 = 6

4 average X 4 = 16

1 complex X 6 = 6

outputs

6 average X 5 = 30

2 complex X 7 = 14

files

5 complex X 15 = 75

inquiries

8 average X 4 = 32

interfaces

3 average X 7 = 21

4 complex X 10 = $\underline{40}$

Unadjusted function points 240

Function Point Analysis

In addition to these individually weighted function points, there are factors that affect the project and/or system as a whole. There are a number (~35) of these factors that affect the size of the project effort, and each is ranked from “0”- no influence to “5”- essential.

The following are some examples of these factors:

- Is high performance critical?
- Is the internal processing complex?
- Is the system to be used in multiple sites and/or by multiple organizations?
- Is the code designed to be reusable?
- Is the processing to be distributed?
- and so forth . . .

Function Point Analysis

Continuing our example . . .

Complex internal processing	= 3
Code to be reusable	= 2
High performance	= 4
Multiple sites	= 3
Distributed processing	= $\underline{5}$
Project adjustment factor	= 17

Adjustment calculation:

Adjusted FP = Unadjusted FP X [0.65 + (adjustment factor X 0.01)]

= 240 X [0.65 + (17 X 0.01)]

= 240 X [0.82]

= 197 Adjusted function points

Function Point Analysis

But how long will the project take and how much will it cost?

- As previously measured, programmers in our organization average 18 function points per month. Thus . . .

  197 FP divided by 18 = 11 man-months

- If the average programmer is paid $5,200 per month (including benefits), then the [labor] cost of the project will be . . .

  11 man-months X $5,200 = $57,200

Problems of FPA

- FPA has been criticized as not being universally applicable to all types of software.
    - For example, FPA doesn’t capture all functional characteristics of real-time software

Other Variants of FPA

- FP was originally designed to be applied to business information systems applications.
  - So, the data dimension was emphasized.
    - So, FPA was inadequate for many engineering and embedded systems.

Other Variants of FPA (Cont’d)

- ***Feature Point***
  - Is a superset of FP
  - Suitable for real-time, process-control and embedded software applications tend to have high algorithmic complexity
  - This method counts a new software characteristics: “*algorithms*”

Other Variants of FPA (Cont’d)

- ***3D Function Point***
  - Developed by Boeing
  - Suitable for applications that emphasize function and control capabilities:
    - *Data dimension*: very similar to basic FP
    - *Functional dimension*: is measured by considering the number of internal operations required to transform input to output data
    - *Control dimension*: is measured by counting the number of transitions between states.
  - Characteristics of all 3 dimensions are counted, quantified and transformed into a measure that provides an indication of the functionality

Other Variants of FPA (Cont’d)

- ***MK II FPA***
  - Developed in the late 80’s by Charles Symons in the UK
- ***NESMA***
  - Is a simpler-to-use variant of the IFPUG method
  - Maintained by the Netherlands software metrics association

Other Variants of FPA (Cont'd)

- ***COSMIC-FPP***
  - Is approved by ISO
  - Designed to measure the functional size of real-time, multi-layered software such as used in telecoms, process control and operating systems as well as business applications, all on the same measurement scale
  - Having been developed in the last few years, the method is compatible with modern specification methods such as UML and OO techniques.